Natural Language Engineering **11** (2): 189–206.
doi:10.1017/S1351324904003559 Printed in the United Kingdom

Machine learning-based named entity recognition via effective integration of various evidences[1]

GUODONG ZHOU and JIAN SU

Institute for Infocomm Research, 21 Heng Mui Keng Terrace Singapore 119613
e-mail: {zhougd,sujian}@i2r.a-star.edu.sg

(*Received 31 October 2002; revised 11 March 2003*)

Abstract

Named entity recognition identifies and classifies entity names in a text document into some predefined categories. It resolves the "who", "where" and "how much" problems in information extraction and leads to the resolution of the "what" and "how" problems in further processing. This paper presents a Hidden Markov Model (HMM) and proposes a HMM-based named entity recognizer implemented as the system PowerNE.[2] Through the HMM and an effective constraint relaxation algorithm to deal with the data sparseness problem, PowerNE is able to effectively apply and integrate various internal and external evidences of entity names. Currently, four evidences are included: (1) a simple deterministic internal feature of the words, such as capitalization and digitalization;[3] (2) an internal semantic feature of the important triggers; (3) an internal gazetteer feature, which determines the appearance of the current word string in the provided gazetteer list; and (4) an external macro context feature, which deals with the name alias phenomena. In this way, the named entity recognition problem is resolved effectively. PowerNE has been benchmarked with the Message Understanding Conferences (MUC) data. The evaluation shows that, using the formal training and test data of the MUC-6 and MUC-7 English named entity tasks, and it achieves the F-measures of 96.6 and 94.1, respectively. Compared with the best reported machine learning system, it achieves a 1.7 higher F-measure with one quarter of the training data on MUC-6, and a 3.6 higher F-measure with one ninth of the training data on MUC-7. In addition, it performs slightly better than the best reported handcrafted rule-based systems on MUC-6 and MUC-7.

1 Introduction

Named entity recognition is a computational linguistics task, in which we seek to identify and classify words or strings of words in a text document into some predefined categories or "none-of-the-above". In the taxonomy of the computational linguistics tasks, it falls under the domain of "information extraction", which extracts specific kinds of information from text documents, as opposed to the more general

[1] A previous version of this paper appeared in ACL'2002 (Zhou and Su 2002).
[2] Power in PowerNE refers to I2R, the abbreviation of the Institute for Infocomm Research.
[3] The term « digitalization » refers to how the numbers in the text are interpreted.

task of "text interpretation", which seeks to extract all of information from text documents.

As entity names provide important content in a text document, named entity recognition is a very important step in information extraction. 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. Firstly, named entity recognition performs what is known as surface parsing, delimiting sequences of tokens that answer these important questions. Secondly, it can also be used as the first step in a chain of processes: a next level process may relate two or more named entities and give their semantic relationship. In this way, further processing could discover the "what" and "how" of a sentence or a body of the text document. Thirdly, named entity recognition can be useful in its own right: an Internet query system may use it to construct a more approximately formed query: "When was Bill Gates born?" could yield the query: "Bill Gates" + "born". Finally, it can be directly and/or indirectly employed for other information retrieval problems.

While named entity recognition is relatively simple and it is fairly easy to build a system with a reasonable performance, there still exist many problems of ambiguity, robustness and portability, which make it difficult to attain the human performance. For example, when is the word "Washington" used as the name of a person, when as the name of a city or state, and when as something else? There has been a considerable amount of work on such named entity recognition problems. During the last decade, named entity recognition has drawn more and more attention from the researchers of the named entity tasks (Chinchor 1995a, 1998a) of the Message Understanding Conferences (MUC6 1995; MUC7 1998), where person names, location names, organization names, dates, times, percentages and money amounts are to be delimited in text documents by means of the 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 does. Such typical systems include Univ. of Sheffield's LaSIE-II (Humphreys, Gaizauskas, Azzam, Huyck, Mitchell, Cunningham and Wilks 1998), SRA's system (Krupka 1995), ISOQuest Inc.'s NetOwl (Aone, Halverson, Hampton and Ramos-Santacruz 1998; Krupka and Hausman 1998) and the University of Edinburgh's LTG (Mikheev, Grover and Moens 1998; Mikheev, Moens and Grover 1999) for English named entity recognition. These systems are mainly based on handcrafted rules. However, handcrafted approaches lack the ability to cope with the problems of robustness and portability. Each new text source requires significant tweaking of handcrafted rules to maintain the optimal performance. This makes the maintenance cost quite steep.

The current trend in named entity recognition is to use machine-learning approaches, which are more attractive in that they are trainable and easy to adapt to a new domain with the increasing availability of annotated corpora. The representative machine-learning approaches used in named entity recognition are the HMM (BBN Technologies' IdentiFinder in Miller, Crystal, Fox, Ramshaw, Schwartz, Stone, Weischedel, and the Annotation Group (1998), and Bikel, Schwartz and Weischedel

