A Systematic Comparison of Feature-Rich Probabilistic Classifiers for NER Tasks

Benjamin Rosenfeld, Moshe Fresko, and Ronen Feldman

Bar-Ilan University, Computer Science Department, Data-Mining Lab.,
Ramat-Gan, Israel
{freskom1, feldman}@cs.biu.ac.il
http://www.cs.biu.ac.il/~freskom1/DataMiningLab

Abstract. In the CoNLL 2003 NER shared task, more than two thirds of the submitted systems used the feature-rich representation of the task. Most of them used maximum entropy to combine the features together. Others used linear classifiers, such as SVM and RRM. Among all systems presented there, one of the MEMM-based classifiers took the second place, losing only to a committee of four different classifiers, one of which was ME-based and another RRM-based. The lone RRM was fourth, and CRF came in the middle of the pack. In this paper we shall demonstrate, by running the three algorithms upon the same tasks under exactly the same conditions that this ranking is due to feature selection and other causes and not due to the inherent qualities of the algorithms, which should be ranked otherwise.

1 Introduction

Recently, feature-rich probabilistic conditional classifiers became state-of-the-art in sequence labeling tasks, such as NP chunking, PoS tagging, and Named Entity Recognition. Such classifiers build a probabilistic model of the task, which defines a conditional probability on the space of all possible labelings of a given sequence. In this, such classifiers differ from the binary classifiers, such as decision trees and rule-based systems, which directly produce classification decisions, and from the generative probabilistic classifiers, such as HMM-based Nymble [2] and SCFG-based TEG [8], which model the joint probability of sequences and their labelings. Modeling the conditional probability allows the classifiers to have all the benefits of probabilistic systems while having the ability to use any property of tokens and their contexts, if the property can be represented in the form of binary features. Since almost all local properties can be represented in such a way, this ability is very powerful.

There are several different feature-rich probabilistic classifiers developed by different researchers, and in order to compare them, one usually takes a known publicly available dataset, such as MUC-7 [23] or CoNLL shared task [12], and compares the performance of the algorithms on the dataset. However, performance of a feature-rich classifier strongly depends upon the feature sets it uses. Since systems developed by different researches are bound to use different feature sets, the differences in performance of complete systems can not reliably teach us about the qualities of the underlying algorithms.

A. Jorge et al. (Eds.): PKDD 2005, LNAI 3721, pp. 217 – 227, 2005.

In this work we compare the performances of three common models (all present in the CoNLL 2003 shared task) – MEMM [15], CRF [16], and RRM (regularized Winnow) [14] – within the same platform, using exactly the same set of features. We also test the effects of different training sizes, different choice of parameters, and different feature sets upon the algorithms' performance.

Our experiments indicate that CRF outperforms MEMM for all datasets and feature sets, which is not surprising, since CRF is a better model of sequence labeling. Surprisingly, though, the RRM performs at the same level or even better than CRF, despite being local model like MEMM, and being significantly simpler to build than both CRF and MEMM.

The following section of the paper we outline the three algorithms. We then present our experiments and their results.

2 Classifiers

The general sequence labeling problem can be described as follows. Given a small set Y of labels, and a sequence $\boldsymbol{x} = \boldsymbol{x}_1\boldsymbol{x}_2\ldots\boldsymbol{x}_{l(\boldsymbol{x})}$, the task is to find a labeling $\boldsymbol{y} = y_1y_2\ldots y_{l(\boldsymbol{x})}$, where each label $y_i \in Y$. In the framework of feature-rich classifiers, the elements $\boldsymbol{x}_i$ of the sequence should not be thought of as simple tokens, but rather as sequence positions, or *contexts*. The contexts are characterized by a set of externally supplied binary *features*. Thus, each context $\boldsymbol{x}_i$ can be represented as a vector $\boldsymbol{x}_i=(x_{i1},x_{i2},\ldots,x_{ik})$, where $x_{ij} = 1$ if the j-th feature is present in the i-th context, and $x_{ij} = 0$ otherwise.

