Maximum Entropy Tagging with Binary and Real-Valued Features

Vanessa Sandrini **Marcello Federico** **Mauro Cettolo**
ITC-irst - Centro per la Ricerca Scientifica e Tecnologica
38050 Povo (Trento) - ITALY
{surname}@itc.it

Abstract

Recent literature on text-tagging reported successful results by applying Maximum Entropy (ME) models. In general, ME taggers rely on carefully selected binary features, which try to capture discriminant information from the training data. This paper introduces a standard setting of binary features, inspired by the literature on named-entity recognition and text chunking, and derives corresponding real-valued features based on smoothed log-probabilities. The resulting ME models have orders of magnitude fewer parameters. Effective use of training data to estimate features and parameters is achieved by integrating a leaving-one-out method into the standard ME training algorithm. Experimental results on two tagging tasks show statistically significant performance gains after augmenting standard binary-feature models with real-valued features.

1 Introduction

The Maximum Entropy (ME) statistical framework (Darroch and Ratcliff, 1972; Berger et al., 1996) has been successfully deployed in several NLP tasks. In recent evaluation campaigns, e.g. DARPA IE and CoNLL 2000-2003, ME models reached state-of-the-art performance on a range of text-tagging tasks.

With few exceptions, best ME taggers rely on carefully designed sets of features. Features correspond to binary functions, which model events, observed in the (annotated) training data and supposed to be meaningful or discriminative for the task at hand. Hence, ME models result in a log-linear combination of a large set of features, whose weights can be estimated by the well known Generalized Iterative Scaling (GIS) algorithm by Darroch and Ratcliff (1972).

Despite ME theory and its related training algorithm (Darroch and Ratcliff, 1972) do not set restrictions on the *range* of feature functions[1], popular NLP text books (Manning and Schutze, 1999) and research papers (Berger et al., 1996) seem to limit them to binary features. In fact, only recently, log-probability features have been deployed in ME models for statistical machine translation (Och and Ney, 2002).

This paper focuses on ME models for two text-tagging tasks: Named Entity Recognition (NER) and Text Chuncking (TC). By taking inspiration from the literature (Bender et al., 2003; Borthwick, 1999; Koeling, 2000), a set of *standard* binary features is introduced. Hence, for each feature type, a corresponding real-valued feature is developed in terms of smoothed probability distributions estimated on the training data. A direct comparison of ME models based on binary, real-valued, and mixed features is presented. Besides, performance on the tagging tasks, complexity and training time by each model are reported. ME estimation with real-valued features is accomplished by combining GIS with the leave-one-out method (Manning and Schutze, 1999).

Experiments were conducted on two publicly available benchmarks for which performance levels of many systems are published on the Web. Results show that better ME models for NER and TC can be developed by integrating binary and real-valued features.

[1] Darroch and Ratcliff (1972) show how any set of real-valued feature functions can be properly handled.

2 ME Models for Text Tagging

Given a sequence of words $w_1^T = w_1, \dots, w_T$ and a set of tags $\mathcal{C}$, the goal of text-tagging is to find a sequence of tags $c_1^T = c_1, \dots, c_T$ which maximizes the posterior probability, i.e.:

$$\hat{c_1^T} = \arg\max_{c_1^T} p(c_1^T \mid w_1^T). \quad (1)$$

By assuming a discriminative model, Eq. (1) can be rewritten as follows:

$$\hat{c_1^T} = \arg\max_{c_1^T} \prod_{t=1}^{T} p(c_t \mid c_1^{t-1}, w_1^T), \quad (2)$$

where $p(c_t | c_1^{t-1}, w_1^T)$ is the target conditional probability of tag c_t given the context (c_1^{t-1}, w_1^T), i.e. the entire sequence of words and the full sequence of previous tags. Typically, independence assumptions are introduced in order to reduce the context size. While this introduces some approximations in the probability distribution, it considerably reduces data sparseness in the sampling space. For this reason, the context is limited here to the two previous tags (c_{t-2}^{t-1}) and to four words around the current word (w_{t-2}^{t+2}). Moreover, limiting the context to the two previous tags permits to apply dynamic programming (Bender et al., 2003) to efficiently solve the maximization (2).