(1999) for English named entity recognition, and KRDL's system in Yu, Bai and Wu (1998) for Chinese named entity recognition.), the maximum entropy (New York Univ.'s MEME in Borthwick, Sterling, Agichtein and Grishman (1998) and Borthwich (1999), and DSO's MENERGI in Chieu and Ng (2002)), and the decision tree (New York University's system in Sekine (1998) and SRA's system in Bennett, Aone and Lovell (1996)). Besides, a variant of Eric Brill's transformation-based learning algorithm (Brill 1995) has been applied to the problem (Aberdeen, Day, Hirschman, Robinson and Vilain 1995). 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 various phenomena, which indicate entity names in a text document. Moreover, the HMM seems more and more widely applied in named entity recognition because of the efficiency of the Viterbi algorithm (Viterbi 1967) used in decoding the entity name-class state sequence. However, the experiments in the newswire domain (MUC6 and MUC7) show that the performance of a machine-learning system (BBN Technologies' IdentiFinder in Bikel *et al.* (1999) and DSO's MENERGI in Chieu *et al.* (2002) on MUC-6; BBN Technologies' IdentiFinder in Miller *et al.* (1998) and New York University's MEME in Borthwick *et al.* (1998) on MUC-7) is always poorer than that of a handcrafted rule-based one (SRA's system in Krupka (1995) on MUC-6; University of Edinburgh's LTG in Mikheev *et al.* (1998) on MUC-7) by about 2% (Chinchor 1995b, 1998b). This may be because current machine-learning approaches capture important evidences much less effectively than human experts, despite the fact that machine-learning approaches always provide important statistical information that is not available to human experts.

As stated in McDonald (1996), there are two kinds of evidences that can be used to solve the ambiguity, robustness and portability problems in named entity recognition. The first is the internal evidence found within the word and/or the word string itself; and the second is the external evidence gathered from its context. In order to effectively apply and integrate various internal and external evidences, we present a HMM-based named entity recognizer implemented as the system PowerNE. The approach behind PowerNE is a HMM-based chunk tagger.[4] Here, an entity name is regarded as a chunk, named "NE-Chunk". Currently, four evidences are included: 1) a simple deterministic internal feature of the words, such as capitalization and digitalization; 2) an effective internal semantic feature of the important triggers; 3) an internal gazetteer feature, which determines whether and how the current word string appears in the provided gazetteer list; 4) an effective external macro context feature, which deals with the name alias phenomena. Furthermore, an effective constraint relaxation algorithm is proposed to deal with the data sparseness problem. To date, PowerNE has been successfully trained and applied in English named entity recognition. To our best knowledge, it outperforms any other published machine-learning system. Compared with the previous best machine learning system

[4] The HMM-based chunk tagger was ranked the best individual system (Zhou, Su and Tey 2000; Zhou and Su 2000) in the CoNLL'2000 text chunking competition (Tjong and Buchholz 2000).

(IdentiFinder of BBN Technologies in Miller *et al.* (1998) and Bikel *et al.* (1999)), it achieves a 1.7 higher F-measure with one quarter of the training data on MUC-6 and a 3.6 higher F-measure with one ninth of the training data on MUC-7. Moreover, it even performs slightly better than the best handcrafted rule-based systems on MUC-6 and MUC-7.

The layout of this paper is as follows. Section 2 gives a brief description of the HMM and its application in named entity recognition: a HMM-based NE-chunk tagger. In section 3, various features are proposed in PowerNE to capture both the internal and external evidences while in section 4, an effective constraint relaxation algorithm is proposed in PowerNE to tackle the data sparseness problem. The experimental results of PowerNE are given in section 5. In section 6, we give a brief comparison of our work with other related works. Finally, we present our conclusion and some future works.

2 HMM-based NE-chunk tagger

2.1 HMM

A Hidden Markov Model (HMM) is a model where a sequence of outputs is generated in addition to the Markov state sequence. It is a latent variable model in the sense that only the output sequence is observed while the state sequence remains "hidden".

Given an output sequence $O_1^n = o_1 o_2 \cdots o_n$, the goal of a HMM is to find a stochastic optimal tag (state) sequence $T_1^n = t_1 t_2 \cdots t_n$ that maximizes (Zhou and Su 2002)

$$\log P\left(T_1^n \middle| O_1^n\right) = \log P\left(T_1^n\right) + \log \frac{P\left(T_1^n, O_1^n\right)}{P\left(T_1^n\right) \cdot P\left(O_1^n\right)} \tag{2.1}$$

The second term in the equation (2.1) is the mutual information between T_1^n and O_1^n. In order to simplify the computation of this term, we assume the mutual information independence:

$$MI\left(T_1^n, O_1^n\right) = \sum_{i=1}^{n} MI\left(t_i, O_1^n\right) \text{ or}$$

$$\log \frac{P\left(T_1^n, O_1^n\right)}{P\left(T_1^n\right) \cdot P\left(O_1^n\right)} = \sum_{i=1}^{n} \log \frac{P\left(t_i, O_1^n\right)}{P\left(t_i\right) \cdot P\left(O_1^n\right)} \tag{2.2}$$

That is, an individual tag is only dependent on the output sequence O_1^n and independent of the other tags in the tag sequence T_1^n. This assumption is reasonable because the dependence among the tags in the tag sequence T_1^n has already been captured by the first term in the equation (2.1). Applying the assumption (2.2) to the equation (2.1), we have:

$$\log P\left(T_1^n | O_1^n\right) = \log P\left(T_1^n\right) - \sum_{i=1}^{n} \log P\left(t_i\right) + \sum_{i=1}^{n} \log P\left(t_i \middle| O_1^n\right) \tag{2.3}$$

From equation (2.3), we can see that:

- The first term can be computed by applying chain rules. In ngram modelling, each tag is assumed to be probabilistically dependent on the N-1 previous tags.
- The second term is the summation of log probabilities of all the individual tags.
- The third term corresponds to the "lexical" component (dictionary) of the tagger.