The feature-rich sequence classifiers have no knowledge of the nature of features and labels. Instead, in order to make predictions, the classifiers are supplied with a training set $T = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1..n}$ of sequences with their intended labelings. The classifiers use the training set to build the model of the task, which is subsequently used to label unseen sequences.

We shall describe the particular algorithms only briefly, referring to the original works to supply the details.

2.1 MEMM

A Maximum Entropy Markov Model classifier [4] builds a probabilistic conditional model of sequence labeling. Labeling each position in each sequence is considered to be a separate classification decision, possibly influenced by a small constant number of previous decisions in the same sequence. In our experiments we use a Markov model of order one, in which only the most recent previous decision is taken into account.

Maximal Entropy models are formulated in terms of *feature functions* $f(\boldsymbol{x}_i, y_i, y_{i-1}) \to \{0, 1\}$, which link together the context features and the target labels. In our formulation, we have a feature function f_{jy} for each context feature j and each label y, and a feature function $f_{iyy'}$ for each context feature and each pair of labels. The functions are defined as follows:

$f_{jy}(\boldsymbol{x}_i, y_i, y_{i-1}) = \boldsymbol{x}_{ij}I_y(y_i)$ and $f_{jyy'}(\boldsymbol{x}_i, y_i, y_{i-1}) = \boldsymbol{x}_{ij}I_y(y_i)I_{y'}(y_{i-1})$, where $I_a(b)$ is one if $a = b$ and zero otherwise. The vector of all feature functions is denoted $\boldsymbol{f}(\boldsymbol{x}_i, y_i, y_{i-1})$.

A trained MEMM model has a real weight λ_f for each feature function f. Together, the weights form the parameter vector $\boldsymbol{\lambda}$. The model has the form

$$(1) \quad P_{\boldsymbol{\lambda}}(y_i \mid \boldsymbol{x}_i, y_{i-1}) = \frac{1}{Z(\boldsymbol{x}_i, y_{i-1})} \exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}_i, y_i, y_{i-1})\right),$$

where $Z(x_i, y_{i-1}) = \sum_y \exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}_i, y, y_{i-1})\right)$ is the factor making the probabilities for different labels sum to one.

Given a model (1), it can be used for inferring the labeling $\boldsymbol{y} = y_1 y_2 \ldots y_{l(\boldsymbol{x})}$ of an unseen sequence $\boldsymbol{x} = \boldsymbol{x}_1 \boldsymbol{x}_2 \ldots \boldsymbol{x}_{l(\boldsymbol{x})}$ by calculating the most probable overall sequence of labels:

$$(2) \quad \boldsymbol{y}(\boldsymbol{x}) := \arg\max_{y_1 y_2 \ldots y_{l(x)}} \sum_{i=1}^{l(x)} \log P_{\boldsymbol{\lambda}}(y_i \mid \boldsymbol{x}_i, y_{i-1}).$$

This most probable sequence can be efficiently calculated using a variant of the Viterbi algorithm.

The model parameters are trained in such a way as to maximize the model's entropy while making the expected value of each feature function agree with the observed relative frequency of the feature function in the training data. Those conditions can be shown to be uniquely satisfied by the model which maximizes the log-likelihood of the training data among all models of the form (1). In order to avoid overfitting, the likelihood can be penalized with a prior $\Pr(\boldsymbol{\lambda})$. Then, the log-likelihood is

$$L_T(\boldsymbol{\lambda}) = \sum_t \sum_{i=1}^{l(x^{(t)})} \log P_{\boldsymbol{\lambda}}(y_i^{(t)} \mid x_i^{(t)}, y_{i-1}^{(t)}) - \Pr(\boldsymbol{\lambda}) =$$

$$= \sum_t \sum_{i=1}^{l(x^{(t)})} \left(\boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}_i^{(t)}, y_i^{(t)}, y_{i-1}^{(t)}) - \log Z(x_i^{(t)})\right) - \Pr(\boldsymbol{\lambda})$$

