Named entity recognition using statistical methods

Stefan Ekenberg
Department of Computer Science
Lund University
d01se@efd.lth.se

Abstract

This report is the result of the project part of a course in computer linguistics. The assignment was to develop a java program that tags proper nouns using statistical methods. To do this, a Support Vector Machine was used and the main part of the project was to find out what parameters to send into the SVM to get a good result.

To evaluate the system the precision and recall values of a corpus were calculated.

1 Introduktion

Syftet med denna rapport är att beskriva hur statiska metoder kan användas i ett system för att tagga egennamn i en text. En träningstext används för att lära systemet vilka taggar som skall användas och i vilken kontext som dessa oftast befinner sig i. Informationen kan sedan användas för att tagga en liknande text.

Till skillnad från tidigare arbete som ”Name Extraction in Car Accident Reports for Swedish”[1] används inga gazetteers[2] eller regler som till exempel reguljära uttryck. Detta ger främst de två fördelarna att systemet blir generellt och kan på så sätt användas i många olika sammanhang samt att den implementerade koden blir mycket enklare att felsöka och få översikt över. Koden med reguljära uttryck blev mycket svårhanterlig och det var mycket svårt att hitta fel samt att lägga till nya uttryck. Dock har även detta nya system utvecklats för att fungera så bra som möjligt på texter om trafikolyckor, precis som tidigare nämnda arbete. Detta har påverkat en del val av parametrar som används för att känna igen kontexten för en viss tagg.

Parametrarna som plockas ut från det ord som skall taggas och dess kontext matas in i LIBSVM som är ett programpaket för att känna igen mönster. Uppgiften i projektet var att hitta de parametrar som skulle matas in i LIBSVM för att få bästa möjliga resultat.

Först kommer denna rapport nämna möjliga användningsområden för systemet och därefter beskrivs LIBSVM kortfattat. Sedan kommer de olika parametrarna som skickas in i detta programpaket behandlas noggrant och till sist görs analys av hur storleken på systemets träningskorpus påverkar resultatet samt en evaluering av korrektheten vid taggning som systemet producerar.

2 Bakgrund

System för att tagga egennamn kan till exempel användas till program som behandlar texter och behöver information om de olika orden i texten för att sedan på något sätt kunna tyda vad meningen betyder. CarSim[3], som är ett text-till-scenomvandlingsprogram för trafikolyckor, är ett exempel på program som har användning av att

[1] Ett projekt som tidigare utförts på LTH i kursen ”Språkbehandling och Datalingvistik”, precis som detta projekt. Se referenser.

[2] Gazetteer – lista med ord och motsvarande tagg som systemet har lagrad.

[3] Pierre Nugues, 2004. Se referenser.

veta vilka ord som är egennamn. Då kan det ta reda på vem som är inblandad i olyckan, var den inträffade och så vidare.

3 LIBSVM

LIBSVM bygger på Support Vector Machines som är en statistisk metod för att känna igen mönster. För att klassificera försöker SVM hitta en hyperyta i rymden av möjliga indata. Denna hyperyta försöker dela de positiva exemplen från de negativa. Delningen görs på ett sådant sätt att hyperytan får så långt avstånd som möjligt till de närmsta av de positiva och negativa exemplen. På så sätt kan en klassificering göras korrekt för testdata som är nära men inte identisk till träningsdatan. För mer information om SVM se ”Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001”, ”A Tutorial on Support Vector Machines for Pattern Recognition” och en hemsida från Microsoft Research CCSP Group på adressen http://research.microsoft.com/~jplatt/svm.html.

4 Parametrar

För att få idéer på vilka parametrar som kan ge bra resultat har bland annat artiklarna ”Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”, ”Named Entity Recognition through Classifier Combination” och ”Named Entity Recognition with a Maximum Entropy Approach” studerats. Dessa användes som en utgångspunkt för vilka parametrar som med stor sannolikhet borde ingå. Därefter har felaktiga taggningar i resultatet studerats för att komma på nya parametrar som skulle kunna användas. Dessa har sedan testats för att se om de gör en positiv eller negativ inverkan på resultatet för att kunna avgöra om de skall användas respektive förkastas. På så sätt har en iterativ metod använts för att komma fram till slutresultatet.