Let $y = c_t$ denote the class to be guessed ($y \in \mathcal{Y}$) at time t and $x = c_{t-2}^{t-1}, w_{t-2}^{t+2}$ its context ($x \in \mathcal{X}$). The generic ME model results:

$$p_\lambda(y \mid x) = \frac{\exp(\sum_{i=1}^n \lambda_i f_i(x, y))}{\sum_{y'} \exp(\sum_{i=1}^n \lambda_i f_i(x, y'))}. \quad (3)$$

The n feature functions $f_i(x, y)$ represent any kind of information about the event (x, y) which can be useful for the classification task. Typically, binary features are employed which model the verification of simple events within the target class and the context.

In Mikheev (1998), binary features for text tagging are classified into two broad classes: *atomic* and *complex*. Atomic features tell information about the current tag and one single item (word or tag) of the context. Complex features result as a combination of two or more atomic features. In this way, if the grouped events are not independent, complex features should capture higher correlations or dependencies, possibly useful to discriminate.

In the following, a standard set of binary features is presented, which is generally employed for text-tagging tasks. The reader familiar with the topic can directly check this set in Table 1.

3 Standard Binary Features

Binary features are indicator functions of specified events of the sample space $\mathcal{X} \times \mathcal{Y}$. Hence, they take value 1 if the event occurs or 0 otherwise. For the sake of notation, the feature name denotes the type of event, while the index specifies its parameters. For example:

$$\texttt{Orth}_{\texttt{person},\texttt{Cap},-1}(x, y)$$

corresponds to an *Orthographic* feature which is active if and only if the class at time t is `person` and the word at time $t-1$ in the context starts with capitalized letter.

3.1 Atomic Features

Lexical features These features model co-occurrences of classes and single words of the context. Lexical features are defined on a window of ±2 positions around the current word. Lexical features are denoted by the name `Lex` and indexed with the triple c, w, d which fixes the current class, i.e. $c_t = c$, the identity and offset of the word in the context, i.e. $w_{t+d} = w$. Formally, the feature is computed by:

$$\texttt{Lex}_{\ c,w,d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(w_{t+d} = w).$$

For example, the lexical feature for word `Verona`, at position t with tag `loc` (location) is:

$$\begin{aligned} \texttt{Lex}_{\texttt{loc},\texttt{Verona},0}(x, y) &= \delta(c_t = \texttt{loc}) \cdot \\ &\cdot \delta(w_t = \texttt{Verona}). \end{aligned}$$

Lexical features might introduce data sparseness in the model, given that in real texts an important fraction of words occur only once. In other words, many words in the test set will have no corresponding features-parameter pairs estimated on the training data. To cope with this problem, all words observed only once in the training data were mapped into the special symbol `oov`.

Syntactic features They model co-occurrences of the current class with part-of-speech or chunk tags of a specific position in the context. Syntactic features are denoted by the name `Syn` and indexed with a 4-tuple $(c, \texttt{Pos}, p, d)$ or $(c, \texttt{Chnk}, p, d)$,

Name	Index	Definition
Lex	c, w, d	$\delta(c_t = c) \cdot \delta(w_{t+d} = w), d \in Z$
Syn	c, T, p, d	$\delta(c_t = c) \cdot \delta(\texttt{T}(w_{t+d}) = p)\ , \texttt{T} \in \{\texttt{Pos}, \texttt{Chnk}\}, d \in Z$
Orth	c, F, d	$\delta(c_t = c) \cdot \texttt{F}(w_{t+d})\ , \texttt{F} \in \{\texttt{IsCap}, \texttt{IsCAP}\}, d \in Z$
Dict	c, L, d	$\delta(c_t = c) \cdot \texttt{InList}(L, w_{t+d}), d \in Z$
Tran	c, c', d	$\delta(c_t = c) \cdot \delta(c_{t-d} = c') \qquad d \in N^+$
Lex+	c, s, k, w_s^{s+k-1}	$\prod_{d=s}^{s+k-1} \texttt{Lex}_{c,w_d,d}(x, y), k \in N^+, s \in Z$
Syn+	c, T, s, k, p_s^{s+k-1}	$\prod_{d=s}^{s+k-1} \texttt{Syn}_{c,T,p_d,d}(x, y), k \in N^+, s \in Z$
Orth+	c, F, k, b_{-k}^{+k}	$\delta(c_t = c) \cdot \prod_{d=-k}^{k} \delta(\texttt{Orth}_{c,F,d}(x, y) = b_d)\ , b_d \in \{0, 1\}, k \in N^+$
Dict+	c, L, k, b_{-k}^{+k}	$\delta(c_t = c) \cdot \prod_{d=-k}^{k} \delta(\texttt{Dict}_{c,L,d}(x, y) = b_d)\ , b_d \in \{0, 1\}, k \in N^+$
Tran+	c, k, c_1^k	$\prod_{d=1'}^{k} \texttt{Tran}_{c,c_d,d}(x, y) \qquad k \in N^+$