and its gradient is

$$\nabla L_T(\boldsymbol{\lambda}) = \sum_t \sum_{i=1}^{l(x^{(t)})} \left(\boldsymbol{f}(\boldsymbol{x}_i^{(t)}, y_i^{(t)}, y_{i-1}^{(t)}) - E_{P\boldsymbol{\lambda}}(\boldsymbol{f}(\boldsymbol{x}_i^{(t)}, Y, y_{i-1}^{(t)}))\right) - \nabla \Pr(\boldsymbol{\lambda}),$$

where

$$E_{P\boldsymbol{\lambda}}(\boldsymbol{f}(\boldsymbol{x}_i^{(t)}, Y, y_{i-1}^{(t)})) = \sum_{y \in Y} P_{\boldsymbol{\lambda}}(y \mid \boldsymbol{x}_i^{(t)}, y_{i-1}^{(t)})\ \boldsymbol{f}(\boldsymbol{x}_i^{(t)}, y, y_{i-1}^{(t)})$$

is the expectation of the feature vector under the model (1).

With a reasonably chosen prior, the function $L_T(\boldsymbol{\lambda})$ is strictly concave, and so can be maximized by any convex optimization algorithm. We use L-BFGS for this purpose.

2.2 CRF

A Conditional Random Fields (CRF) [7] classifier also builds a probabilistic model of sequence labeling. CRF uses the maximal entropy principle to model the labeling of a sequence as a whole, in contrast to MEMM, which builds a model of separate labeling decisions at different sequence positions.

The model is built upon exactly the same vector $\boldsymbol{f}(\boldsymbol{x}_i, y_i, y_{i-1})$ of feature functions as MEMM. The feature functions are summed along a sequence to produce a *sequence feature functions vector*

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{l(x)} \boldsymbol{f}(\boldsymbol{x}_i, y_i, y_{i-1}), \tag{3}$$

which is then used for constructing the maximal entropy model

$$P_{\boldsymbol{\lambda}}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})\right).$$

A trained model can be used for inferring the most probable labeling of an unseen sequence. The decomposition (3) allows to use the Vitterbi algorithm almost identically to the MEMM case, except that in (2), instead of $\log P_{\boldsymbol{\lambda}}(y_i \mid \boldsymbol{x}_i, y_{i-1}) = \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}_i, y_i, y_{i-1}) - \log Z(\boldsymbol{x}_i, y_{i-1})$, simple $\boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}_i, y_i, y_{i-1})$ is used. Since $Z(\boldsymbol{x})$ does not depend on labeling, it need not be calculated at all during inference.

To train the CRF model, we need to maximize the model entropy while satisfying the expectation constrains, expressed this time in terms of the sequence feature functions. As before, this is equivalent to maximizing the log-likelihood of the training data, which can also be penalized with a prior to avoid overfitting:

$$L_T(\boldsymbol{\lambda}) = \sum_t \log P_{\boldsymbol{\lambda}}(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(t)}) - \frac{\|\boldsymbol{\lambda}\|^2}{2\sigma^2} = \sum_t \left(\boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) - \log Z(\boldsymbol{x}^{(t)})\right) - \Pr(\boldsymbol{\lambda}).$$

The gradient is

$$\nabla L_T(\boldsymbol{\lambda}) = \sum_t \left(\boldsymbol{F}(\boldsymbol{x}^{(t)}, y^{(t)}) - E_{P\boldsymbol{\lambda}}(\boldsymbol{F}(\boldsymbol{x}^{(t)}, \boldsymbol{Y}^{(t)}))\right) - \nabla \Pr(\boldsymbol{\lambda}),$$

where $\boldsymbol{Y}^{(t)}$ is the set of label sequences of length $l(x^{(t)})$, and $E_{P\boldsymbol{\lambda}}(\boldsymbol{F}(\boldsymbol{x}^{(t)}, \boldsymbol{Y})) = \sum_{y \in Y} P_{\boldsymbol{\lambda}}(\boldsymbol{y} \mid \boldsymbol{x}^{(t)}) \boldsymbol{F}(\boldsymbol{x}^{(t)}, y)$ is the expectation of the sequence feature functions vector under the model (3).