I Tabell 1 finns en sammanställning av de parametrar som gjorde en positiv inverkan på resultatet och därför ingår i systemet.

Tabell 1. De olika parametrarna som används och deras relativa inverkan på resultatet. Vid framtagningen av värdena i tabellen användes en träningskorpus på 9502 ord och en testkorpus på 3805 ord. Värden större än 1 betyder att då parametern används så ökar värdet. Om en parameter är bra så skall den alltså ha värden som är större än 1 i kolumnerna ”Korrekta taggningar”, ”Precision” och ”Recall” medan värdet för ”Feltaggningar” skall vara mindre än 1 (förklaring till precision och recall finns i stycke 6).

Parametertyp	Korrekta taggningar:	Feltaggningar:	Precision:	Recall:
Ordets suffix	1,12	0,82	1,10	1,12
- Längden 4	1,02	0,93	1,03	1,02
- Längden 3	1,01	1,01	0,99	1,01
- Längden 2	1,03	0,95	1,02	1,03
- Längden 1	1,01	0,98	1,01	1,01
Föregående och efterföljande ord	1,07	0,84	1,11	1,07
- Ordet innan föregående	1,00	0,95	1,02	1,00
- Föregående ord	1,02	0,95	1,02	1,02
- Efterföljande ord	1,04	0,93	1,03	1,04
Inledande stor bokstav	1,06	0,89	1,02	1,06
Ordklassinformation	1,04	0,93	1,03	1,04
Frasinformation	1,03	0,95	1,02	1,03
Efterföljande är ett nummer	1,03	0,95	1,01	1,03
Efterföljande ord har stor bokstav	1,04	0,91	1,05	1,04

4.1 Ordets suffix

Suffix av det ord som skall taggas gav positivt resultat upp till längden fyra. Hela ordet används även som parameter men suffix kan få fram generella mönster på orden för de olika taggarna som inte enbart själva ordet kan. En möjlig anledning till att suffix av längden fem inte gav något förbättrat resultat är att inte träningsdatan var tillräckligt omfattande för att LIBSVM skulle kunna känna igen mönster av längden fem.

Suffix av längden ett (sista bokstaven) skulle man kunna tro var för generell för att användas. Dock var så inte fallet som visas i Tabell 1. Detta kan förklaras med samband som till exempel att gatunamn ofta slutar på bokstaven n vilket är sista bokstaven i ”gatan” och ”vägen”.

Suffix av längden två hade en större inverkan på resultatet men någon direkt förklaring till detta är svårt att ge. Dock är ju denna mindre generell än endast en bokstav vilket gör att mönster är lättare att upptäcka.

Suffix av längden tre är bra för att upptäcka mönster i efternamn vilka ofta slutar på ”son”, som till exempel ”Andersson” och ”Johnson”.

Suffix av längden fyra gav även ett positivt resultat. En möjlig förklaring till det positiva resultatet är till exempel samband som att städer ofta slutar på ”borg” som i ”Helsingborg” och ”Ludviksborg”.

Suffix av längden fem gav som tidigare inget förbättrat resultat. Dock skall det observeras att så är fallet när även alla de andra suffixen används. Om alla de andra ej används medan suffix av längden fem används så ger denna en positiv inverkan på resultatet jämfört med om inget suffix alls används.

4.2 Föregående och efterföljande ord

För att kunna utnyttja ett ords kontext används föregående och efterföljande ord som parametrar. Dessa medför att programmet tar hänsyn till i vilket sammanhang som ordet befinner sig i.

Testning har visat att systemet fungerar bäst då de två föregående orden var för sig samt det efterföljande ordet används som parametrar. Träningsdatans begränsade storlek kan ses som en anledning till att systemet inte fungerar bättre med ännu fler föregående och efterföljande ord som parametrar. Om det skall vara möjligt att se mönster på ett längre avstånd från ordet som skall taggas så måste systemet tränas mer eftersom orden får ett mindre och mindre samband med ordet som skall taggas ju längre avståndet är.