We will not discuss either the first or the second term further in this paper, because ngram modelling has been well studied in the literature (Katz 1987; Jelinek 1989). We will focus on the third term $\sum_{i=1}^{n} \log P(t_i|O_1^n)$. Ideally, it can be estimated by using the forward-backward algorithm (Rabiner 1989) recursively for the 1st-order (Rabiner 1989) or 2nd-order HMMs (Watson and Chunk 1992). For efficiency, an alternative back-off modelling approach by constraint relaxation is applied in this paper (see details in section 4).

2.2 HMM-based NE-chunk tagger

Given the previous HMM, for NE-chunk tagging, we have (Zhou and Su 2002):

- $W_1^n = w_1 w_2 \cdots w_n$ is the word sequence; $F_1^n = f_1 f_2 \cdots f_n$ is the feature set sequence and f_i is the feature set of the word w_i; the output $o_i = <f_i, w_i>$.
- the tag t_i: Here, an entity name is regarded as a chunk (called "NE-chunk") and the tags are used to bracket and differentiate various types of NE-chunks. For convenience, here the tag used in named entity recognition is called "NE-chunk tag". The NE-chunk tag t_i is structural and consists of three parts:
-

1. **Boundary Category (BOUNDARY)**: it is a set of four values: "whole"/"start"/"middle"/"end", where "whole" means that current word is a whole entity name and "start"/"middle"/"end" means that current word is at the start/in the middle/at the end of an entity name.
2. **Entity Category (ENTITY)**: used to denote the class of the entity name.
3. **Feature Set (FEATURE)**: Because of the limited number of boundary and entity categories, the feature set is added into the structural NE-chunk tag to represent more accurate models.

Obviously, there exist some constraints between NE-chunk tags $t_{i-1}(BOUNDARY_{i-1}\_ENTITY_{i-1}\_FEATURE_{i-1})$ and $t_i(BOUNDARY_i\_ENTITY_i\_FEATURE_i)$ 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 only when the condition $ENTITY_{i-1} = ENTITY_i$ is satisfied. Such constraints have been applied in the Viterbi decoding algorithm to ensure valid entity name tagging.

Table 1. *Boundary category constraints between the NE-chunk tags t_{i-1} and t_i (Column: $BOUNDARY_{i-1}$ in t_{i-1}; Row: $BOUNDARY_i$ in t_i)*

Boundary Cateogry	whole	start	middle	end
whole	Valid	Valid	Invalid	Invalid
start	Invalid	Invalid	Valid on	Valid on
middle	Invalid	Invalid	Valid on	Valid on
end	Valid	Valid	Invalid	Invalid

3 Determination of the feature set

As stated previously, any output o_i is denoted as an ordered pair of the word w_i itself and its feature set f_i: $o_i = <f_i, w_i>$. Here, the feature set is gathered from simple computations on the word and/or the word string with appropriate consideration of its context as looked up in the lexicon or added to the context.

In PowerNE, the feature set of a word consists of several internal and external features.

3.1 Internal features

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

3.1.1 f^1: the simple deterministic internal feature of the words

f^1 is the basic feature exploited in PowerNE, 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 characters such as punctuation marks. In particular, FirstWord arises from the fact that if a word is capitalized and it is the first word of a 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 feature is language dependent. Fortunately, the computation of this feature is an extremely small part of the implementation.

This feature has been widely used in typical machine-learning systems such as BBN's IdentiFinder and New York Univ.'s MEME. The rationale behind this feature is clear as follows:

- Capitalization gives a good evidence of entity names in Roman languages;
- Numeric symbols can automatically be grouped into categories.

Table 2. *Feature f^1: the simple deterministic internal feature of the words*

Feature f^1	Example	Explanation
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

3.1.2 f^2: the internal semantic feature of the important triggers

f^2 is the semantic classification of the important triggers, as seen in Table 3. It is based on the intuitions that the important triggers are useful for named entity recognition and can be classified according to their semantics. This feature applies to both a single word and a word string. The important triggers are collected semi-automatically. For a certain type of named entities, the triggers are first automatically collected from the headwords and the surrounding contexts of the entity names. Those frequently occurring triggers are then manually checked and classified further.

3.1.3 f^3: the internal gazetteer 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 types of named entities. This feature can be determined by finding a match in the gazetteer of the corresponding named entity type where n represents the word number in the matched word string.

Instead of collecting the gazetteer lists from the training data, we collect them from public resources: a list of 20 public holidays in several countries, a list of about 5,000 locations from websites such as GeoHive,[5] a list of about 10,000 organization

[5] `http://www.geohive.com/`.

Table 3. *Feature f^2: the semantic classification of the important triggers (the number in parentheses indicates the number of the important triggers for the corresponding type of named entities*

Named Entity Type	Feature f^2	Example	Explanation
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	Michael	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 4. *Feature f^3: the internal gazetteer feature (the number in parentheses indicates the size of the gazetteer for the corresponding type of named entities and the letter G means Global gazetteer)*

Named Entity Type	Feature f^3	Example
DATE (20)	DATE*n*G*n*	Christmas Day: DATE2G2
PERSON (10,000)	PERSON*n*G*n*	Bill Gates: PERSON2G2
LOC (5,000)	LOC*n*G*n*	Beijing: LOC1G1
ORG (10,000)	ORG*n*G*n*	United Nation: ORG2G2

names from websites such as Yahoo[6] and a list of about 10,000 famous people from websites such as Scope Systems.[7]

3.2 External features

Table 5 is captured in PowerNE. f^4 is about whether and how the encountered entity name candidate occurs in the list of entity names already recognized from the

[6] http://www.yahoo.com/.
[7] http://www.scopesys.com/.