In order to maximize $L_T(\boldsymbol{\lambda})$, we need a way to calculate $\log Z(\boldsymbol{x})$ and $E_{P\boldsymbol{\lambda}}(\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{Y}))$ for the given sequence $\boldsymbol{x}$. It is possible to do this efficiently, using a variant of the Forward-Backward algorithm. Details can be found in [7] and [19].

2.3 RRM

The Robust Risk Minimization classifier [14] results from regularization of the Winnow algorithm [21]. Winnow is a multiplicative-update online algorithm used for estimating the weights of a binary linear classifier, which has the following general form:

$$y = \text{sign}(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}),$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ is the weight vector, and $y \in \{+1, -1\}$ is the classification decision.

It was shown in [20], that using a risk function of a special form, the regularized Winnow can produce such weights $\boldsymbol{w}$ that

$$P(y = +1 \mid \boldsymbol{x}) \approx (Tr_{[-1,1]}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}) + 1) / 2,$$

where $Tr_{[a,b]}(s) = \min(b, \max(a, s))$ is a truncation of s onto $[a, b]$.

Although the derivation is elaborate, the resulting algorithm is very simple. It consists of iteratively going over the training set $T = \{(\boldsymbol{x}^{(t)}, y^{(t)})\}_{t=1..n}$ (here, $y^{(t)} = \pm 1$), and incrementally updating

$$\begin{aligned} \alpha_t &:= Tr_{[0,2c]}\left(\alpha_t + \eta\left(1 - \frac{\alpha_t}{c} - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}^{(t)} y^{(t)}\right)\right) \\ w_j &:= \mu_j \exp\left(\sum_t \alpha_t x_j^{(t)} y^{(t)}\right) \end{aligned} \tag{4}$$

The α_t are the *dual* weights, initialized to zero and kept between the iterations. c is the *regularization* parameter, η is the *learning rate*, and μ_j is the *prior*.

The $y^{(t)}$ in (4) are binary decisions. In order to use the RRM for sequence labeling task with more than two labels, we can build a separate classifier for each label and then combine them together within a single Vitterbi search.

3 Experimental Setup

The goal of this work is to compare the three sequence labeling algorithms in several different dimensions: absolute performance, dependence upon the corpus, dependence upon the training set size and the feature set, and dependence upon the hyperparameters.

3.1 Datasets

For our experiments we used four datasets: CoNLL-E, the English CoNLL 2003 shared task dataset, CoNLL-D, the German CoNLL 2003 shared task dataset, the MUC-7 dataset [23], and the proprietary CLF dataset [8]. For the experiments with smaller training sizes, we cut training corpora into chunks of 10K, 20K, 40K, 80K, and 160K tokens. The corresponding datasets are denoted <Corpus>_<Size>, e.g. "CoNLL-E_10K".

3.2 Feature Sets

There are many properties of tokens and their contexts that could be used in a NER system. We experiment with the following properties, ordered according to the difficulty of obtaining them:

A. The exact character strings of tokens in a small window around the given position.
B. Lowercase character strings of tokens.
C. Simple properties of characters inside tokens, such as capitalization, letters vs digits, punctuation, etc.
D. Suffixes and prefixes of tokens with length 2 to 4 characters.

E. Presence of tokens in local and global dictionaries, which contain words that were classified as certain entities someplace before – either anywhere (for global dictionaries), or in the current document (for local dictionaries).
F. PoS tags of tokens.
G. Stems of tokens.
H. Presence of tokens in small manually prepared lists of semantic terms – such as months, days of the week, geographical features, company suffixes, etc.
I. Presence of tokens inside gazetteers, which are huge lists of known entities.

The PoS tags are available only for the two CoNLL datasets, and the stems are available only for the CoNLL-D dataset. Both are automatically generated and thus contain many errors.

The gazetteers and lists of semantic terms are available for all datasets except CoNLL-D.