Ett exempel på en mening där kontexten har stor betydelse är ”Mannen och kvinnan åkte till Malmö.” som innehåller typiska ord för en viss sorts tagg innan ordet ”Malmö”. Om orden ”åkte” och ”till” befinner sig innan ett ord är det stor sannolikhet att ordet skall taggas med en tagg som LOCATION eller CITY, beroende på vilka taggar som systemet har tränats med. Ett exempel på en ytterligare mening är ”Jag vill ha glass, sade Johan” där ordet ”sade” innan ”Johan” kan hjälpa systemet att tagga rätt eftersom ord efter ”sade” med stor sannolikhet skall ha en tagg av typen PERSON.

Ett försök med bigram som parameter, det vill säga de två föregående orden tillsammans, gjordes även med ett misslyckat resultat som följd. Att skicka in orden var för sig som parametrar gav mycket bättre resultat. Återigen borde detta bero på träningsdatans storlek eftersom det finns många fler variationer av bigram än av orden var för sig.

Efterföljande ord kan även ha stor betydelse för att systemet skall ha möjlighet att tagga rätt. I meningar som ”De med svårast skador flyttades till Falu lasarett.” kan system har svårt att tagga ”Falu lasarett” som LOCATION om det inte vet om att ordet ”lasarett” kommer efter ”Falu”. Tabell 1 visar även att det efterföljande ordet inverkar mer på resultatet än de två föregående var för sig.

4.3 Inledande stor bokstav

En typisk egenskap för ett egennamn är att det inleds med stor bokstav. Därför används en parameter som kan anta värdena ”true” och ”false” som anger om ordet inleds med stor bokstav eller inte. Tabell 1 visar dock att parametern inte har så stor betydelse som man skulle kunna tro. Detta beror till största del på att den inte säger så mycket vilken tagg som skall sättas utan mer att någon tagg överhuvudtaget skall sättas. Att ett ord inleds med stor bokstav avslöjar inte om det är en stad eller ett personnamn.

Något som också måste observeras är att alla egennamn inte inleds med stor bokstav. Till ex-

empel ordgruppen ”riksväg 13” som möjligtvis skall taggas som ROAD har inte stor bokstav i något av orden.

4.4 Ordklassinformation

Om texten som skall taggas även innehåller information om ordets ordklass hjälper detta även systemet att tagga korrekt. Ett egennamn har då oftast ordklasstaggen ”pm.nom” vilket hjälper systemet på samma sätt som inledande stor bokstav. Dock kan det förekomma fel i ordklasstaggarna eftersom dessa antagligen är satta av en automatisk ordklasstaggare som inte är hundraprocentig. Detta har nackdelen att man får in fler felkällor i resultatet men som Tabell 1 visar så har denna parameter ändå en positiv inverkan på resultatet.

Den positiva inverkan som ordklassinformationen för med sig måste sättas i relation till den extra beräkningskraft som krävs för att köra ordklasstaggaren. Resultatet visar att ordklassinformationen inte har en så pass stor inverkan att den är nödvändig och därför kan systemet klara sig utan den då den ej finns tillgänglig.

4.5 Frasinformation

Precis som med ordklassinformationen kan texten som skall taggas först gå igenom en frastyptaggare som sätter ut frastypsinformation. Eftersom egennamn oftast taggas som substantivfras kan detta hjälpa systemet att identifiera att en tagg skall sättas, dock ej vilken. Frastypinformationen medför även den att det finns en felkälla eftersom en frastyptaggare inte heller alltid taggar korrekt. Denna parameter förbättrar ändå resultatet, dock är inte förbättringen inte lika tydlig som med ordklassinformationen.

Precis som med ordklassinformationen så visar resultatet i Tabell 1 att den extra beräkningskraft som krävs för att sätta ut frasinformationen inte alltid är motiverad. Frasinformation är alltså inte heller en nödvändighet men har en positiv inverkan på resultatet.

4.6 Efterföljande är ett nummer

Systemet kontrollerar om den grupp av tecken som befinner sig efter ordet som skall taggas är ett nummer. Denna information används som en parameter som kan anta värdena ”sant” eller ”falskt”. Taggen används för att kunna märka upp taggar som ROAD där dessa ofta skrivs som till exempel ”riksväg 13”. För att systemet skall ha möjlighet att tagga ”riksväg” korrekt behövs informationen att ”13” är ett nummer. Detta är en parameter som är lite specialanpassad till den korpus med trafikolyckstexter som användes för att träna och testa systemet och har ingen märkbar betydelse på andra typer av korpus.

