

1 Introduktion

Ordklasstagning är en vanlig grundtillämpning inom språkteknologin. En manuellt annoterad korpus förväntas åtminstone ha information om ordklass och många typer av parsers kan ta sin grund i tvetydiga eller unikt tilldelade taggar ur vilka man sedan kan bilda en hierarkisk satsstruktur. Vidare kan även andra språkteknologiska system, där en fullständig parsning är svår eller onödig, vara betjänta av den grundläggande syntaktiska information ordklasstaggar innebär. Exempel här kan vara dialogsystem, korrekt intonation och betoning i talsyntes från skrivna manus, stavnings- och grammatikkontroll samt informationsextraktion och -sökning.

Vissa av dessa tillämpningar kräver, som redan nämnts, entydiga taggar, medan andra kan ta information om ett antal kandidater, med någon typ av rangordning eller sannolikhet för dessa.

Det finns ett antal olika paradigmer inom området, där utvecklingen har gått mot datadrivna metoder. Det har helt enkelt visat sig mer effektivt att bygga upp stora korpusar med annotation och sedan konstruera relativt generella system som utifrån en korpus automatiskt kan utforma en språkmodell. Samma system kan då i idealfallet utan förändring användas på korpusar för olika genrer och språk. Samtidigt får man inte förneka att alla system innehåller förmodanden och optimeringar baserade på det språk som analyserats under utvecklandet av systemen.

Inom datadrivna metoder finns det ett antal olika strategier, såsom minnesbaserade och transformationsbaserade. En tredje är dolda Markov-modeller, där en ledande implementation är TnT (Brants, 2000).

Modellen där bygger i korthet på att ett enskilt ords tagg anses prediceras av de två föregående taggarna där de två taggarna, tillsammans, utgör det "dolda" tillståndet, tillsammans med en viktning för kända taggar för det specifika ordet. Baserat på detta görs en vandring genom en hel mening, där sedan den mest sannolika taggföljden som leder fram till den mest sannolika sluttaggen väljs som resultat. Trots att de enskilda sannolikheterna alltså bara "tittar bakåt" i trigram, med det egna ordet inräknat, påverkas vilken tagg som väljs för ett ord även av de följande orden. Detta får av naturliga skäl störst betydelse för inledande ord.

Ytterligare en sak som utmärker TnT är dess mycket höga prestanda och dess synnerligen höga tillförlitlighet (80 – 90 %) vad gäller okända ord, vilket är ledande för svenska utan externt ordformslexikon (Megyesi, 2001). Detta uppnås främst genom ett suffixträd som används för att tilldela sannolikheter för olika taggar motsvarande den information man får genom de förekommande taggarna för ett ord som faktiskt återfinns i korpusen.

Syftet med denna uppsats är att undersöka om en variation av TnTs Markov-modell kan ge bättre resultat, till priset av mer omfattande beräkningar. Bland de viktigare inslagen återfinns att föregående och efterföljande ord (inte bara taggar) ingår explicit i modellen, samt att viktningarna bestäms genom en Monte Carlo-baserad gradientsökningsalgoritm, i stället för genom den deterministiska algoritmen som används i TnT.

2 CNtag

Bifogat till uppsatsen finns ett program, CNtag¹, som använts för att utpröva olika modellvarianter. Programmet är skrivet i C++, då prestanda varit centrala. Systemet använder Boost-bibliotekets (Boost C++ Libraries, 2005) minnesallokerare `pool` i en något modifierad version. Denna är betydligt minnessnålare än en vanlig heap när många små objekt allokeras, vilket sker när STL utnyttjas flitigt, som i exempelimplementationen. Programmet skall vara kompatibelt med den ursprungliga versionen, men då denna dubblar minnesförbrukningen varje gång minnet är fullt kan det vara en klar nackdel på vissa system.

CNtag är skrivet för Windows-miljö, men använder på det stora hela inga plattformsspecifika funktioner, utom vissa funktioner för tidtagning och processhantering som främst utnyttjats under utvecklingsarbetet.