Table 1: Standard set of binary features for text tagging.

which fixes the class c_t, the considered syntactic information, and the tag and offset within the context. Formally, these features are computed by:

$$\texttt{Syn}_{c,\texttt{Pos},p,d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(\texttt{Pos}(w_{t+d}) = p)$$

$$\texttt{Syn}_{c,\texttt{Chnk},p,d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(\texttt{Chnk}(w_{t+d}) = p).$$

Orthographic features These features model co-occurrences of the current class with surface characteristics of words of the context, e.g. check if a specific word in the context starts with capitalized letter (IsCap) or is fully capitalized (IsCAP). In this framework, only capitalization information is considered. Analogously to syntactic features, orthographic features are defined as follows:

$$\texttt{Orth}_{c,\texttt{IsCap},d}(x, y) \hat{=} \delta(c_t = c) \cdot \texttt{IsCap}(w_{t+d})$$

$$\texttt{Orth}_{c,\texttt{IsCAP},d}(x, y) \hat{=} \delta(c_t = c) \cdot \texttt{IsCAP}(w_{t+d}).$$

Dictionary features These features check if specific positions in the context contain words occurring in some prepared list. This type of feature results relevant for tasks such as NER, in which *gazetteers* of proper names can be used to improve coverage of the training data. Atomic dictionary features are defined as follows:

$$\texttt{Dict}_{c,L,d}(x, y) \hat{=} \delta(c_t = c) \cdot \texttt{InList}(L, w_{t+d})$$

where L is a specific pre-compiled list, and InList is a function which returns 1 if the specified word matches one of the multi-word entries of list L, and 0 otherwise.

Transition features Transition features model Markov dependencies between the current tag and a previous tag. They are defined as follows:

$$\texttt{Tran}_{c,c',d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(c_{t-d} = c').$$

3.2 Complex Features

More complex events are defined by combining two or more atomic features in one of two ways. $Product$ features take the intersection of the corresponding atomic events. $Vector$ features consider all possible outcomes of the component features.

For instance, the product of 3 atomic Lexical features, with class c, offsets $-2, -1, 0$, and words v_{-2}, v_{-1}, v_0, is:

$$\texttt{Lex+}_{c,-2,3,v_{-2},v_{-1},v_0}(x, y) \hat{=} \prod_{d=-2}^{0} \texttt{Lex}_{c,v_d,d}(x, y).$$

Vector features obtained from three Dictionary features with the same class c, list L, and offsets, respectively, -1,0,+1, are indexed over all possible binary outcomes b_{-1}, b_0, b_1 of the single atomic features, i.e.:

$$\texttt{Dict+}_{c,L,1,b_{-1},b_0,b_{+1}}(x, y) \hat{=} \delta(c_t = c) \times \prod_{d=-1}^{1} \delta(\texttt{Dict}_{c,L,d}(x, y) = b_d).$$

Complex features used in the experiments are described in Table 1.

The use of complex features significantly increases the model complexity. Assuming that there are $10,000$ words occurring more than once in the training corpus, the above lexical feature potentially adds $O(|\mathcal{C}|10^{12})$ parameters!

As complex binary features might result prohibitive from a computational point of view, real-valued features should be considered as an alternative.