We tested the following feature sets:
set0: checks properties A, B, C at the current and the previous token.
set1: A, B, C, B+C in a window [-2…0].
set2: A, B, C, B+C in a window [-2…+2].
set2x: Same as set2, but only properties appearing > 3 times are used.
set3: A, B, C, B+C in a window [-2…+2], D at the current token.
set4: A, B, C, B+C in a window [-2…+2], D at the current token, E.
set5: A, B, C, B+C, F, G in a window [-2…+2] , D at the current token, E.
set6: set4 or set5, H
set7: set4 or set5, H, I

3.3 Hyperparameters

The MaxEntropy-based algorithms, MEMM and CRF, have similar hyperparameters, which define the priors for training the models. We experimented with two different priors – Laplacian (double exponential) $\Pr_{LAP}(\boldsymbol{\lambda}) = \alpha\Sigma_i|\lambda_i|$ and Gaussian $\Pr_{GAU}(\boldsymbol{\lambda}) = (\Sigma_i\lambda_i^2) / (2\sigma^2)$. Each prior depends upon a single hyperparameter specifying the "strength" of the prior. Note, that $\nabla\Pr_{LAP}(\boldsymbol{\lambda})$ has discontinuities at zeroes of λ_i. Because of that, a special consideration must be given to the cases when λ_i approaches or is at zero. Namely,

(1) if λ_i tries to change sign, set $\lambda_i := 0$, and allow it to change sign only on the next iteration, and
(2) if $\lambda_i = 0$, and $\left|\frac{\partial}{\partial\lambda_i} L_T(\boldsymbol{\lambda})\right| < \alpha$, do not allow λ_i to change, because it will immediately be driven back toward zero.

In some of the previous works (e.g., [22]) the Laplacian prior was reported to produce much worse performance than the Gaussian prior. Our experiments show them to perform similarly. The likely reason for the difference is poor handling of the zero discontinuities.

The RRM algorithm has three hyperparameters – the prior μ, the regularization parameter c, and the learning rate η.

4 Experimental Results

It is not possible to test every possible combination of algorithm, dataset and hyper-parameter. Therefore, we tried to do a meaningful series of experiments, which would together highlight the different aspects of the algorithms.

All of the results are presented as final microaveraged F1 scores.

4.1 Experiment 1

In the first series of experiments we evaluated the dependence of the performance of the classifiers upon their hyperparameters. We compared the performance of the

Table 1. RRM results on CoNLL-E dataset

	CoNLL-E_40K_set7			CoNLL-E_80K_set7			CoNLL-E_160K_set7		
μ=0.01	c=0.001	c=0.01	c=0.1	c=0.001	c=0.01	c=0.1	c=0.001	c=0.01	c=0.1
η=0.001	78.449	78.431	78.425	81.534	81.534	81.510	84.965	84.965	84.965
η=0.01	**85.071**	**85.071**	84.922	87.766	87.774	87.721	**90.246**	90.238	90.212
η=0.1	82.918	83.025	83.733	**87.846**	87.835	88.031	89.761	89.776	89.904
μ=0.1									
η=0.001	84.534	84.552	84.534	87.281	87.281	87.264	89.556	89.556	89.573
η=0.01	85.782	**85.800**	**85.800**	89.032	89.032	**89.066**	**91.175**	**91.175**	**91.150**
η=0.1	82.439	82.709	83.065	63.032	63.032	63.032	30.741	30.741	56.445
μ=1.0									
η=0.001	85.973	85.973	**85.990**	**89.108**	**89.108**	89.100	**91.056**	**91.056**	**91.056**
η=0.01	83.850	83.877	83.904	88.141	88.141	88.119	90.286	90.317	90.351
η=0.1	0	0	29.937	0	0	0	0	0	0

Table 2. RRM results on other datasets

	CoNLL-D_20K_set7			MUC7_40K_set2x			CLF_80K_set2		
μ=0.01	c=0.001	c=0.01	c=0.1	c=0.001	c=0.01	c=0.1	c=0.001	c=0.01	c=0.1
η=0.001	43.490	43.490	43.453	48.722	48.722	48.650	49.229	49.229	49.244
η=0.01	46.440	46.438	**46.472**	63.220	63.207	62.915	64.000	**64.040**	63.710
η=0.1	44.878	44.943	45.995	61.824	62.128	**63.678**	58.088	58.628	61.548
μ=0.1									
η=0.001	44.674	44.674	44.671	60.262	60.249	60.221	59.943	59.943	59.943
η=0.01	44.799	44.845	**44.957**	65.529	**65.547**	65.516	**64.913**	**64.913**	64.811
η=0.1	43.453	43.520	44.192	60.415	60.958	63.120	55.040	55.677	60.161
μ=1.0									
η=0.001	44.682	44.682	**44.694**	**66.231**	**66.231**	66.174	**65.408**	**65.408**	**65.408**
η=0.01	43.065	43.080	43.195	62.622	62.579	62.825	59.197	59.311	59.687
η=0.1	0	0	6.123	2.922	2.922	8.725	0	0	1.909