Table 5. *Feature f^4: the external macro context feature (the letter **L** means **L**ocal document)*

Named Entity Type	Feature f^4	Example	Explanation
PERSON	PERSON*n*L*m*	Gates: PERSON2L1	"Bill Gates" already recognized as a person name
LOC	LOCnL*m*	N.J.: LOC2L2	"New Jersey" already recognized as a location name
ORG	ORGnL*m*	UN: ORG2L2	"United Nation" already recognized as an org name

document (*n* is the word number in the matched entity name from the recognized entity name list and *m* is the matched word number between the word string and the matched entity name with the corresponding named entity type.).

The intuition behind this feature is the name alias phenomena that relevant entities will be referred to in many ways throughout a given text and thus success of the named entity recognition task is conditional on success at determining when one noun phrase refers to the very same entity as another noun phrase.

During decoding, the entity names already recognized from the previous part of the document are stored in a list. When the system encounters an entity name candidate (e.g. a word or a sequence of words with the initial letters capitalized), a name alias algorithm is invoked to first dynamically determine whether the entity name candidate might be an alias for a previously recognized entity name in the recognized list and then decide the relationship between them. For example, when the decoding process encounters the word "UN", the word "UN" is proposed as an entity name candidate and the name alias algorithm is invoked to check if the word "UN" is an alias of a recognized entity name by taking the initial letters of a recognized entity name. If "United Nation" is an organization entity name recognized earlier in the document, the word "UN" is determined as an alias of "United Nation" with the external macro context feature ORG2L2.

Initially, we have also considered the part-of-speech (POS) together with f^1 (the simple deterministic internal feature of the words) and f^2 (the internal semantic feature of important triggers). However, the experiments show that incorporation of the POS decreases the performance by about 2%. This may be because capitalization information of a word is submerged by several POS tags and the performance of POS tagging is not satisfactory, especially for unknown capitalized words (since many entity names include unknown capitalized words.). Therefore, the POS is discarded from PowerNE.

4 Back-off Modelling

Given the model in section 2 and the feature set in section 3, the main task of PowerNE is how to compute $\sum_{i=1}^{n} P(t_i|O_1^n)$. Ideally, we should 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 on new data, especially in line with the complex feature set described above.

For efficiency, we assume $P(t_i|O_1^n) \approx P(t_i|E_i)$, where $E_i = o_{i-2}o_{i-1}o_io_{i+1}o_{i+2}$. That is, we only consider the context in a window of five words. Here, $o_i = <f_i, w_i>$, w_i is the current word and $f_i = <f_i^1, f_i^2, f_i^3, f_i^4>$ is the set of the four features described in section 3. In the meantime, we denote $P(\bullet|E_i)$ as the probability distribution of various NE-chunk tags given the pattern entry E_i and $P(t_i|E_i)$ as the probability of the NE-chunk tag t_i given E_i.

Computing $P(\bullet|E_i)$ becomes a problem of finding an optimal frequently occurring pattern entry E_i^0, which occurs at least N (e.g. 10) times in the training corpus and satisfies $P(\bullet|E_i) \approx P(\bullet|E_i^0)$. In this paper, a back-off modelling approach by constraint relaxation is proposed. Here, the constraints include all the f^1, f^2, f^3, f^4 and w (the subscripts are omitted) in E_i. With the large number of ways in which the constraints could be relaxed, the challenge is how to avoid intractability and drop in efficiency. Two restrictions are applied to keep the relaxation process tractable and manageable:

- Relaxation is done through iteratively dropping a constraint from the pattern entry.
- The pattern entry after relaxation should have a valid form, defined as $ValidEntryForm = \{f_{i-2}f_{i-1}f_iw_i,\ f_{i-1}f_iw_if_{i+1},\ f_iw_if_{i+1}f_{i+2},\ f_{i-1}f_iw_i,\ f_iw_if_{i+1},\ f_{i-1}w_{i-1}f_i,\ f_if_{i+1}w_{i+1},\ f_{i-2}f_{i-1}f_i,\ f_{i-1}f_if_{i+1},\ f_if_{i+1}f_{i+2},\ f_iw_i,\ f_{i-1}f_i,\ f_if_{i+1},\ f_i\}$. In the meantime, each f_k in the pattern entry after relaxation should have a valid form, defined as $ValidFeatureForm = \{<f_k^1, f_k^2, f_k^3, f_k^4>,\ <f_k^1, f_k^2, f_k^3, \Phi>,\ <f_k^1, f_k^2, \Phi, f_k^4>, <f_k^1, f_k^2, \Phi, \Phi>, <f_k^1, \Phi, f_k^3, \Phi\}>, <f_k^1, \Phi, \Phi, f_k^4>, <f_k^1, \Phi, \Phi, \Phi>\}$. Φ means empty (dropped or not available)

By means of the constraint relaxation principle, the back-off modelling approach estimates $P(\bullet|E_i)$ as follows:

The constraint relaxation algorithm for computing $P(\bullet|E_i)$

Assume an initial pattern entry $E_i = o_{i-2}o_{i-1}o_io_{i+1}o_{i+2}$, where $o_i = <f_i, w_i>$ and $f_i = <f_i^1, f_i^2, f_i^3, f_i^4>$.