Feature	Index	Probability Distribution
Lex	d	$p(c_t \mid w_{t+d})$
Syn	T, d	$p(c_t \mid \mathtt{T}(w_{t+d}))$
Orth	F, d	$p(c_t \mid \mathtt{F}(w_{t+d}))$
Dict	$List, d$	$p(c_t \mid \mathtt{IsIn}(List, w_{t+d}))$
Tran	d	$p(c_t \mid c_{t-d})$
Lex+	s, k	$p(c_t \mid w_{t+s}, .., w_{t+s+k-1}$
Syn+	T, s, k	$p(c_t \mid \mathtt{T}(w_{t+s}, \ldots, w_{t+s+k-1}))$
Orth+	k, F	$p(c_t \mid \mathtt{F}(w_{t-k}), \ldots, \mathtt{F}(w_{t+k}))$
Dict+	k, L	$p(c_t \mid \mathtt{InList}(\mathtt{L}, \mathtt{w_{t-k}}), \ldots, \mathtt{InList}(\mathtt{L}, \mathtt{w_{t+k}}))$
Tran+	k	$p(c_t \mid c_{t-k}, \ldots, c_{t+k}))$

Table 2: Corresponding standard set of real-values features.

4 Real-valued Features

A binary feature can be seen as a probability measure with support set made of a single event. According to this point of view, we might easily extend binary features to probability measures defined over larger event spaces. In fact, it results convenient to introduce features which are logarithms of conditional probabilities. It can be shown that in this way linear constraints of the ME model can be interpreted in terms of Kullback-Leibler distances between the target model and the conditional distributions (Klakow, 1998).

Let $p_1(y|x), p_2(y|x), \ldots, p_n(y|x)$ be n different conditional probability distributions estimated on the training corpus. In our framework, each conditional probability p_i is associated to a feature f_i which is defined over a subspace $[\mathcal{X}]_i \times \mathcal{Y}$ of the sample space $\mathcal{X} \times \mathcal{Y}$. Hence, $p_i(y|x)$ should be read as a shorthand of $p(y \mid [x]_i)$.

The corresponding real-valued feature is:

$$f_i(x, y) = \log p_i(y \mid x). \quad (4)$$

In this way, the ME in Eq. (3) can be rewritten as:

$$p_\lambda(y|x) = \frac{\prod_i^n p_i(y|x)^{\lambda_i}}{\sum_{y'} \prod_i p_i(y'|x)^{\lambda_i}}. \quad (5)$$

According to the formalism adopted in Eq. (4), real-valued features assume the following form:

$$f_i(c_t, c_{t-2}^{t-1}, w_{t-2}^{t+2}) = \log p_i(c_t \mid c_{t-2}^{t-1}, w_{t-2}^{t+2}). \quad (6)$$

For each so far presented type of binary feature, a corresponding real-valued type can be easily defined. The complete list is shown in Table 2. In general, the context subspace was defined on the basis of the offset parameters of each binary feature. For instance, all lexical features selecting two words at distances -1 and 0 from the current position t are modeled by the conditional distribution $p(c_t \mid w_{t-1}, w_t)$. While distributions of lexical, syntactic and transition features are conditioned on words or tags, dictionary and orthographic features are conditioned on binary variables.

An additional real-valued feature that was employed is the so called *prior feature*, i.e. the probability of a tag to occur:

$$\mathtt{Prior}(x, y) = \log p(c_t)$$

A major effect of using real-valued features is the drastic reduction of model parameters. For example, each complex lexical features discussed before introduce just one parameter. Hence, the small number of parameters eliminates the need of smoothing the ME estimates.

Real-valued features present some drawbacks. Their level of granularity, or discrimination, might result much lower than their binary variants. For many features, it might result difficult to compute reliable probability values due to data sparseness. For the last issue, smoothing techniques developed for statistical language models can be applied (Manning and Schutze, 1999).

5 Mixed Feature Models

This work, beyond investigating the use of real-valued features, addresses the behavior of models combining binary and real-valued features. The reason is twofold: on one hand, real-valued features allow to capture complex information with fewer parameters; on the other hand, binary features permit to keep a good level of granularity over salient characteristics. Hence, finding a compromise between binary and real-valued features might help to develop ME models which better trade-off complexity vs. granularity of information.

6 Parameter Estimation

From the duality of ME and maximum likelihood (Berger et al., 1996), optimal parameters λ_* for model (3) can be found by maximizing the log-likelihood function over a training sample $\{(x_t, y_t) : t = 1, \ldots, N\}$, i.e.:

$$\lambda_* = \arg\max_{\lambda} \sum_{t=1}^{N} \log p_\lambda(y_t|x_t). \qquad (7)$$

Now, whereas binary features take only two values and do not need any estimation phase, conditional probability features have to be estimated on some data sample. The question arises about how to efficiently use the available training data in order to estimate the parameters and the feature distributions of the model, by avoiding over-fitting.