Table 3. CRF results on a selection of datasets

CRF	CLF			CoNLL-D			MUC7	CoNLL-E
	20K_set2	40K_set2	80K_set2	40K_set1	80K_set1	160K_set1	80K_set0	80K_set0
GAU σ = 1	**76.646**	**78.085**	**80.64**	29.851	35.516	**39.248**	**80.756**	69.247
GAU σ = 3	75.222	77.553	79.821	28.530	35.771	38.254	80.355	**69.693**
GAU σ = 5	75.031	77.525	79.285	29.901	35.541	38.671	79.853	69.377
GAU σ = 7	74.463	77.633	79.454	**30.975**	**36.517**	38.748	79.585	69.341
GAU σ = 10	74.352	77.05	77.705	29.269	36.091	38.833	80.625	68.974
LAP α=0.01	73.773	77.446	79.071	29.085	**35.811**	38.947	79.738	69.388
LAP α=0.03	75.023	77.242	78.810	31.082	34.097	38.454	79.044	**69.583**
LAP α=0.05	**76.314**	77.037	79.404	30.303	35.494	**39.248**	**79.952**	69.161
LAP α=0.07	74.666	76.329	**80.841**	30.675	34.530	38.882	79.724	68.806
LAP α=0.1	74.985	**77.655**	80.095	**31.161**	35.187	39.234	79.185	68.955

Table 4. MEMM results on a selection of datasets

MEMM	CLF			CoNLL-D			MUC7	CoNLL-E
	20K_set2	40K_set2	80K_set2	40K_set1	80K_set1	160K_set1	80K_set0	80K_set0
GAU σ = 1	**75.334**	**78.872**	**79.364**	**30.406**	35.013	**40.164**	**78.773**	67.537
GAU σ = 3	74.099	75.693	77.278	28.484	**35.330**	40.005	77.295	67.401
GAU σ = 5	73.959	74.685	77.316	28.526	35.043	39.799	77.489	67.870
GAU σ = 7	73.411	74.505	77.563	28.636	34.630	38.531	77.255	67.897
GAU σ = 10	73.351	74.398	77.379	28.488	33.955	37.830	77.094	**68.043**
LAP α=0.01	71.225	74.04	75.721	28.316	34.329	40.074	**78.312**	67.871
LAP α=0.03	72.603	72.967	76.540	29.086	35.159	38.621	77.385	67.401
LAP α=0.05	71.921	**75.523**	75.370	30.425	33.942	39.984	78.262	**67.908**
LAP α=0.07	72.019	74.486	**77.197**	30.118	**35.250**	39.195	76.646	67.833
LAP α=0.1	**72.695**	75.311	76.335	**30.315**	33.487	**40.861**	78.141	67.421

classifiers on a selection of datasets, with different hyperparameter values. All of the algorithms showed moderate and rather irregular dependence upon their hyperparameters. However, single overall set of values can be selected.

The RRM results are shown in the Table 1 and the Table 2. As can be seen, selecting $\mu = 0.1$, $c = 0.01$ and $\eta = 0.01$ gives reasonably close to optimal performance on all datasets. All subsequent experiments were done with those hyperparameter values.

Likewise, the ME-based algorithms have no single best set of hyperparameter values, but have close enough near-optimal values. A selection of MEMM and CRF results is shown in the Table 3 and Table 4. For subsequent experiments we use CRF with Laplacian prior with $\alpha = 0.07$ and MEMM with Gaussian prior with $\sigma = 1$.

