Named Entity Recognition using an HMM-based Chunk Tagger

GuoDong Zhou   Jian Su
Laboratories for Information Technology
21 Heng Mui Keng Terrace
Singapore 119613
zhougd@lit.org.sg   sujian@lit.org.sg

Abstract

This paper proposes a Hidden Markov Model (HMM) and an HMM-based chunk tagger, from which a named entity (NE) recognition (NER) system is built to recognize and classify names, times and numerical quantities. Through the HMM, our system is able to apply and integrate four types of internal and external evidences: 1) simple deterministic internal feature of the words, such as capitalization and digitalization; 2) internal semantic feature of important triggers; 3) internal gazetteer feature; 4) external macro context feature. In this way, the NER problem can be resolved effectively. Evaluation of our system on MUC-6 and MUC-7 English NE tasks achieves F-measures of 96.6% and 94.1% respectively. It shows that the performance is significantly better than reported by any other machine-learning system. Moreover, the performance is even consistently better than those based on handcrafted rules.

1   Introduction

Named Entity (NE) Recognition (NER) is to classify every word in a document into some predefined categories and "none-of-the-above". In the taxonomy of computational linguistics tasks, it falls under the domain of "information extraction", which extracts specific kinds of information from documents as opposed to the more general task of "document management" which seeks to extract all of the information found in a document.

Since entity names form the main content of a document, NER is a very important step toward more intelligent information extraction and management. The atomic elements of information extraction -- indeed, of language as a whole -- could be considered as the "who", "where" and "how much" in a sentence. NER performs what is known as surface parsing, delimiting sequences of tokens that answer these important questions. NER can also be used as the first step in a chain of processors: a next level of processing could relate two or more NEs, or perhaps even give semantics to that relationship using a verb. In this way, further processing could discover the "what" and "how" of a sentence or body of text.

While NER is relatively simple and it is fairly easy to build a system with reasonable performance, there are still a large number of ambiguous cases that make it difficult to attain human performance. There has been a considerable amount of work on NER problem, which aims to address many of these ambiguity, robustness and portability issues. During last decade, NER has drawn more and more attention from the NE tasks [Chinchor95a] [Chinchor98a] in MUCs [MUC6] [MUC7], where person names, location names, organization names, dates, times, percentages and money amounts are to be delimited in text using SGML mark-ups.

Previous approaches have typically used manually constructed finite state patterns, which attempt to match against a sequence of words in much the same way as a general regular expression matcher. Typical systems are Univ. of Sheffield's LaSIE-II [Humphreys+98], ISOQuest's NetOwl [Aone+98] [Krupha+98] and Univ. of Edinburgh's LTG [Mikheev+98] [Mikheev+99] for English NER. These systems are mainly rule-based. However, rule-based approaches lack the ability of coping with the problems of robustness and portability. Each new source of text requires significant tweaking of rules to maintain optimal performance and the maintenance costs could be quite steep.

The current trend in NER is to use the machine-learning approach, which is more

attractive in that it is trainable and adaptable and the maintenance of a machine-learning system is much cheaper than that of a rule-based one. The representative machine-learning approaches used in NER are HMM (BBN's IdentiFinder in [Miller+98] [Bikel+99] and KRDL's system [Yu+98] for Chinese NER.), Maximum Entropy (New York Univ.'s MEME in [Borthwick+98] [Borthwich99]) and Decision Tree (New York Univ.'s system in [Sekine98] and SRA's system in [Bennett+97]). Besides, a variant of Eric Brill's transformation-based rules [Brill95] has been applied to the problem [Aberdeen+95]. Among these approaches, the evaluation performance of HMM is higher than those of others. The main reason may be due to its better ability of capturing the locality of phenomena, which indicates names in text. Moreover, HMM seems more and more used in NE recognition because of the efficiency of the Viterbi algorithm [Viterbi67] used in decoding the NE-class state sequence. However, the performance of a machine-learning system is always poorer than that of a rule-based one by about 2% [Chinchor95b] [Chinchor98b]. This may be because current machine-learning approaches capture important evidence behind NER problem much less effectively than human experts who handcraft the rules, although machine-learning approaches always provide important statistical information that is not available to human experts.