Two alternative techniques, borrowed from statistical language modeling, have been considered: the *Held-out* and the *Leave-one-out* methods (Manning and Schutze, 1999).

Held-out method. The training sample $\mathcal{S}$ is split into two parts used, respectively, to estimate the feature distributions and the ME parameters.

Leave-one-out. ME parameters and feature distributions are estimated over the same sample $\mathcal{S}$. The idea is that for each addend in eq. (7), the corresponding sample point (x_t, y_t) is removed from the training data used to estimate the feature distributions of the model. In this way, it can be shown that occurrences of novel observations are simulated during the estimation of the ME parameters (Federico and Bertoldi, 2004).

In our experiments, language modeling smoothing techniques (Manning and Schutze, 1999) were applied to estimate feature distributions $p_i(y|x)$. In particular, smoothing was based on the discounting method in Ney et al. (1994) combined to interpolation with distributions using less context. Given the small number of smoothing parameters involved, leave-one-out probabilities were approximated by just modifying count statistics on the fly (Federico and Bertoldi, 2004). The rationale is that smoothing parameters do not change significantly after removing just one sample point.

For parameter estimation, the GIS algorithm by Darroch and Ratcliff (1972) was applied. It is known that the GIS algorithm requires feature functions $f_i(x, y)$ to be non-negative. Hence, features were re-scaled as follows:

$$f_i(x, y) = \log p_i(y|x) + log\frac{1+\epsilon}{\min p_i}, \qquad (8)$$

where ϵ is a small positive constant and the denominator is a constant term defined by:

$$\min p_i = \min_{(x,y)\in\mathcal{S}} p_i(y|x). \qquad (9)$$

The factor $(1 + \epsilon)$ was introduced to ensure that real-valued features are always positive. This condition is important to let features reflect the same behavior of the conditional distributions, which assign a positive probability to each event.

It is easy to verify that this scaling operation does not affect the original model but only impacts on the GIS calculations. Finally, a *slack* feature was introduced by the algorithm to satisfy the constraint that all features sum up to a constant value (Darroch and Ratcliff, 1972).

7 Experiments

This section presents results of ME models applied to two text-tagging tasks, Named Entity Recognition (NER) and Text Chunking (TC).

After a short introduction to the experimental framework, the detailed feature setting is presented. Then, experimental results are presented for the following contrastive conditions: binary versus real-valued features, training via held-out versus leave-one-out, atomic versus complex features.

7.1 Experimental Set-up

Named Entity Recognition English NER experiments were carried out on the CoNLL-2003 shared task[2]. This benchmark is based on texts from the Reuters Corpus which were manually annotated with parts-of-speech, chunk tags, and named entity categories. Four types of categories are defined: person, organization, location and miscellaneous, to include e.g. nations, artifacts, etc. A filler class is used for the remaining words. After including tags denoting the start of multiword entities, a total of 9 tags results. Data are partitioned into training (200K words), development (50K words), and test (46K words) samples.

[2] Data and results in http://cnts.uia.ac.be/conll2003/ner.

Text Chunking English TC experiments were conducted on the CoNLL-2000 shared task[3]. Texts originate from the Wall Street Journal and are annotated with part-of-speech tags and chunks. The chunk set consists of 11 syntactic classes. The set of tags which also includes start-markers consists of 23 classes. Data is split into training (210K words) and test (47K words) samples.

Evaluation Tagging performance of both tasks is expressed in terms of F-score, namely the harmonic mean of precision and recall. Differences in performance have been statistically assessed with respect to precision and recall, separately, by applying a standard test on proportions, with significance levels $\alpha = 0.05$ and $\alpha = 0.1$. Henceforth, claimed differences in precision or recall will have their corresponding significance level shown in parenthesis.

7.2 Settings and Baseline Models

Feature selection and setting for ME models is an art. In these experiments we tried to use the same set of features with minor modifications across both tasks. In particular, used features and their settings are shown in Table 3.

Training of models with GIS and estimation of feature distributions used in-house developed toolkits. Performance of binary feature models was improved by smoothing features with Gaussian priors (Chen and Rosenfeld, 1999) with mean zero and standard deviation $\sigma = 4$. In general, tuning of models was carried out on a development set.