Assume a valid set of the pattern entry forms $ValidEntryForm = \{f_{i-2}f_{i-1}f_iw_i,\ f_{i-1}f_iw_if_{i+1},\ f_iw_if_{i+1}f_{i+2},\ f_{i-1}f_iw_i,\ f_iw_if_{i+1},\ f_{i-1}w_{i-1}f_i,\ f_if_{i+1}w_{i+1},\ f_{i-2}f_{i-1}f_i,\ f_{i-1}f_if_{i+1},\ f_if_{i+1}f_{i+2},\ f_iw_i,\ f_{i-1}f_i,\ f_if_{i+1},\ f_i\}$

Assume a valid set of the feature set forms $ValidFeatureForm = \{<f_k^1, f_k^2, f_k^3, f_k^4>,\ <f_k^1, \Phi, f_k^3, \Phi\}>, <f_k^1, \Phi, \Phi, f_k^4>, <f_k^1, f_k^2, \Phi, \Phi¿, <f_k^1, \Phi, \Phi, \Phi>\}$.

Assume *FrequentEntryDictionary* is the pattern entry dictionary which stores all the frequently occurring pattern entries with the probability distributions of various NE-chunk tags

Assume $likelihood(E_i)$ the likelihood of a pattern entry E_i as the optimal frequently occurring pattern entry to be found.

$$likelihood(E_i) = \frac{number\ of\ f^2, f^3\ and\ f^4\ in\ E_i + 0.1}{number\ of\ f^1, f^2, f^3, f^4\ and\ w\ in\ E_i}$$

Assume E_i^0 is the optimal frequently occurring pattern entry to be found.
BEGIN-of-algorithm
IF $E_i \in FrequentEntryDictionary$ **THEN**
BEGIN
  Set $E_i^0 = E_i$
  RETURN $P(\bullet|E_i) = P(\bullet|E_i^0)$
END IF-THEN
Compute $likelihood(E_i)$
Initialize$InputEntrySet = \{<E_i, likelihood(E_i)>\}$ and $OutputEntrySet = \{\}$
LOOP
  DO for every entry $<E_j, likelihood(E_j)> \in InputEntrySet$
    DO for every constraint in E_j
      Assume C_j^k is the constraint in E_j
      Set $E_j' = E_j - C_j^k$ as the pattern entry after relaxation of the constraint C_j^k in E_j
      IF E_j' conforms to $ValidFeatureForm$ **THEN**
      BEGIN
        Compute $likelihood(E_j')$
        Set
        $OutputEntrySet = OutputEntrySet \cup \{<E_j', likelihood(E_j')>\}$
      END IF-THEN
    END DO
  END DO
  Get$E_i^0 = \underset{\substack{<E, likelihood(E)> \in OutputEntrySet \\ E \in FrequentEntryDictionary \\ E \text{ conforms to } ValidEntryForm}}{\arg\max} likelihood(E)$
  IF E_i^0 is not found
  THEN Set $InputEntrySet = OutputEntrySet$ and $OutputEntrySet = \{\}$
  ELSE exit LOOP
END LOOP
RETURN $P(\bullet|E_i) = P(\bullet|E_i^0)$
END-of-algorithm

The previous constraint relaxation algorithm solves the problem by iteratively relaxing a constraint in the initial pattern entry E_i until a near optimal frequently occurring pattern entry E_i^0 is reached. If E_i frequently occurs, we just return E_i as E_i^0. Otherwise, a near optimal entry E_i^0 will be found by iteratively relaxing the initial entry E_i. At each iteration, we have a set of entries *InputEntrySet* as the input and return a set of entries *OutputEntrySet* as the output, which is the input for the next iteration. The entries in *InputEntrySet* and *OutputEntrySet* are ranked according to their likelihoods[8] as the optimal frequently occurring pattern entry to be found. In

[8] The likelihood of a pattern entry is simply determined by the percentage of the features f^2, f^3 and f^4 in it. The intuition comes from that the semantic feature of the important

this case, *InputEntrySet* is initialized as $\{<E_i, likelihood(E_i)>\}$ and *OutputEntrySet* can be generated by relaxing any constraint (validated by *ValidFeatureForm*) in every pattern entry of *InputEntrySet*. If no pattern entry in *OutputEntrySet* frequently occurs, the pattern entries in *OutputEntrySet* is fed back to *InputEntrySet* for the next iteration and a further set of pattern entries *OutputEntrySet* can be generated by relaxing any constraint (again, validated by *ValidFeatureForm*) in each pattern entry of *InputEntrySet*. The process continues until E_i^0 is found. In order to remain efficient, only the top N (e.g. 5) pattern entries are kept in *InputEntrySet* and *OutputEntrySet* according to their likelihoods.

To better understand the algorithm, let's look at an example step by step. Given a sentence:

Ms. Washington said there were 20 students in her class.

For simplicity, we only consider a window size of three (instead of five) and only the top three pattern entries are kept according to their likelihoods. Assume the current word is "Washington". From Tables 2 to 5, we can have an initial pattern entry $E_2 = o_1 o_2 o_3$,[9] where

$$o_1 = <f_1^1 = CapOtherPeriod, f_1^2 = \Pr efixPerson1, f_1^3 = \Phi, f_1^4 = \Phi, w_1 = Ms.>$$
$$o_2 = <f_2^1 = InitialCap, f_2^2 = \Phi, f_2^3 = PER2L1, f_2^4 = LOC1G1, w_2 = Washington>$$
$$o_3 = <f_3^1 = LowerCase, f_3^2 = \Phi, f_3^3 = \Phi, f_3^4 = \Phi, w_3 = said>$$