¹ Utläses Complex N-gram Tagger, men även Carl Nettelblad Tagger.

2.1 Grundläggande arkitektur

TnT tittar på unigram (taggfrekvenser oberoende av andra faktorer), bigram (aktuellt ords tagg utifrån föregående tagg) och trigram (aktuellt ords tagg utifrån de två föregående ordens taggar). Det finns flera metoder för hur dessa skall viktas sinsemellan, samt hur det skall hanteras om det inte finns några data om exempelvis ett visst trigrams frekvenser. Standardläget bygger på s.k. deleted interpolation, beskriven i (Brants, 2000), där man utifrån de trigram som faktiskt finns analyserar hur viktningen bör se ut för att ge samma resultat även om dessa trigram skulle plockas bort. Detta antas även beskriva hur de trigram som saknas i korpusen, eller har få (antaget icke-representativa) förekomster, uppför sig.

CNtag behandlar många fler möjliga ”mönster”. Där TnT behandlar unigram, bigram och trigram kan CNtag behandla godtyckliga kombinationer av de två föregående ordens tagg, graford, aktuellt graford, samt efterföljande ords graford och uppskattade tagg. Totalt har vi då 4 möjligheter för varje ord – ignoreras, bara tagg, bara graford, graford + tagg – för totalt fyra ord – -2, -1, +0, +1 relativt fokusordet. Inalles blir detta 256 olika möjliga mönster, men i den praktiska implementationen har de mest komplexa begränsats och möjligheten till data för graford + tagg strukits för att spara minne, eftersom data där i princip är redundanta med andra mönster.

2.2 Viktning

För vart och ett av dessa mönster finns en uppsättning vikter. TnT använder gemensamma vikter för alla trigram (o.s.v), men CNtag observerar att det finns två egenskaper som kan påverka hur vederhäftiga korpusdata om ett mönsters möjliga taggar är:

1. Hur dominerande ett visst taggval är. Om ett mönster med mycket hög säkerhet ger en viss tagg, finns det skäl att ge en högre vikt. Motsatt gäller att det, om många taggar är möjliga med ungefär samma sannolikhet, finns skäl att inte lägga så stor vikt vid det mönstret.
2. Hur många förekomster det finns av ett visst taggval med det aktuella mönstret. Ett mönster som ger en viss tagg med mycket hög säkerhet och förekommer hundratals gånger förtjänar (förmodligen) en betydligt högre vikt än om säkerheten förvisso är 100 %, men detta bara baseras på en enda förekomst i korpusens material.

Detta kan även uttryckas som att både relativa och absoluta frekvenser är dimensioner som avgör vilken vikt som skall användas. Vikterna ordnas i ett antal diskreta steg, där gränserna mellan olika steg avgörs i källkoden.

Specifikt finns det $4 * 4$ vikter för varje mönster, vilka optimeras oberoende av varandra. Vidare bidrar varje mönster till taggsannolikheterna i två olika ”serier”. Den första serien har ett startvärde 0, den andra ett startvärde 1. Serierna multipliceras sedan elementvis för att ge en slutlig sannolikhet. På detta vis kan sådana faktorer som ett enskilt ords taggfördelning även påverka taggvalet utifrån mönster som tar hänsyn till andra delar av området kring fokusordet. Ett specialfall av detta är den hänsyn TnT tar till grafordets taggfördelning. En skillnad är att TnT utesluter taggar som inte är kända för ordet, medan grundvikten ”1” i den andra serien i CNtag tillåter möjligheter som aldrig har förekommit.

Slutligen används två olika viktmatriser, en för kända ord och en för okända. Detta bygger på antagandet att okända ord inte är godtyckliga ord som endast händelsevis är obekanta sedan tidigare, utan att de kan ha egenskaper som bättre återspeglas med en separat matris. I grunden finns naturligtvis öppna och slutna ordklasser, men slutfaktorer för viktningen ingår inte i CNtag, även om det vore tänkbart. Det kan även betonas att de båda matriserna, om orden faktiskt inte uppför sig olika, bör konvergera mot identiska slutresultat.