Most of the comparative experiments were performed on the NER task. Three baseline models using atomic features `Lex`, `Syn`, and `Tran` were investigated first: model BaseBin, with all binary features; model BaseReal, with all real-valued features plus the prior feature; model BaseMix, with real-valued `Lex` and binary `Tran` and `Syn`. Models BaseReal and BaseMix were trained with the held-out method. In particular, feature distributions were estimated on the training data while ME parameters on the development set.

7.3 Binary vs. Real-valued Features

The first experiment compares performance of the baseline models on the NER task. Experimental results are summarized in Table 4. Models BaseBin, BaseReal, and BaseMix achieved F-scores of 77.22, 76.84, and 77.34. Statistically meaningful differences were in terms of recall, between BaseBin and BaseReal ($\alpha = 0.1$), and between BaseMix and BaseReal ($\alpha = 0.05$).

Model ID	Num	P%	R%	F-score
BaseBin	580K	78.82	75.62	77.22
BaseReal	10	79.74	74.15	76.84
BaseMix	753	78.90	75.85	77.34

Table 4: Performance of baseline models on the NER task. Number of parameters, precision, recall, and F-score are reported for each model.

Model	Methods	P%	R%	F-score
BaseMix	Held-Out	78.90	75.85	77.34
BaseMix	L-O-O	80.64	76.40	78.46

Table 5: Performance of mixed feature models with two different training methods.

Despite models BaseMix and BaseBin perform comparably, the former has many fewer parameters, i.e. 753 against 580,000. In fact, BaseMix requires storing and estimating feature distributions, which is however performed at a marginal computational cost and off-line with respect to GIS training.

7.4 Training with Mixed Features

An experiment was conducted with the BaseMix model to compare the held-out and leave-one-out training methods. Results in terms of F-score are reported in Table 5. By applying the leave-one-out method F-score grows from 77.34 to 78.46, with a meaningful improvement in recall ($\alpha = 0.05$). With respect to models BaseBin and BaseReal, leave-one-out estimation significantly improved precision ($\alpha = 0.05$).

In terms of training time, ME models with real-valued features took significantly more GIS iterations to converge. Figures of cost per iteration and number of iterations are reported in Table 6. (Computation times are measured on a single CPU Pentium-4 2.8GHz.) Memory size of the training process is instead proportional to the number n of parameters.

7.5 Complex Features

A final set of experiments aims at comparing the baseline ME models augmented with complex features, again either binary only (model FinBin),

[3] Data and results in http://cnts.uia.ac.be/conll2000/chunking.

Feature	Index	NE Task	Chunking Task
Lex	c, w, d	$N(w) > 1, -2 \leq d \leq +2$	$-2 \leq d \leq +2$
Syn	c, T, p, d	$T \in \{\texttt{Pos}, \texttt{Chnk}\}, d = 0$	$T = \texttt{Pos}, -2 \leq d \leq +2$
Tran	c, c', d	$d = -2, -1$	$d = -2, -1$
Lex+	c, s, k, w_s^{s+k-1}	$s = -1, 0, k = 1$	$s = -1, 0\ k = 1$
Syn+	c, T, s, k, p_s^{s+k-1}	not used	$s = -1, 0\ \ k = 1$
Orth+	c, k, F, b_{-k}^{+k}	$F = \{\texttt{Cap}, \texttt{CAP}\},\ k = 2$	$F = \texttt{Cap},\ \ k = 1$
Dict+	c, k, L, b_{-k}^{+k}	$k = 3L = \{\texttt{LOC}, \texttt{PER}, \texttt{ORG}, \texttt{MISC}\}$	not used
Tran+	c, k, c_1^k	$k = 2$	$k = 2$

Table 3: Setting used for binary and real-valued features in the reported experiments.

Model	Single Iteration	Iterations	Total
BaseBin	54 sec	750	≈ 11 h
BaseReal	9.6 sec	35,000	≈ 93 h
BaseMix	42 sec	4,000	≈ 46 h

Table 6: Computational cost of parameter estimation by different baseline models.

Model	Num	P%	R%	F-score
FinBin	673K	81.92	80.36	81.13
FinReal	19	83.58	74.03	78.07
FinMix	3K	84.34	80.38	82.31

Table 7: Results with complex features on the NER task.