First, the algorithm looks up the entry E_2 in *FrequentEntryDictionary*. If the entry is found, it means the entry E_2 frequently occurs in the training corpus and we can simply return it as the optimal frequently occurring pattern entry. In this example, the entry E_2 is not found in *FrequentEntryDictionary*. Therefore, the generalization process begins by relaxing the constraints. This is done by dropping one constraint at each iteration. For the entryE_2, there are nine possible generalized entries since there are nine non-empty constraints. However, only six of them are valid according to *ValidFeatureForm*. Then the likelihoods of the six valid entries are computed and only the top three generalized entries are kept: $E_2 - w1$ with the likelihood 0.34, $E_2 - w2$ with the likelihood 0.34 and $E_2 - w3$ with the likelihood 0.34. Again, the three generalized entries are checked whether they exist in *FrequentEntryDictionary*. However, none of them is found and the above generalization process continues for each of the three generalized entries. After five generalization processes, we will have a generalized entry $E_2 - w_1 - w_2 - w_3 - f_1^3 - f_2^4$ with the top likelihood 0.5. Since this entry is found in *FrequentEntryDictionary*, the generalized entry

triggers (f^2), the internal gazetteer feature (f^3) and the external macro context feature (f^4) are more informative in determining entity names than the internal feature of capitalization and digitalization (f^1) and the words themselves (w). To guarantee the likelihood is bigger than zero if the proposed pattern entry frequently occurs, the number 0.1 is added in the likelihood computation of a proposed pattern entry.

[9] Assume that "Washington" exists in the LOCATION gazetteer and that some expanded form, e.g. "Susan Washington", occurs in a previous sentence of the document and has been recognized as a person name. Therefore, the word "Washington" has the gazetteer feature of LOC1G1 and the macro context feature of PER2L1. See Tables 4 and 5 for details about the two features.

Table 6. *Statistics of data from the MUC-6 and MUC-7 named entity tasks*

Statistics (K Words)	Training Data	Dry Run Data	Formal Test Data
MUC-6	162	15	15
MUC-7	86	19	68

Table 7. *Performance of PowerNE on the MUC-6 and MUC-7 named entity tasks*

Named Entity Task	F-measure	Precision	Recall
MUC-6	96.6	96.3	96.9
MUC-7	94.1	93.7	94.5

$E_2 - w_1 - w_2 - w_3 - f_1^3 - f_2^4$ is returned as the optimal frequently occurring pattern entry with the probability distribution of various NE-chunk tags.

5 Experimental results

In this section, we report the experimental results of PowerNE for English named entity recognition on the MUC-6 and MUC-7 shared tasks, as shown in Table 6. Then we will present the impact of different training data sizes on the performance using the MUC-7 training data. For each experiment, we have the MUC formal training data as the only training data, the MUC dry-run data as the held-out development data, and the MUC formal test data as the held-out test data while all the performances are evaluated using the MUC standard evaluation program.

For both the MUC-6 and MUC-7 named entity tasks, Table 7 shows the performance of PowerNE. Here, the precision (P) measures the number of the correct entity names in the answer file over the total number of the entity names in the answer file; the recall (R) measures the number of the correct entity names in the answer file over the total number of the entity names in the key file which holds the original entity names; and the F-measure (van Rijsbergen 1979) is the weighted harmonic mean of the precision (P) and recall (R): $F = \frac{(\beta^2+1)RP}{\beta^2 R+P}$ with $\beta^2=1$.

Table 7 shows that, using the formal training and test data of the MUC-6 and MUC-7 English named entity tasks, PowerNE achieves the F-measures of 96.6 and 94.1 respectively. Figure 1 shows a comparison of PowerNE with the other best-reported systems using the MUC benchmark standard test data and the MUC standard evaluation program. Here, the other best-reported systems include both the machine-learning-based systems (Bikel *et al.* 1999; Chieu *et al.* 2002, on MUC-6; Miller *et al.* 1998; Borthwick *et al.* 1998, on MUC-7) and the handcrafted rule-based systems (Krupka 1995, on MUC-6; Mikheev *et al.* 1998, on MUC-7). Figure 1 shows that PowerNE performs significantly better than the other machine-learning systems. Moreover, it also shows that PowerNE even performs consistently better than the handcrafted rule-based systems. Finally, it is interesting to note from Figure 1 that

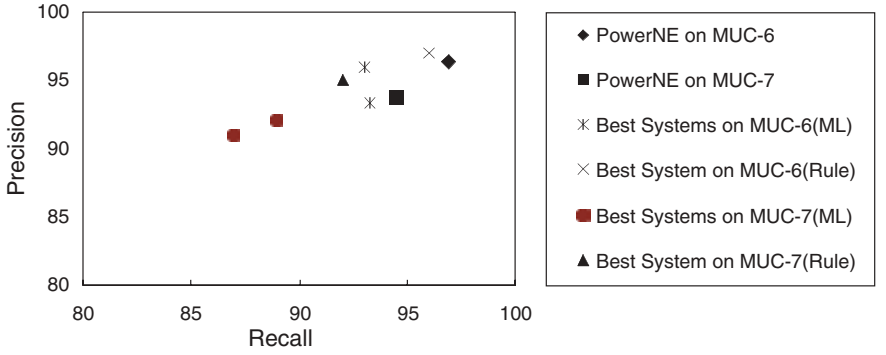

Fig. 1. Comparison of PowerNE on the MUC-6 and MUC-7 tasks with others using the MUC standard test data and the MUC standard evaluation program ("ML": machine learning-based; "Rule": handcrafted rule-based).