As defined in [McDonald96], there are two kinds of evidences that can be used in NER to solve the ambiguity, robustness and portability problems described above. The first is the internal evidence found within the word and/or word string itself while the second is the external evidence gathered from its context. In order to effectively apply and integrate internal and external evidences, we present a NER system using a HMM. The approach behind our NER system is based on the HMM-based chunk tagger in text chunking, which was ranked the best individual system [Zhou+00a] [Zhou+00b] in CoNLL'2000 [Tjong+00]. Here, a NE is regarded as a chunk, named "NE-Chunk". To date, our system has been successfully trained and applied in English NER. To our knowledge, our system outperforms any published machine-learning systems. Moreover, our system even outperforms any published rule-based systems.

The layout of this paper is as follows. Section 2 gives a description of the HMM and its application in NER: HMM-based chunk tagger. Section 3 explains the word feature used to capture both the internal and external evidences. Section 4 describes the back-off schemes used to tackle the sparseness problem. Section 5 gives the experimental results of our system. Section 6 contains our remarks and possible extensions of the proposed work.

2　HMM-based Chunk Tagger

2.1　HMM Modeling

Given a token sequence $G_1^n = g_1 g_2 \cdots g_n$, the goal of NER is to find a stochastic optimal tag sequence $T_1^n = t_1 t_2 \cdots t_n$ that maximizes　　　(2-1)

$$\log P(T_1^n \mid G_1^n) = \log P(T_1^n) + \log \frac{P(T_1^n, G_1^n)}{P(T_1^n) \cdot P(G_1^n)}$$

The second item in (2-1) is the mutual information between T_1^n and G_1^n. In order to simplify the computation of this item, we assume mutual information independence:

$$MI(T_1^n, G_1^n) = \sum_{i=1}^{n} MI(t_i, G_1^n) \qquad \text{or} \qquad (2\text{-}2)$$

$$\log \frac{P(T_1^n, G_1^n)}{P(T_1^n) \cdot P(G_1^n)} = \sum_{i=1}^{n} \log \frac{P(t_i, G_1^n)}{P(t_i) \cdot P(G_1^n)} \quad (2\text{-}3)$$

Applying it to equation (2.1), we have:

$$\log P(T_1^n \mid G_1^n) = \log P(T_1^n) - \sum_{i=1}^{n} \log P(t_i)$$
$$+ \sum_{i=1}^{n} \log P(t_i \mid G_1^n) \qquad (2\text{-}4)$$

The basic premise of this model is to consider the raw text, encountered when decoding, as though it had passed through a noisy channel, where it had been originally marked with NE tags. The job of our generative model is to directly generate the original NE tags from the output words of the noisy channel. It is obvious that our generative model is reverse to the generative model of traditional HMM[1], as used

---

[1] In traditional HMM to maximise $\log P(T_1^n \mid G_1^n)$, first we apply Bayes' rule:

$$P(T_1^n \mid G_1^n) = \frac{P(T_1^n, G_1^n)}{P(G_1^n)}$$

and have:

in BBN's IdentiFinder, which models the original process that generates the NE-class annotated words from the original NE tags. Another difference is that our model assumes mutual information independence (2-2) while traditional HMM assumes conditional probability independence (I-1). Assumption (2-2) is much looser than assumption (I-1) because assumption (I-1) has the same effect with the sum of assumptions (2-2) and (I-3)[2]. In this way, our model can apply more context information to determine the tag of current token.

From equation (2-4), we can see that:
1) The first item can be computed by applying chain rules. In ngram modeling, each tag is assumed to be probabilistically dependent on the N-1 previous tags.
2) The second item is the summation of log probabilities of all the individual tags.
3) The third item corresponds to the "lexical" component of the tagger.

We will not discuss both the first and second items further in this paper. This paper will focus on the third item $\sum_{i=1}^{n} \log P(t_i \mid G_1^n)$, which is the main difference between our tagger and other traditional HMM-based taggers, as used in BBN's IdentiFinder. Ideally, it can be estimated by using the forward-backward algorithm [Rabiner89] recursively for the 1st-order [Rabiner89] or 2nd-order HMMs [Watson+92]. However, an alternative back-off modeling approach is applied instead in this paper (more details in section 4).