Model	Num	P%	R%	F-score
FinBin	2M	91.04	91.48	91.26
FinReal	19	88.73	90.58	89.65
FinMix	6K	91.93	92.24	92.08

Table 8: Results with complex features on the TC task.

real-valued only (FinReal), or mixed (FinMix). Results are provided both for NER and TC.

This time, compared models use different feature settings. In fact, while previous experiments aimed at comparing the same features, in either real or binary form, these experiments explore alternatives to a full-fledged binary model. In particular, real-valued features are employed whose binary versions would introduce a prohibitively large number of parameters. Parameter estimation of models including real-valued features always applies the leave-one-out method.

For the NER task, model FinBin adds Orth+ and Dict+; FinReal adds Lex+, Orth+ and Dict+; and, FinMix adds real-valued Lex+ and binary-valued Orth+ and Dict+.

In the TC task, feature configurations are as follows: FinBin uses Lex, Syn, Tran, and Orth+; FinReal uses Lex, Syn, Tran, Prior, Orth+, Lex+, Syn+, Tran+; and, finally, FinMix uses binary Syn, Tran, Orth+ and real-valued Lex, Lex+, Syn+.

Performance of the models on the two tasks are reported in Table 7 and Table 8, respectively.

In the NER task, all final models outperform the baseline model. Improvements in precision and recall are all significant ($\alpha = 0.05$). Model FinMix improves precision with respect to model FinBin ($\alpha = 0.05$) and requires two order of magnitude fewer parameters.

In the TC task, the same trend is observed. Again, best performance is achieved by the model combining binary and real-valued features. In particular, all observable differences in terms of precision and recall are significant ($\alpha = 0.05$).

8 Discussion

In summary, this paper addressed improvements to ME models for text tagging applications. In particular, we showed how standard binary features from the literature can be mapped into corresponding log-probability distributions. ME training with the so-obtained real-valued features can be accomplished by combining the GIS algorithm with the leave-one-out or held-out methods.

With respect to the best performing systems at the CoNLL shared tasks, our models exploit a relatively smaller set of features and perform significantly worse. Nevertheless, performance achieved by our system are comparable with those reported by other ME-based systems taking part in the evaluations.

Extensive experiments on named-entity recognition and text chunking have provided support to the following claims:

- The introduction of real-valued features drastically reduces the number of parameters of the ME model with a small loss in performance.
- The leave-one-out method is significantly more effective than the held-out method for training ME models including real-valued features.
- The combination of binary and real-valued features can lead to better ME models. In particular, state-of-the-art ME models with binary features are significantly improved by adding complex real-valued features which model long-span lexical dependencies.

Finally, the GIS training algorithm does not seem to be the optimal choice for ME models including real-valued features. Future work will investigate variants of and alternatives to the GIS algorithm. Preliminary experiments on the Base-Real model showed that training with the Simplex algorithm (Press et al., 1988) converges to similar parameter settings 50 times faster than the GIS algorithm.

9 Acknowledgments

This work was partially financed by the European Commission under the project FAME (IST-2000-29323), and by the Autonomous Province of Trento under the the FU-PAT project WebFaq.

References

O. Bender, F. J. Och, and H. Ney. 2003. Maximum entropy models for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 148–151. Edmonton, Canada.

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–72.

A. Borthwick. 1999. *A Maximum Entropy approach to Named Entity Recognition*. Ph.D. thesis, Computer Science Department - New York University, New York, USA.

S. Chen and R. Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University.

J.N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Liner models. *Annals of Mathematical Statistics*, 43:1470–1480.

M. Federico and N. Bertoldi. 2004. Broadcast news lm adaptation over time. *Computer Speech and Language*, 18(4):417–435, October.

D. Klakow. 1998. Log-linear interpolation of language models. In *Proceedings of the International Conference of Spoken Language P rocessing (ICSLP)*, Sidney, Australia.

R. Koeling. 2000. Chunking with maximum entropy models. In *Proceedings of CoNLL-2000*, pages 139–141, Lisbon, Portugal.

C. D. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

A. Mikheev. 1998. Feature lattices for maximum entropy modelling. In *COLING-ACL*, pages 848–854.

H. Ney, U. Essen, and R. Kneser. 1994. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8(1):1–38.

F.J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machin e translation. In *ACL02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, PA, Philadelphia.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1988. *Numerical Recipes in C*. Cambridge University Press, New York, NY.