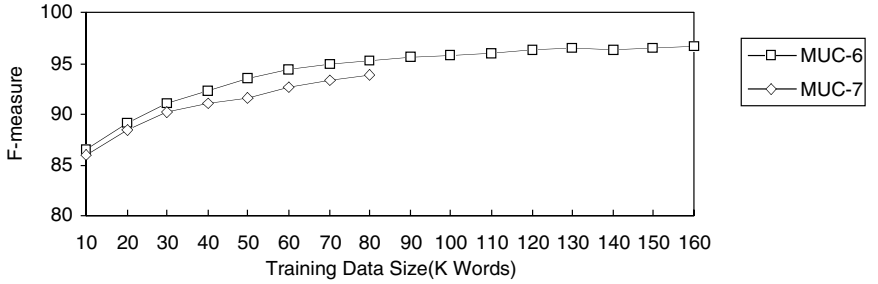

Fig. 2. Impact of different training data sizes on PowerNE evaluated on the MUC-6 and MUC-7 named entity tasks.

our better performance mainly comes from the higher recall compared with those of the other systems. This may be due to the rich features employed in PowerNE, which extend the coverage of entity names.

With any learning techniques, one important question is how much training data is required to achieve an acceptable performance. More generally how does the performance vary as the training data size changes? The answer is shown in Figure 2 for the impact of different training data sizes on PowerNE evaluated on the MUC-6 and MUC-7 named entity tasks. It shows that 30 K words of training data would have given the F-measure of 90, while reducing to 10 K words would have had a significant decrease in the performance. The trend in Figure 2 also shows that PowerNE still has some room of performance improvement if more training data are available.

Another important question is about the effect of the different features in PowerNE. Table 8 answers the question on the MUC-7 named entity task:

- Applying only f^1 gives the F-measure of 77.6.
- f^2 is very useful for named entity recognition and increases the F-measure further by 10 to 87.4.
- f^4 is impressive too with another 5.5 F-measure improvement.

Table 8. *Impact of the different features on MUC-7 named entity task*

Named Entity Task	F-measure	Precision	Recall
$f = f^1$	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

- However, f^3 contributes only further 1.2 to the F-measure. 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 entity name and there is no reference to other entity names in the macro context of the document. The entity names 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 information.

6 Related work

There have been a few attempts in applying machine learning approaches to the task of named entity recognition. Among them, the representative research efforts include Aberdeen *et al.* (1995), Bennett *et al.* (1996), Borthwich *et al.* (1998, 1999), Miller *et al.* (1998), Bikel *et al.* (1999) and Chien *et al.* (2002). All of them employ similar features of f^1 (the internal deterministic feature of the words, such as capitalization and digitalization) and f^3 (the internal gazetteer feature), and w (the word itself).

Aberdeen *et al.* (1995) employ Eric Brill's transformation-based learning algorithm (Brill 1995), and report the F-measure of about 85 on MUC-6. Bennett *et al.* (1996) use decision trees and achieves the F-measure of about 91 on MUC-6. MENE of New York Univ. in Borthwich *et al.* (1998, 1999) apply the maximum entropy principle and achieves the F-measure of about 89 on MUC-7. Chieu *et al.* (2002) also applies the maximum entropy principle and achieves the F-measures of 93 and 87 on MUC-6 and MUC-7, respectively. All of them use the MUC formal training data as the only training data, the MUC dry-run data as the held-out development data, and the MUC formal test data as the held-out test data.

IdentiFinder of BBN Technologies in Bikel *et al.* (1999) and Miller *et al.* (1998) present a similar HMM that learns to identify and classify entity names. It achieves so far the best F-measures on MUC-6 and MUC-7 for a machine learning approach. Bikel *et al.* (1999) use 650 K words of training data (four times the MUC-6 formal training data) and achieves the F-measure of 94.9 on MUC-6 while Miller *et al.* (1998) uses 790 K words of training data (nine times the MUC-7 formal training data) and achieve the F-measure of 90.4. However, IdentiFinder in Bikel *et al.* (1999) only has the F-measure of 91.8 when the size of the training data decreases to the same size of the MUC-6 formal training data.

Compared with the best reported machine learning system (IdentiFinder of BBN Technologies in Miller *et al.* (1998) and Bikel *et al.* (1999)), PowerNE achieves a 1.7 higher F-measure with one fourth of the training data on MUC-6 and a 3.6 higher F-measure with one ninth of the training data on MUC-7. The contribution of our work lies in the effective features of f^2 (the semantic feature of the important triggers) and f^4 (the external macro context feature). Through the semantic classification of the important triggers, f^2 not only increases the F-measure but also decreases the requirement of the training data size while f^4 effectively deals with the name alias phenomena. Another contribution is the proposal of a back-off modelling approach by means of constraint relaxation in dealing with the data sparseness problem in such a rich feature matrices. Such an approach enables the decoding process to effectively find a near optimal frequently occurring pattern entry in determining the probability distribution of various NE-chunk tags of the current word.

7 Conclusion

In this paper, we present a machine learning approach to named entity recognition. Various features are applied and integrated effectively to capture internal and external evidences for the named entity recognition problem. In addition, an effective back-off modelling approach by constraint relaxation is proposed to deal with the data sparseness problem. As a result, a named entity recognition system with better performance and better portability (with much less training data) is achieved.

At this stage, PowerNE can only work with mixed case texts. Future works include the exploration of new evidences in named entity recognition and the adaptation of our system to upper case texts and, if possible, speech transcription.

Acknowledgements

The authors would like to thank the three anonymous reviewers for their valuable and detailed comments.