2.3 Suffixhantering

CNtag har en modell för suffixhantering som i mycket liknar TnT. Alla ingående ord bidrar till suffixinformationen, men varje koppling mellan mönster och tagg räknas bara en gång. Detta bygger på tanken att vi här handskas med ovanliga ord. Att r som slutbokstav ofta innebär ord som ”för”, ”eller”, ”efter” är mindre intressant än att väldigt många, *olika*, verb har presensböjningar som slutar med r och att många substantiv har pluralformer som slutar med r.

I princip lagras samma mönster som för den vanliga informationen, men med suffix av olika längd i stället för information om graford. Tekniska begränsningar sätter en gräns vid 4 tecken, men det finns praktiska skäl att göra viktningsmatrisen m.m. mindre och använda en kortare längd. De nödvändiga förändringarna för att hantera en längd motsvarande TnTs 10 vore inte alltför omfattande och av just denna anledning har det inte heller setts som angeläget att införa detta stöd i denna första version.

2.4 Värderingsfunktionen

Halva CNtag består av hur en enskild text taggas. Det är beskrivet ovan. Implementationen av algoritmen är dock specialiserad på att kunna hantera förändringar av enskilda värden i viktningsmatrisen mycket fortare än en ursprunglig taggning. En annan central aspekt i en optimerande algoritm är värderingsfunktionen. Vid en optimering måste det finnas en metod som kan bedöma om en förändring var ett framsteg eller inte. En enkel metod är helt enkelt tillförlitligheten (accuracy) för viktningsmatrisen på en aktuell provtext. Mer om provtext och liknande följer i avsnittet om själva optimeringsalgoritmen.

En nackdel med tillförlitlighet som kriterium är att det bara är diskreta, större förändringar som ger positiva följder för detta värde. Samtidigt är det ganska tydligt att en höjning av sannolikheten för den korrekta taggen bör leda fram mot bättre resultat, även om den enskilda höjningen inte leder fram till att något enskilt ord taggas rätt i stället för fel. Samtidigt förblir tillförlitligheten det slutgiltigt mest intressanta måttet för en taggare som alltid ger en distinkt tagg för varje ord.

En kompromiss är en summering av andelen ”korrekt sannolikhet” respektive eventuellt en bestraffning av något slag för ”felaktig sannolikhet” (felvalda taggar) och tillförlitligheten. En viktning kan ske av tillförlitligheten så att denna har företräde, även om det sänker andelen korrekt sannolikhet i textmaterialet. Värderingsfunktionen i CNtag är av detta slag, med ett tillägg i att orden i texten läggs i två separata delmängder. För att en ändring skall vara bra måste värdet öka för båda dessa delmängder.

Syftet med denna uppdelning är att försöka undvika de problem med ”överträning” som kan uppstå i algoritmer av denna typ. Om en förändring är positiv för båda halvorna i deltexten antas sannolikheten att den även är positiv för texter från samma genre som provtexten vara större. I annat fall skulle risken för att en specifik ändring anpassade sig efter mer enskilda egenheter vara ännu större.

2.5 Optimeringen

Viktningsmatrisen initieras till värdet 1 för serie 1 och ett betydligt lägre värde för serie 2, för att i grunden ge linjära sannolikheter utifrån rådata. Efter detta går systemet in i en i teorin oändlig slinga där olika värden prövas. Här uppträder ett problem – värderingsfunktionen är inte med nödvändighet kontinuerlig och monotont växande. En förändring kan, som konstateras ovan, försämra sannolikhetsandelen, men ändå vara fördelaktig genom ökad tillförlitlighet i det faktiska taggningsvalet.