2.2 HMM-based Chunk Tagger

$\arg\max_{T} \log P(T_1^n \mid G_1^n)$

$= \arg\max_{T} (\log P(G_1^n \mid T_1^n) + \log P(T_1^n))$

Then we assume conditional probability independence: $P(G_1^n \mid T_1^n) = \prod_{i=1}^{n} P(g_i \mid t_i)$ (I-1)
and have:

$\arg\max_{T} \log P(T_1^n \mid G_1^n)$

$= \arg\max_{T} (\sum_{i=1}^{n} \log P(g_i \mid t_i) + \log P(T_1^n))$ (I-2)

[2] We can obtain equation (I-2) from (2.4) by assuming $\log P(t_i \mid G_1^n) = \log P(g_i \mid t_i)$ (I-3)

For NE-chunk tagging, we have token $g_i = < f_i, w_i >$, where $W_1^n = w_1 w_2 \cdots w_n$ is the word sequence and $F_1^n = f_1 f_2 \cdots f_n$ is the word-feature sequence. In the meantime, NE-chunk tag t_i is structural and consists of three parts:
1) **Boundary Category**: BC = {0, 1, 2, 3}. Here 0 means that current word is a whole entity and 1/2/3 means that current word is at the beginning/in the middle/at the end of an entity.
2) **Entity Category**: EC. This is used to denote the class of the entity name.
3) **Word Feature**: WF. Because of the limited number of boundary and entity categories, the word feature is added into the structural tag to represent more accurate models.

Obviously, there exist some constraints between t_{i-1} and t_i on the boundary and entity categories, as shown in Table 1, where "valid" / "invalid" means the tag sequence $t_{i-1}t_i$ is valid / invalid while "valid on" means $t_{i-1}t_i$ is valid with an additional condition $EC_{i-1} = EC_i$. Such constraints have been used in Viterbi decoding algorithm to ensure valid NE chunking.

	0	1	2	3
0	Valid	Valid	Invalid	Invalid
1	Invalid	Invalid	Valid on	Valid on
2	Invalid	Invalid	Valid	Valid
3	Valid	Valid	Invalid	Invalid

Table 1: Constraints between t_{i-1} and t_i (Column: BC_{i-1} in t_{i-1}; Row: BC_i in t_i)

3   Determining Word Feature

As stated above, token is denoted as ordered pairs of word-feature and word itself: $g_i = < f_i, w_i >$. Here, the word-feature is a simple deterministic computation performed on the word and/or word string with appropriate consideration of context as looked up in the lexicon or added to the context.

In our model, each word-feature consists of several sub-features, which can be classified into internal sub-features and external sub-features. The internal sub-features are found within the word and/or word string itself to capture internal evidence while external sub-features are derived within the context to capture external evidence.

3.1 Internal Sub-Features

Our model captures three types of internal sub-features: 1) f^1: simple deterministic internal feature of the words, such as capitalization and digitalization; 2) f^2: internal semantic feature of important triggers; 3) f^3: internal gazetteer feature.

1) f^1 is the basic sub-feature exploited in this model, as shown in Table 2 with the descending order of priority. For example, in the case of non-disjoint feature classes such as ContainsDigitAndAlpha and ContainsDigitAndDash, the former will take precedence. The first eleven features arise from the need to distinguish and annotate monetary amounts, percentages, times and dates. The rest of the features distinguish types of capitalization and all other words such as punctuation marks. In particular, the FirstWord feature arises from the fact that if a word is capitalized and is the first word of the sentence, we have no good information as to why it is capitalized (but note that AllCaps and CapPeriod are computed before FirstWord, and take precedence.) This sub-feature is language dependent. Fortunately, the feature computation is an extremely small part of the implementation. This kind of internal sub-feature has been widely used in machine-learning systems, such as BBN's IdendiFinder and New York Univ.'s MENE. The rationale behind this sub-feature is clear: a) capitalization gives good evidence of NEs in Roman languages; b) Numeric symbols can automatically be grouped into categories.