4.7 Efterföljande ord har stor bokstav

Denna parameter ökar systemets förmåga att tagga personnamn korrekt. Till exempel ”Joakim” i ”Joakim Palmkvist” har större sannolikhet att få rätt tagg när denna parameter används. Däremot så hjälper den ju inte när till exempel endast förnamnet finns utskrivet i texten. Precis som parametern med kontrollen om efterföljande teckengrupp är ett nummer, så består även denna av antingen värdet ”sant” eller ”falskt” då efterföljande ord har inledande stor bokstav respektive ej inledande stor bokstav.

4.8 Föregående ords egennamnstagg

Innan denna parameter användes hade systemet stora problem med det gjorde konstiga taggningar som till exempel för ”Udo Theil”:

```
Udo I-PERSON
Theil I-CITY
```

Det vill säga taggningar som skulle innefatta flera ord fick istället olika taggar för de enstaka orden. Genom att även använde föregående ords tagg som parameter i systemet kunde taggningens kvalitet öka avsevärt vilken syns tydligt i Tabell 1. Det är ju mycket mindre sannolikt att två ord efter varandra har olika taggar än att de båda har samma tagg. Då denna parameter används får ”Udo Theil” sin korrekta taggning, det vill säga:

```
Udo I-PERSON
Theil I-PERSON
```

4.9 Kommentarer till val av parametrar

Tester har även gjorts med lite mer avancerade parametrar. Försök att komma till rätta med problemet att personnamn inte fick någon tagg då inte både för- och efternamn fanns utskrivet har gjorts, tyvärr med misslyckat resultat. Denna parameter använde det faktum att då ett namn nämns i en artikel så står nästan alltid hela namnet utskrivet först, det vill säga med både för- och efternamn och dessa kunde systemet tagga korrekt som PERSON. Dessa namn sparades sedan undan i en lista och för varje ord som skulle taggas så kontrollerades det mot denna lista för att se om det tidigare taggats som namn. I så fall fick parametern värdet ”sant”, annars värdet ”falskt”. Resultatet blev dock att många taggningar där både för- och efternamn fanns med nu istället blev felaktiga och systemet blev inte heller bättre på syftet med taggen – att tagga personnamn som endast bestod av för- eller efternamn.

Orsaken till detta är antagligen att parametern inte alltid är antingen ”sant” eller ”falskt” då systemet tränas eftersom första gången namnet förekommer är den ”falskt” och nästa gång är den ”sant”. På så sätt förvillar istället parametern eftersom den inte har samma värde för alla ord som skall taggas som PERSON.

En lösning på problemet hade varit att inte gå via SVM och använda informationen som en parameter utan att själv tagga ord som PERSON då de finns med i listan. Då förloras dock syftet med systemet eftersom det inte längre blir generellt och all information som finns i de andra parametrarna tas ingen hänsyn till. Det behöver ju inte vara ett personnamn bara för att det finns med i listan. Till exempel ”Berg” kan först finnas med i ”Johan Berg” och sedan inleda en mening som ”Berg är vackra att se på!”.

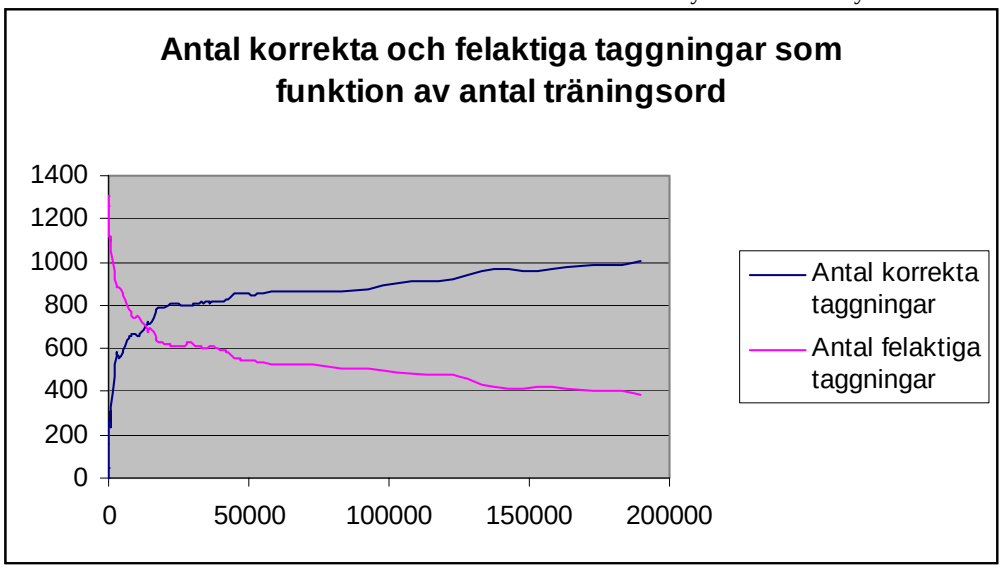
Läxan som kan läras av detta är att man inte skall anstränga sig för mycket vid val av parametrar och hitta på komplicerade samband, utan ta med generella saker som finns i ordets närhet, det vill säga den typen av parametrar som finns med i Tabell 1.