Om man bara använde små diskreta steg skulle metoden vara en normal gradientsökning, så som den måste implementeras om man inte direkt kan derivera för att hitta gradienten, utan är tvungen att pröva olika möjligheter. Med avsikten att delvis undvika problemet med lokala maxima, som det ovan beskrivna i praktiken utgör, prövas successivt växande värden. Det aktuella värdet ömsom multipliceras, ömsom divideras, med värden upp till omkring 100 000 av det aktuella. Det specifika värdet slumpas i varje steg, men med en fördelning som gör att fler värden med små förändringar prövas. En homogen sannolikhetsfördelning skulle, om ändå värden i samma storleksordning någon gång skulle prövas, bara mycket långsamt göra de små, lokala, justeringar som också krävs.

Några problem med denna metod är att den är långsam och att valet av optimeringsalgoritm, liksom valet av värderingsfunktion, påverkar i hur hög grad träningen blir verklig – en generell förbättring av taggningsförmågan – och i hur hög grad det enbart blir en överträning – en viktning som råkar passa aktuella testdata.

2.6 Valet av provtext

TnT använder bara en tränings-text, den annoterade korpusen. CNtag behöver två typer av material – tränings-texten och provtexten. Den senare bör, liksom tränings-texten, vara snarlikt den text som skall taggas. Den bör däremot inte vara en delmängd av själva tränings-texten – en viktig aspekt av viktningen är att hantera ofullständig information, både okända ord och mönster. Om man bara hade redan kända trigram i data skulle anledningen att vikta upp mindre täckande mönster vara mycket begränsad.

CNtag har inga egna funktioner för att ta fram provtexter, men ett tillhörande verktyg CNskipper kan göra slumpmässiga uppdelningar av en total tränings-text i en mindre provtext och en tränings-text för enbart det aktuella försöket. Det finns skäl att hålla provtexten liten, inte enbart av prestandaskäl. Viktningarna utformas för en specifik tränings-korpusstorlek och om diskrepansen mot den verkliga korpus som sedan används blir alltför stor inverkar detta i sig menligt på programmets prestanda.

2.7 Markov-modeller med Viterbis algoritm

TnT har inget behov av att kontinuerligt tagga om någon provtext och en fullständig CNtag-tagging innebär att långt fler mönster måste beaktas. CNtag lagrar mellan varje ny provad förändring i princip alla beräknade data och kan därför snabbt och effektivt beräkna resultaten av en ändring, med åtminstone flera tusen taggade tokens varje sekund.

Problemet är här att Viterbis algoritm, som är en typisk tvådimensionell backtracking med dynamisk programmering, blir alltför komplex och minnesförbrukande med CNtags upplägg. I det fullständiga utförandet hålls aggregerad sannolikhet fram till det aktuella ordet för varje tillstånd, som i detta fall är de två föregående taggarna. Från denna lista för ett token kan sedan fullständiga sannolikheter för nästa token beräknas. När man nått slutet har man sannolikheter för taggen i slutposition, där allt passerat har haft inverkan på resultatet. Från detta går man bakåt och väljer vid varje steg den mest sannolika taggen.

Redan i TnT har man insett att detta system kan påverka prestanda negativt. Man kan därför ange en gräns för hur osannolik en ”vandring” eller ”stråle” med taggval skall vara för att uteslutas. Normalt går gränsen vid strålar (representerade av deras slutpunkter vid det token som behandlas för tillfället) med 1/1000 av sannolikheten hos den maximala strålen.

En sådan lösning är inte praktiskt möjlig för CNtag, som måste lagra alla mellanliggande beräknade sannolikheter. I stället kan man i källkoden med hjälp av en konstant förändra hur många av de högst poängsatta taggarna från närmast föregående ord som tas med i bedömningen för nästa token. Rimliga värden på denna konstant är 2 - 3. Då systemet tittar på kontext 2 ord bakåt innebär ett värde 3 på konstanten att totalt 9 vägar bakåt analyseras. Det faktum att CNtag även tittar ett ord framåt bidrar till att dessa 9 vägar väljs med större omsorg.