2) f^2 is the semantic classification of important triggers, as seen in Table 3, and is unique to our system. It is based on the intuitions that important triggers are useful for NER and can be classified according to their semantics. This sub-feature applies to both single word and multiple words. This set of triggers is collected semi-automatically from the NEs and their local context of the training data.

3) Sub-feature f^3, as shown in Table 4, is the internal gazetteer feature, gathered from the look-up gazetteers: lists of names of persons, organizations, locations and other kinds of named entities. This sub-feature can be determined by finding a match in the gazetteer of the corresponding NE type where n (in Table 4) represents the word number in the matched word string. In stead of collecting gazetteer lists from training data, we collect a list of 20 public holidays in several countries, a list of 5,000 locations from websites such as GeoHive[3], a list of 10,000 organization names from websites such as Yahoo[4] and a list of 10,000 famous people from websites such as Scope Systems[5]. Gazetters have been widely used in NER systems to improve performance.

3.2 External Sub-Features

For external evidence, only one external macro context feature f^4, as shown in Table 5, is captured in our model. f^4 is about whether and how the encountered NE candidate is occurred in the list of NEs already recognized from the document, as shown in Table 5 (n is the word number in the matched NE from the recognized NE list and m is the matched word number between the word string and the matched NE with the corresponding NE type.). This sub-feature is unique to our system. The intuition behind this is the phenomena of name alias.

During decoding, the NEs already recognized from the document are stored in a list. When the system encounters a NE candidate, a name alias algorithm is invoked to dynamically determine its relationship with the NEs in the recognized list.

Initially, we also consider part-of-speech (POS) sub-feature. However, the experimental result is disappointing that incorporation of POS even decreases the performance by 2%. This may be because capitalization information of a word is submerged in the muddy of several POS tags and the performance of POS tagging is not satisfactory, especially for unknown capitalized words (since many of NEs include unknown capitalized words.). Therefore, POS is discarded.

---

[3] http://www.geohive.com/

[4] http://www.yahoo.com/

[5] http://www.scopesys.com/

Sub-Feature f^1	Example	Explanation/Intuition
OneDigitNum	9	Digital Number
TwoDigitNum	90	Two-Digit year
FourDigitNum	1990	Four-Digit year
YearDecade	1990s	Year Decade
ContainsDigitAndAlpha	A8956-67	Product Code
ContainsDigitAndDash	09-99	Date
ContainsDigitAndOneSlash	3/4	Fraction or Date
ContainsDigitAndTwoSlashs	19/9/1999	DATE
ContainsDigitAndComma	19,000	Money
ContainsDigitAndPeriod	1.00	Money, Percentage
OtherContainsDigit	123124	Other Number
AllCaps	IBM	Organization
CapPeriod	M.	Person Name Initial
CapOtherPeriod	St.	Abbreviation
CapPeriods	N.Y.	Abbreviation
FirstWord	First word of sentence	No useful capitalization information
InitialCap	Microsoft	Capitalized Word
LowerCase	Will	Un-capitalized Word
Other	$	All other words

Table 2: Sub-Feature f^1: the Simple Deterministic Internal Feature of the Words

NE Type (No of Triggers)	Sub-Feature f^2	Example	Explanation/Intuition
PERCENT (5)	SuffixPERCENT	%	Percentage Suffix
MONEY (298)	PrefixMONEY	$	Money Prefix
	SuffixMONEY	Dollars	Money Suffix
DATE (52)	SuffixDATE	Day	Date Suffix
	WeekDATE	Monday	Week Date
	MonthDATE	July	Month Date
	SeasonDATE	Summer	Season Date
	PeriodDATE1	Month	Period Date
	PeriodDATE2	Quarter	Quarter/Half of Year
	EndDATE	Weekend	Date End
	ModifierDATE	Fiscal	Modifier of Date
TIME (15)	SuffixTIME	a.m.	Time Suffix
	PeriodTime	Morning	Time Period
PERSON (179)	PrefixPERSON1	Mr.	Person Title
	PrefixPERSON2	President	Person Designation
	FirstNamePERSON	Micheal	Person First Name
LOC (36)	SuffixLOC	River	Location Suffix
ORG (177)	SuffixORG	Ltd	Organization Suffix
Others (148)	Cardinal, Ordinal, etc.	Six,, Sixth	Cardinal and Ordinal Numbers