5 Träningskorpusens storlek

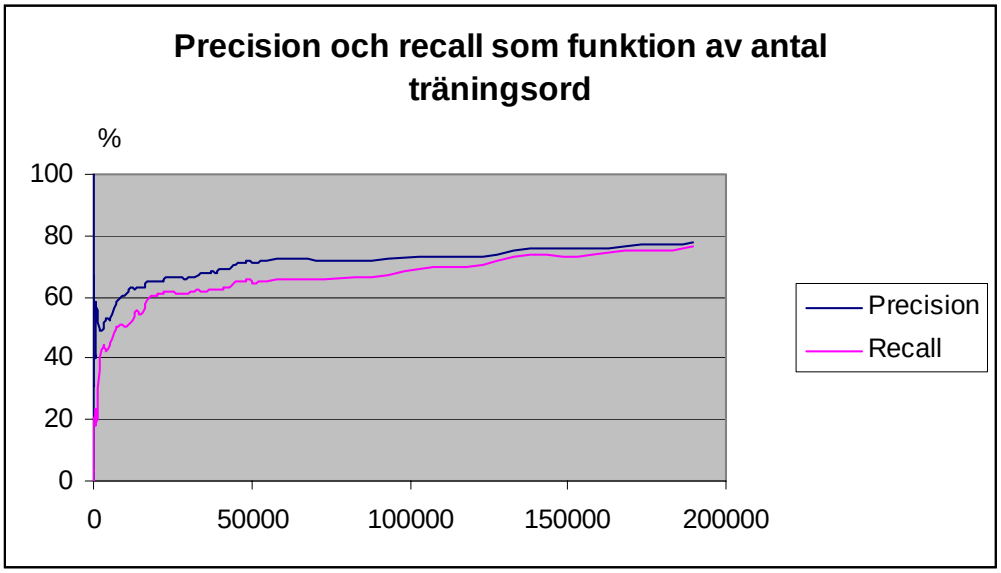
Det är inte bara parametrarna som har en stor betydelse för hur resultatet blir utan även storleken på träningskorpusen har en stor inverkan. Detta illustreras i Figur 1 och Figur 2 som visar hur antalet korrekta och felaktiga taggningar beror av antal träningsord respektive hur precision[4] och recall[5] beror av antalet träningsord.

I figurerna syns att kurvorna fortfarande vid 190000 ord inte har stabiliserat sig vilket betyder att en ännu större träningskorpus hade gett ett bättre resultat. Tyvärr fanns inte större träningskorpus tillgänglig för att vidare utforska hur stor den måste vara för att en ytterligare storleksök-

[4] Vad precision betyder beskrivs i stycke 6
[5] Vad recall betyder beskrivs i stycke 6



Figur 1. Visar hur antalet korrekta och felaktiga taggningar beror av träningsdatans storlek



Figur 2. Visar hur precision och recall beror av träningsdatans storlek

ning inte skall påverka resultatet.

6 Resultat

För att kunna evaluera systemet har en testkorpus med trafikolyckstexter, som innehåller 3805 ord, taggats manuellt vilket gav 182 taggar. Denna taggning jämfördes sedan med systemets producerade taggning. Innan dess tränades systemet med en annan trafikolyckskorpus innehållandes 9502 ord och 436 taggar. Dock var denna träningskorpus maskinellt taggad vilket gör att det inte är helt korrekt och detta påverkar systemets resultat negativt. Dessutom är det inte tillräckligt stort för att systemet skall ha en chans att lära sig de olika taggarnas egenskaper. Med en större korpus som är korrekt taggat hade alltså resultatet varit bättre.