Minnesanvändningen för den förlagrade informationen som gör taggningsprocessen snabbare vid träning växer också kvadratisk, medan tidsanvändningen växer ännu snabbare. Anledningen till det sistnämnda är att resultatet, om något av dessa 9 förändras i något token, måste räknas om i högre grad. Med tillräckligt många värden blir risken att någon måste räknas om allt högre.

Efter beräkningen går man bakåt på ett sätt som motsvarar Viterbis algoritm. Det skall kanske betonas att både TnT och CNtag egentligen bara skulle behöva utföra Viterbis algoritm, respektive CNtags grovt förenklade version av denna, på en mening i taget – då kontexten inte går mellan meningar. Den höga minnesanvändningen i CNtags förlagring är inget krav när man inte tränar. Godtyckligt långa texter kan taggas, dock inte godtyckligt långa meningar. Då det egentligen bara är ordkontexter som behöver lagras och inte ens då hela poängmatriserna är även detta minneskrav mycket modest. Den potentiellt avskräckande minnesanvändningen i systemet är alltså något som mildras när man inte tränar utan vill ”använda” systemet.

3 Utvärdering

All utvärdering har skett med utsnitt ur Stockholm Umeå Corpus (SUC) i en version med PAROLE-taggar. Filformatet har varit baserat på det filformat TnT förväntar sig. Dock finns inte stöd för de kommentarrader med inledande procenttecken (%) som TnT hanterar. Vidare antas tomrader

markera meningsgränser. Till skillnad från TnT finns inga funktioner för att spontant betrakta relevanta skiljetecken som meningsskiljande. Beroende på typen av material torde en sådan segmentering vara enkel att göra med regler. Korta försök antyder att CNtag inte är direkt beroende av dessa markörer, utan kan ta hela texten som ett "flöde" om det är vad som erbjuds.

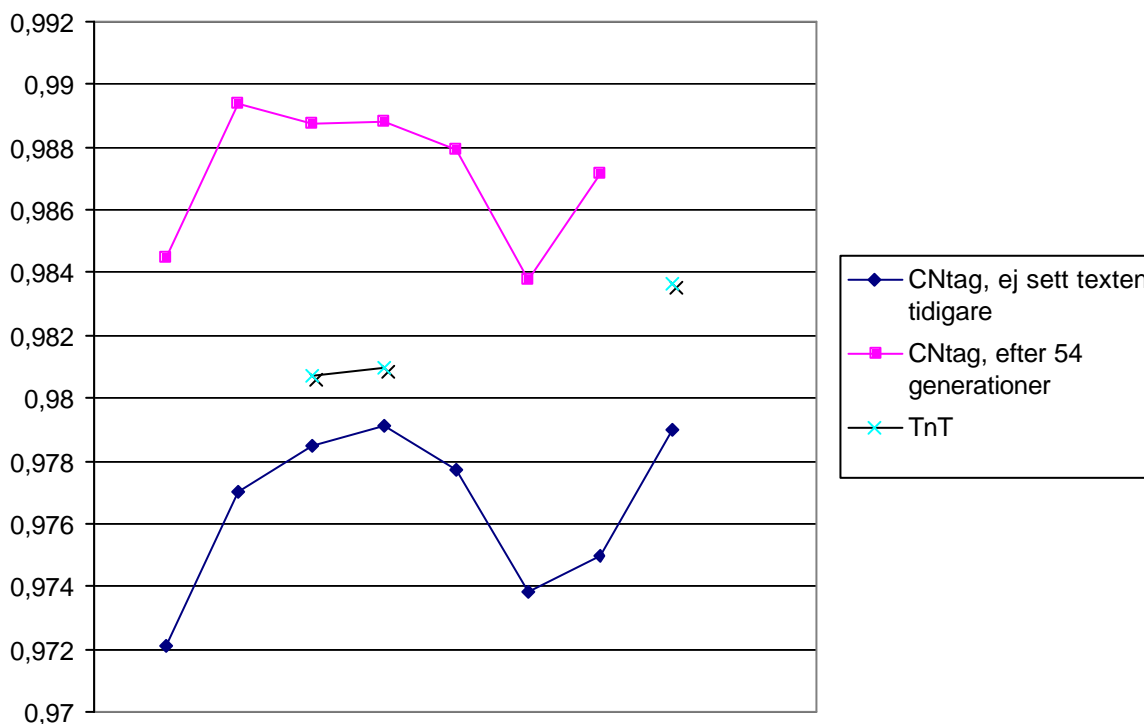
Som nämndes i inledningen kan det finnas tillämpningar där en lista med tänkbara taggar är lika relevant som en entydig tilldelning. En grammatikkontroll bör exempelvis kanske inte markera att "ett rå" skall vara "ett rått", även om taggaren kommer till slutsatsen att rå mest sannolikt är ett adjektiv, men med en hög sannolikhet för substantivtolkningen (skogs)rå.