Table 3: Sub-Feature f^2: the Semantic Classification of Important Triggers

NE Type (Size of Gazetteer)	Sub-Feature f^3	Example
DATE (20)	DATEnGn	Christmas Day: DATE2G2
PERSON (10,000)	PERSONnGn	Bill Gates: PERSON2G2
LOC (5,000)	LOCnGn	Beijing: LOC1G1
ORG (10,000)	ORGnGn	United Nation: ORG2G2

Table 4: Sub-Feature f^3: the Internal Gazetteer Feature (G means Global gazetteer)

NE Type	Sub-Feature	Example
PERSON	PERSONnLm	Gates: PERSON2L1 ("Bill Gates" already recognized as a person name)
LOC	LOCnLm	N.J.: LOC2L2 ("New Jersey" already recognized as a location name)
ORG	ORGnLm	UN: ORG2L2 ("United Nation" already recognized as a org name)

Table 5: Sub-feature f^4: the External Macro Context Feature (L means Local document)

4 Back-off Modeling

Given the model in section 2 and word feature in section 3, the main problem is how to compute $\sum_{i=1}^{n} P(t_i / G_1^n)$. Ideally, we would have sufficient training data for every event whose conditional probability we wish to calculate. Unfortunately, there is rarely enough training data to compute accurate probabilities when decoding on new data, especially considering the complex word feature described above. In order to resolve the sparseness problem, two levels of back-off modeling are applied to approximate $P(t_i / G_1^n)$:

1) First level back-off scheme is based on different contexts of word features and words themselves, and G_1^n in $P(t_i / G_1^n)$ is approximated in the descending order of $f_{i-2}f_{i-1}f_i w_i$, $f_i w_i f_{i+1}f_{i+2}$, $f_{i-1}f_i w_i$, $f_i w_i f_{i+1}$, $f_{i-1}w_{i-1}f_i$, $f_i f_{i+1}w_{i+1}$, $f_{i-2}f_{i-1}f_i$, $f_i f_{i+1}f_{i+2}$, $f_i w_i$, $f_{i-2}f_{i-1}f_i$, $f_i f_{i+1}$ and f_i.

2) The second level back-off scheme is based on different combinations of the four sub-features described in section 3, and f_k is approximated in the descending order of $f_k^1 f_k^2 f_k^3 f_k^4$, $f_k^1 f_k^3$, $f_k^1 f_k^4$, $f_k^1 f_k^2$ and f_k^1.

5 Experimental Results

In this section, we will report the experimental results of our system for English NER on MUC-6 and MUC-7 NE shared tasks, as shown in Table 6, and then for the impact of training data size on performance using MUC-7 training data. For each experiment, we have the MUC dry-run data as the held-out development data and the MUC formal test data as the held-out test data.

For both MUC-6 and MUC-7 NE tasks, Table 7 shows the performance of our system using MUC evaluation while Figure 1 gives the comparisons of our system with others. Here, the precision (P)

measures the number of correct NEs in the answer file over the total number of NEs in the answer file and the recall (R) measures the number of correct NEs in the answer file over the total number of NEs in the key file while F-measure is the weighted harmonic mean of precision and recall: $F = \frac{(\beta^2 + 1)RP}{\beta^2 R + P}$ with $\beta^2 = 1$. It shows that the performance is significantly better than reported by any other machine-learning system. Moreover, the performance is consistently better than those based on handcrafted rules.