Vid utvärderingen används följande vokabulär:

- **Answer file** – texten, maskinellt taggad av vårt program.
- **Key file** – texten, manuellt taggad. Denna text utgör definitionen på korrekt taggning.
- **Recall** – antalet korrekta taggar i Answer file dividerat med det totala antalet taggar i Key file.
- **Precision** – antalet korrekta taggar i Answer file dividerat med det totala antalet taggar i Answer file.

Systemet gav följande resultat:

```
Antal korrekta taggningar:
144
Antal feltaggningar: 55
Precision: 77.84%
Recall: 79.12%
```

Eftersom kvaliteten på den träningskorpus som användes inte var den bästa kan inte alltför stora slutsatser dras av detta resultat. Det ger dock en fingervisning på hur pass bra systemet är.

Systemet har även testats på en korpus som innehåller ekonomiska texter på engelska. Träningskorpusen innehöll ca 190000 ord och testkorpusen innehöll ca 23000 ord. Tyvärr var både tränings- och testkorpus maskinellt taggade vilket återigen gör att resultatet endas kan ses som en fingervisning. Då erhölls följande resultat:

```
Antal korrekta taggningar:
1001
Antal feltaggningar: 383
Precision: 77.96%
Recall: 76.41%
```

Resultatet visar att systemet även fungerar väl på texter skrivna i andra språk än svenska samt andra typer av texter. Som en jämförelse kan man studera resultatet från CoNLL-2003 Shared Task[6]

[6] Se referenser.

som gick ut på att tagga egennamn. Där fick det bästa systemet resultatet

Precision: 88.99%
Recall: 88.54%

på engelska texter vilket är ett betydligt bättre resultat. Dock använde detta system, precis som de flesta andra som ställde upp, gazetteers vilka ej skulle användas i detta system eftersom det skall vara så generellt som möjligt.

7 Slutsatser

Vikten av valet av bra parametrar kan inte nog påpekas när man skall utveckla ett system som använder en statistisk metod som detta system. För att ta fram parametrar som fungerar bra måste man testa sig fram. Dock kan man alltid ha en tanke i bakhuvudet som säger att generella parametrar som ligger i ordets kontext fungerar bäst. Man skall inte försöka hitta komplicerade samband.

Systemet som implementerades med hjälp av enbart statistiska metoder får tyvärr anses inte vara tillräckligt bra för kommersiell användning. Då måste precision och recall säkert upp till 97 % innan det kan anses intressant. Dit har systemet en lång väg. För att komma dit måste antagligen knep som gazetteers användas på bekostnad att systemens generallitet.

Tack

Jag vill tacka Richard Johansson för implementeringen av interface till LIBSVM, allmänna tips samt stödet under projektets gång.

8 Referenser

Lisa Persson and Magnus Danielsson. 2003. *Name Extraction in Car Accident Reports for Swedish.* Projektarbete på Lunds Tekniska Högskola i kursen "Språkbehandling och Datalingvistik".

Pierre Nugues. 2004. *Development of a Text-to-Scene Converter for Vehicle Accident Reports*, http://www.lucas.lth.se/lt/carsim.shtml

Chih-Chung Chang and Chih-Jen Lin. 2001. *LIBSVM: a library for support vector machines*. Mjukvara tillgänglig på internet-adressen http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Christopher J.C. Burges. 1998. *A Tutorial on Support Vector Machines for Pattern Recognition*. Bell Laboratories, Lucent Technologies.

Microsoft Research. *Support Vector Machines*. http://research.microsoft.com/~jplatt/svm.html.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. CNTS – Language Technology Group University of Antwerp.

Radu Florian, Abe Ittycheriah, Hongyan Jing and Tong Zhang. 2003. *Named Entity Recognition through Classifier Combination*. IBM T.J. Watson Research Center 1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA.

Hai Leong Chieu and Hwee Tou Ng. 2003. *Named Entity Recognition with a Maximum Entropy Approach.* Department of Computer Science National University of Singapore 3 Science Drive 2 Singapore 117543.