En träning utfördes därför med CNtag i några "varv" med CNskipper, på ett SUC-utsnitt om totalt 125 211 tokens, inklusive separata meningsgränsmarkörer. I varje varv om 54 träningsgenerationer med CNtag användes omkring 1 / 30 av detta material som provtext, respektive 7 / 30 som träningstext. Vid varje enskilt "varv" var dessa disjunkta, medan träningsexterna totalt sett naturligtvis måste ha överlappat något och även provtexterna kan ha gjort det. Då meningsurvalet var slumpmässigt gäller även, som Brants påpekar, att träningsexterna kan innehålla meningar som ligger intill provtextens meningar och att detta påverkar främst *andelen* okända ord i en riktning som ger realistiskt goda resultat.

Här fokuseras emellertid på tillförlitligheten att korrekt tagg finns bland de två högsta förslagen för varje ord. Således användes 4 möjliga vägar (2 taggar för varje token). Försöket gjordes med en tidigare version av CNtag, där en bugg innebar att i praktiken endast 2 av de 4 möjliga taggarna togs i beaktande. Suffixlängden låstes till 1, både i TnT och CNtag, för att inte slösa tid. Naturligtvis påverkar tillförlitligheten för okända ord även den för kända, men effekterna är relativt begränsade.

Diagrammet nedan innehåller kurvor för CNtag när den först mötte texten i ett visst varv, samt efter 54 generationer. Vi kommer in då beteendet stabiliserat sig något, d.v.s. efter ett första initialt varv då den kraftigaste optimeringen sker. Vissa datapunkter saknas då processen pågick under lång tid och vissa loggar inte fungerade som förväntat. TnT-siffrorna erhöles genom växlarna -m -z0/2 -a1 -Z1000 (Brants, 1999).

3.1.1 CNtag, tvetydig taggning, kända ord



Vi kan här se att CNtag med överträning markant slår TnT för kända ord. Detta gäller inte för okända ord med suffixlängden 1, vilket även inverkar menligt på de kända. Vid högre suffixlängder

gäller inte detta. Vi ser även att de tre olika kurvorna på det stora hela följer samma tendens för vilka provtexter som är lätttagade och inte.

Vid upprepade försök med TnT kontra CNtag med varierande provtexter i en tidigare version gav CNtag konsekvent 0 – 2 procentenheters lägre tillförlitlighet än TnT för kända ord vid första möte med ny provtext. Denna version var markant enklare, då den enbart valde den mest sannolika taggen med enbart lokal kontext och förde det valet framåt. Dessa nedslående resultat bidrog till att den tidigare endast övervägda flervägsalternativet implementerades. Skillnaden om korpusstorleken tredubblades var negligerbar. Det kan ju även konstateras att TnT i sig inte gav signifikant bättre resultat vid test på så små material som CNtags provtexter innebär.