Statistics (KB)	Training Data	Dry Run Data	Formal Test Data
MUC-6	1330	121	124
MUC-7	708	156	561

Table 6: Statistics of Data from MUC-6 and MUC-7 NE Tasks

	F	P	R
MUC-6	96.6	96.3	96.9
MUC-7	94.1	93.7	94.5

Table 7: Performance of our System on MUC-6 and MUC-7 NE Tasks

| Composition	F	P	R
$f = f^1$	77.6	81.0	74.1
$f = f^1 f^2$	87.4	88.6	86.1
$f = f^1 f^2 f^3$	89.3	90.5	88.2
$f = f^1 f^2 f^4$	92.9	92.6	93.1
$f = f^1 f^2 f^3 f^4$	94.1	93.7	94.5

Table 8: Impact of Different Sub-Features

With any learning technique, one important question is how much training data is required to achieve acceptable performance. More generally how does the performance vary as the training data size changes? The result is shown in Figure 2 for MUC-7 NE task. It shows that 200KB of training data would have given the performance of 90% while reducing to 100KB would have had a significant decrease in the performance. It also shows that our system still has some room for performance improvement. This may be because of

the complex word feature and the corresponding sparseness problem existing in our system.
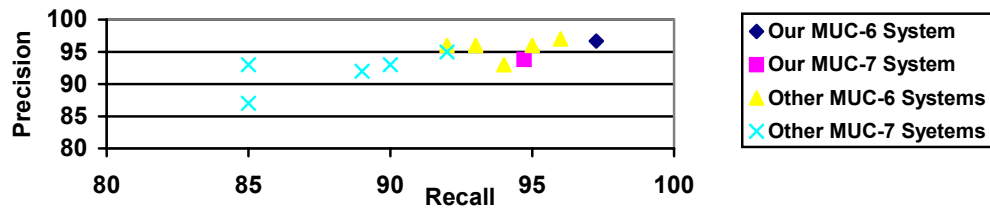
Figure 1: Comparison of our system with others on MUC-6 and MUC-7 NE tasks
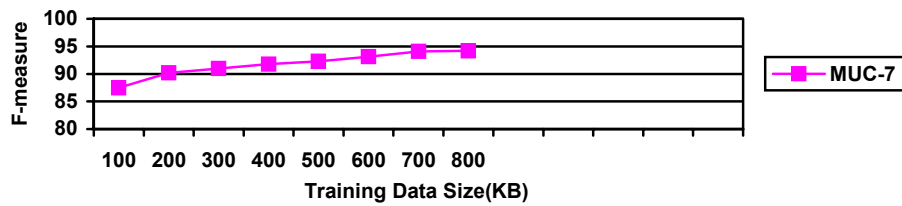
Figure 2: Impact of Various Training Data on Performance

Another important question is about the effect of different sub-features. Table 8 answers the question on MUC-7 NE task:

1) Applying only f^1 gives our system the performance of 77.6%.

2) f^2 is very useful for NER and increases the performance further by 10% to 87.4%.

3) f^4 is impressive too with another 5.5% performance improvement.

4) However, f^3 contributes only further 1.2% to the performance. This may be because information included in f^3 has already been captured by f^2 and f^4. Actually, the experiments show that the contribution of f^3 comes from where there is no explicit indicator information in/around the NE and there is no reference to other NEs in the macro context of the document. The NEs contributed by f^3 are always well-known ones, e.g. Microsoft, IBM and Bach (a composer), which are introduced in texts without much helpful context.

6   Conclusion

This paper proposes a HMM in that a new generative model, based on the mutual information independence assumption (2-3) instead of the conditional probability independence assumption (I-1) after Bayes' rule, is applied. Moreover, it shows that the HMM-based chunk tagger can effectively apply and integrate four different kinds of sub-features, ranging from internal word information to semantic information to NE gazetteers to macro context of the document, to capture internal and external evidences for NER problem. It also shows that our NER system can reach "near human performance". To our knowledge, our NER system outperforms any published machine-learning system and any published rule-based system.

While the experimental results have been impressive, there is still much that can be done potentially to improve the performance. In the near feature, we would like to incorporate the following into our system:

- List of domain and application dependent person, organization and location names.
- More effective name alias algorithm.
- More effective strategy to the back-off modeling and smoothing.

References

[Aberdeen+95] J. Aberdeen, D. Day, L. Hirschman, P. Robinson and M. Vilain. MITRE: Description of the Alembic System Used for MUC-6. *MUC-6*. Pages141-155. Columbia, Maryland. 1995.