References

Aberdeen, J., Day, D., Hirschman, F., Robinson, P. and Vilain, M. (1995) MITRE: Description of the Alembic system used for MUC-6. *Proceedings Sixth Message Understanding Conference (MUC-6)*. pp. 141–155. Columbia, Maryland.

Aone, C., Halverson, L., Hampton, T. and Ramos-Santacruz, M. (1998) SRA: Description of the IE2 system used for MUC-7. *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Bennett, S. W., Aone, C. and Lovell, C. (1996) Learning to tag multilingual texts through observation. *Proceedings Second Conference on Empirical Methods in Natural Language Processing (EMNLP'1996)*. pp. 109–116. Providence, RI.

Bikel, D. M., Schwartz, R. and Weischedel, R. M. (1999) An algorithm that learns what's in a name. *Machine Learning* (Special Issue on NLP) **34**(3): 211–231.

Borthwick, A., Sterling, J., Agichtein, E. and Grishman, R. (1998) NYU: Description of the MEME named entity system as used in MUC-7. *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Borthwick, A. (1999) A maximum entropy approach to named entity recognition. *PhD Thesis*. New York University.

Brill, E. (1995) Transform-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* **21**(4): 543–565.

Chieu, H. L. and Ng, H. T. (2002) Named entity recognition: a maximum entropy approach using global information. *Proceedings 19th International Conference on Computational Linguistics (COLING'2002)*. Pp. 190–196. Taipei, Taiwan.

Chinchor, N. (1995) MUC-6 named entity task definition (version 2.1). *Proceedings Sixth Message Understanding Conference (MUC-6)*. pp. 15–21. Columbia, MD.

Chinchor, N. (1995) Statistical significance of MUC-6 results. *Proceedings Sixth Message Understanding Conference (MUC-6)*. pp. 39–43. Columbia, MD.

Chinchor, N. (1998) MUC-7 named entity task definition (version 3.5). *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Chinchor, N. (1998) Statistical significance of MUC-7 results. *Proceedings Sixth Message Understanding Conference (MUC-7)*. Fairfax, VA.

Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunningham, H. and Wilks, Y. (1998) University of Sheffield: Description of the LaSIE-II system as used for MUC-7. *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Jelinek, F. (1989) Self-Organized language modelling for speech recognition. In: Waibel, A. and Lee, K.-F., editors, *Readings in Speech Recognition,* pp. 450–506. Morgan Kaufmann.

Katz, S. M. (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans Acoustics, Speech & Signal Process.* **35**:400–401.

Krupka, G. R. (1995) SRA: Description of the SRA system as used for MUC-6. *Proceedings Sixth Message Understanding Conference (MUC-6)*. Pp. 221–235. Columbia, MD.

Krupka, G. R. and Hausman, K. (1998) IsoQuest Inc.: Description of the NetOwl™ extractor system as used for MUC-7. *Proceedings Sixth Message Understanding Conference (MUC-7)*. Fairfax, VA.

McDonald, D. (1996) Internal and external evidence in the identification and semantic categorization of proper names. In: Boguraev, B. and Pustejovsky, J., editors, *Corpus Processing for Lexical Acquisition*. pp. 21–39. MIT Press.

Miller, S., Crystal, M., Fox, H., Ramshaw, L., Schwartz, R., Stone, R., Weischedel, R. and the Annotation Group. (1998) BBN: Description of the SIFT system as used for MUC-7. *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Mikheev, A., Grover, C. and Moens, M. (1998) Description of the LTG system used for MUC-7. *Proceedings Sixth Message Understanding Conference (MUC-7)*. Fairfax, VA.

Mikheev, A., Moens, M. and Grover, C. (1999) Named entity recognition without gazeteers. *Proceedings Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL'1999)*. pp. 1–8. Bergen, Norway.

MUC6 (1995)*Proceedings Sixth Message Understanding Conference (MUC-6)*. Columbia, MD.

MUC7 (1998)*Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Rabiner, L. (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of IEEE **77**(2): 257–285.

Sekine, S. (1998) Description of the Japanese NE system used for MET-2. *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Tjong Kim Sang, E. F. and Buchholz, S. (2000) Introduction to the CoNLL-2000 shared task: chunking. *Proceedings Sixth Conference on Computational Natural Language Learning (CoNLL'2000)*. pp. 127–132. Lisbon, Portugal.

van Rijsbergen, C. J. (1979) *Information Retrieval* (2nd ed). Butterworth.

Viterbi, A. J. (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Infor. Theory*. **IT**(13): 260–269.

Watson, B. and Chunk Tsoi, A. (1992) Second-order Hidden Markov Models for speech recognition. *Proceeding 4th Australian International Conference on Speech Science and Technology*. pp. 146–151.

Yu, S., Bai, S. and Wu, P. (1998) Description of the Kent Ridge Digital Labs system used for MUC-7. *Proceedings Seventh Message Understanding Conference (MUC-7)*. Fairfax, VA.

Zhou, G. D., Su, J. and Tey, T. G. (2000) Hybrid text chunking. *Proceedings Sixth Conference on Computational Natural Language Learning (CoNLL'2000)*. pp. 163–166. Lisbon, Portugal.

Zhou, G. D. and Su, J. (2000) Error-driven HMM-based chunk tagger with context-dependent lexicon. *Proceedings 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC'2000)*, pp. 145–152. Hong Kong.

Zhou, G. D. and Su, J. (2002) Named entity recognition using an HMM-based chunk tagger. *Proceedings Fortieth Annual Meeting of the Association for Computational Linguistics (ACL'2002)*. pp. 473–480. Philadelphia.