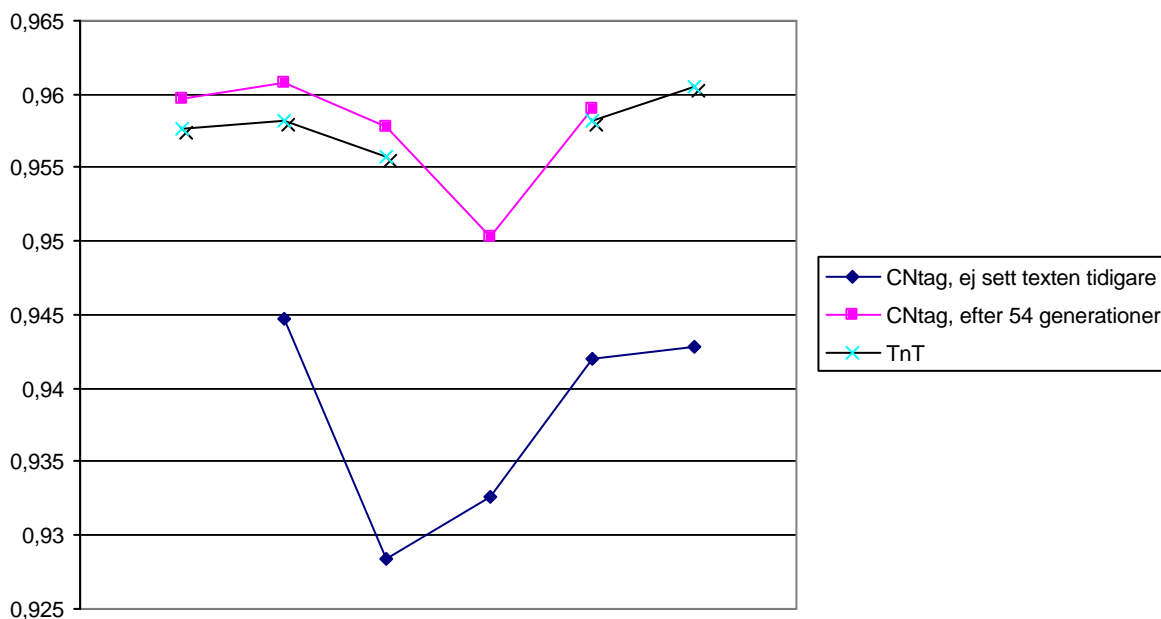
En annan frågeställning av intresse är beteendet för suffixhanteringen vid ökande suffixlängd. CNtag tilläts här överträna på en och samma provtext med respektive suffixlängd. Detta gjordes också med en äldre version, snarlik den som redovisades kvalitativt här närmast innan. Texten omfattade omkring 800 meningar ur början av SUC, med samma träningskorpus som ovan, fast i sin helhet. Tränings- och provtexterna var helt disjunkta.

	Kända	Okända	Totalt
TnT, suffix 0	6 400	510	6 910
CNtag, suffix 1	6 472	878	7 350
TnT, suffix 1	6 419	863	7 282
CNtag, suffix 2	6 476	985	7 461
CNtag, suffix 3	6 473	1 036	7 509
TnT, suffix 10	6 430	1 075	7 505

Här rör det sig alltså återigen om potentiell överträning på en och samma text, varför överprestationerna relativt TnT med samma suffixlängd är fullt väntade. Mätningarna togs efter ett generationsantal då förbättringen börjat plana ut, men det bör nämnas att de omkring 50 generationer som använts både här och i övriga redovisningar inte är något teoretiskt maximum för optimeringsalgoritmen. Tvärtom förekom vid långförsök resultat från CNtag med över 6500 korrekt taggade kända ord. De matriser som erhöles där gav, när de provades med andra texter, inte intrycket att denna förbättring var mycket mer än ytterligare överträning.

Nedan redovisas slutligen ett diagram över tillförlitlighet för kända ord. Urvalet påminner om det tidigare, men med 1/26 som provtext och 22/26 tränings-text. Antalet följda vägar begränsades till 4 för snabbare resultat. Suffixlängden var 1 för båda.