[Aone+98] C. Aone, L. Halverson, T. Hampton, M. Ramos-Santacruz. SRA: Description of the IE2 System Used for MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[Bennett+96] S.W. Bennett, C. Aone and C. Lovell. Learning to Tag Multilingual Texts Through Observation. *EMNLP'1996*. Pages109-116. Providence, Rhode Island. 1996.

[Bikel+99] Daniel M. Bikel, Richard Schwartz and Ralph M. Weischedel. An Algorithm that Learns What's in a Name. *Machine Learning* (Special Issue on NLP). 1999.

[Borthwick+98] A. Borthwick, J. Sterling, E. Agichtein, R. Grishman. NYU: Description of the MENE Named Entity System as Used in MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[Borthwick99] Andrew Borthwick. A Maximum Entropy Approach to Named Entity Recognition. *Ph.D. Thesis*. New York University. September, 1999.

[Brill95] Eric Brill. Transform-based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-speech Tagging. *Computational Linguistics* 21(4). Pages543-565. 1995.

[Chinchor95a] Nancy Chinchor. MUC-6 Named Entity Task Definition (Version 2.1). *MUC-6*. Columbia, Maryland. 1995.

[Chinchor95b] Nancy Chinchor. Statistical Significance of MUC-6 Results. *MUC-6*. Columbia, Maryland. 1995.

[Chinchor98a] Nancy Chinchor. MUC-7 Named Entity Task Definition (Version 3.5). *MUC-7*. Fairfax, Virginia. 1998.

[Chinchor98b] Nancy Chinchor. Statistical Significance of MUC-7 Results. *MUC-7*. Fairfax, Virginia. 1998.

[Humphreys+98] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, Y. Wilks. Univ. of Sheffield: Description of the LaSIE-II System as Used for MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[Krupka+98] G. R. Krupka, K. Hausman. IsoQuest Inc.: Description of the NetOwl™ Extractor System as Used for MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[McDonald96] D. McDonald. Internal and External Evidence in the Identification and Semantic Categorization of Proper Names. In B. Boguraev and J. Pustejovsky editors: *Corpus Processing for Lexical Acquisition*. Pages21-39. MIT Press. Cambridge, MA. 1996.

[Miller+98] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, R. Weischedel, and the Annotation Group. BBN: Description of the SIFT System as Used for MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[Mikheev+98] A. Mikheev, C. Grover, M. Moens. Description of the LTG System Used for MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[Mikheev+99] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazeteers. *EACL'1999*. Pages1-8. Bergen, Norway. 1999.

[MUC6] Morgan Kaufmann Publishers, Inc. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Columbia, Maryland. 1995.

[MUC7] Morgan Kaufmann Publishers, Inc. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia. 1998.

[Rabiner89] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". *IEEE 77(*2). Pages257-285. 1989.

[Sekine98] Satoshi Sekine. Description of the Japanese NE System Used for MET-2. *MUC-7*. Fairfax, Virginia. 1998.

[Tjong+00] Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. *CoNLL'2000*. Pages127-132. Lisbon, Portugal. 11-14 Sept 2000.

[Viterbi67] A. J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*. IT(13). Pages260-269, April 1967.

[Watson+92] B. Watson and Tsoi A Chunk. Second Order Hidden Markov Models for Speech Recognition". *Proceeding of 4th Australian International Conference on Speech Science and Technology*. Pages146-151. 1992.

[Yu+98] Yu Shihong, Bai Shuanhu and Wu Paul. Description of the Kent Ridge Digital Labs System Used for MUC-7. *MUC-7*. Fairfax, Virginia. 1998.

[Zhou+00] Zhou GuoDong, Su Jian and Tey TongGuan. Hybrid Text Chunking. *CoNLL'2000*. Pages163-166. Lisbon, Portugal, 11-14 Sept 2000.

[Zhou+00b] Zhou GuoDong and Su Jian, Error-driven HMM-based Chunk Tagger with Context-dependent Lexicon. *EMNLP/ VLC'2000*. Hong Kong, 7-8 Oct 2000.