3.1.2 CNtag, entydig taggning, kända ord



Vi kan konstatera att resultaten påminner om de kvalitativt beskrivna tidigare, dock något sämre. En anledning kan vara att de korta suffixen, i kombination med de få följda vägarna, ger upphov till fler fel. De andra resultaten gjordes med längre suffix, men bara en följd väg. Det tycks alltså som att den "globala" kontext Viterbis algoritmen, respektive CNtags enklare variant av flera följda vägar, innebär, inte ger bättre resultat. Vi kan också se att träning på en enda provtext respektive efter flera inte tycks påverka tillförlitligheten på följande texter markant.

4 Slutsats

CNtag innehåller många olika idéer, där den centrala får anses vara viktning av olika mönster, samt viktning efter deras säkerhet. Samtidigt kan bara denna algoritmen utvärderas genom den optimeringsalgoritmen och värderingsfunktion som används.

Begränsning av antalet mönster, endast en följd väg, respektive diverse under utvecklandet successivt introducerade och avlägsnade fel i programmet ger ändå upphov till förvånansvärt likartade resultat, i princip aldrig över TnT på okänd text, men ej heller långt under, i storleksordningen någon procentenhet.

Det enda säkra resultatet är att de bästa siffrorna med en viss provtext blir sämre med metoden för två uppdelade poäng. Eventuellt ges en mycket liten sänkning även för tillförlitligheten i efterföljande provtexter, men det verkar snarast röra sig om en minskad överträning.

Det faktum att åtskilliga rättade fel och följande av flera vägar praktiskt taget inte påverkar slutresultatet antyder att de verkliga resultaten snarast är avhängiga av egenheter i vad optimeringsalgoritmen kan och inte kan hitta, i stället för hur bra, allmänna, resultat som viktningmatrismetoden kan uttrycka. En fråga är exempelvis om en tillräckligt stor provtext skulle göra att en större del av den artefaktiska överträningen faktiskt var verklig träning som uttryckte språkliga samband i den aktuella genren. Man kan också tänka sig att byta provtext mycket mer frekvent, kanske mellan varje träningsgeneration. På samma tid skulle man då, eventuellt, nå högre verklig tillförlitlighet, och mindre överträning.

Det kan också betonas att en delmängd av CNtag är vad TnT gör med ett begränsat antal vägar ("beams"), så en bra optimeringsmetod borde kunna uppnå motsvarande siffror. Att så ej sker visar på den nuvarande optimeringens brister.

Samtidigt kan man konstatera att CNtag vinner mer än TnT på längre suffix, men att flera experiment här utfördes endast med suffixlängd 1, på grund av tidsåtgången. Metoden tycks alltså inte ge några entydiga förbättringar, utom i det specialfallet att den, med en enda sökväg, är bättre än en TnT med endast en sökväg. Detta skulle kunna vara av värde i tillämpningar där man kan veta följande ord, men vill ha en taggning innan hela meningen är avslutad.

Källförteckning

T. Brants, 1999. TnT – A Statistical Part-of-Speech Tagger (manual). Elektronisk. Tillgänglig på <<http://www-old.coli.uni-saarland.de/~thorsten/publications/Brants-TR-TnT.pdf>> (2005-04-06).

T. Brants, 2000. TnT – A Statistical Part-of-Speech Tagger. *Proceedings of the 6th Applied Natural Language Processing Conference*, ss. 225-231. Seattle, Washington, USA.

Boost C++ Libraries, 2005. Elektronisk. <<http://www.boost.org>> (2005-04-06).

B. Megyesi, 2001. Comparing Data-Driven Learning Algorithms for PoS tagging of Swedish, ss. 151-158. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Bilaga

Källkoden till CNtag (CNtag_main.cpp) bifogas. Den är skriven för att vara snabb och har utvecklats i flera omgångar, men är i gengäld rikligt kommenterad för att ändå vara begriplig.