4.2 Training Size

In this series of experiments we evaluated the performance of the algorithms using progressively bigger training datasets: 10K, 200K, 400K, 800K and 1600K tokens. The results are summarized in the Fig.1. As expected, the algorithms exhibit very similar training size vs. performance behavior.

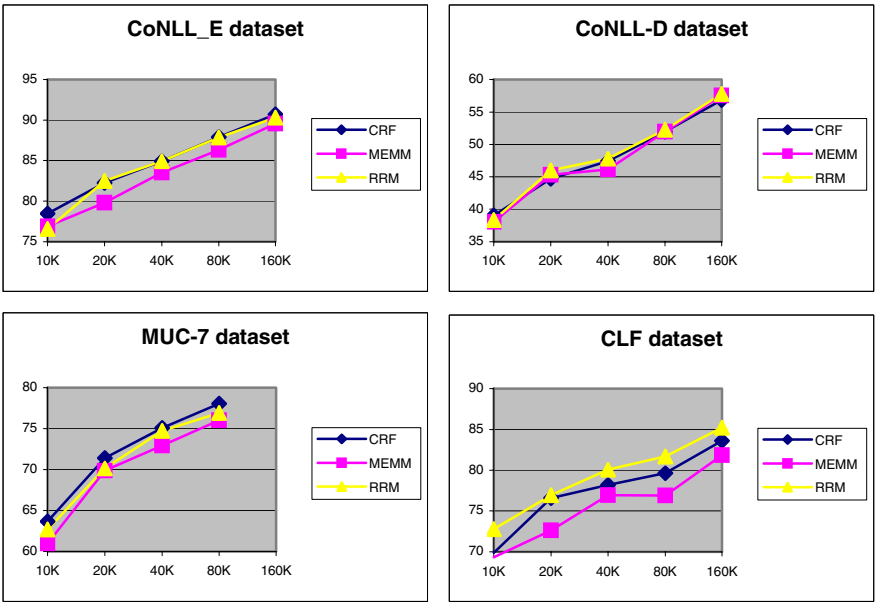

Fig. 1. Performance of the algorithms with different training sizes

Table 5. Performance of the algorithms with different feature sets

	MUC7			CoNLL-D			CoNLL-E		
	CRF	MEMM	RRM	CRF	MEMM	RRM	CRF	MEMM	RRM
set0	**75.748**	66.582	62.206	**48.988**	43.36	40.109	**87.379**	82.281	76.887
set1	**75.544**	67.075	68.405	**50.672**	49.164	48.046	**87.357**	82.516	81.788
set2	**75.288**	74.002	74.755	**52.128**	~52.01	51.537	86.891	87.089	**87.763**
set3	**76.913**	76.333	76.794	~60.172	59.526	**61.103**	88.927	88.711	**89.110**
set4	**78.336**	77.887	77.828	62.79	63.58	**65.802**	~90.037	~90.047	**90.722**
set5				~65.649	65.319	**67.813**	~90.139	~90.115	**90.559**
set6	**78.969**	78.442	78.016				~90.569	~90.492	**90.982**
set7	**81.791**	80.923	81.057				~91.414	90.88	**91.777**

4.3 Feature Sets

In this series of experiments we trained the algorithms with all available training data, but using different feature sets. The results are summarized in the Table 5. The results were tested for statistical significance using the McNemar test. All the perform-

ance differences between the successive feature sets are significant at least at the level p=0.05, except for the difference between set4 and set5 in CoNLL-E dataset for all models, and the differences between set0, set1, and set2 in CoNLL-E and MUC7 datasets for the CRF model. Those are statistically insignificant. The differences between the performance of different models that use same feature sets are also mostly significant. Exceptions are the numbers preceded by a tilda "~". Those numbers are not significantly different from the best results in their corresponding rows.

As can be seen, both CRF and RRM generally outperform MEMM. Among the two, the winner appears to depend upon the dataset. Also, it is interesting to note that CRF always wins, and by a large margin, on feature sets 0 and 1, which are distinguished from the set 2 by absense of "forward-looking" features. Indeed, using "forward-looking" features produces little or no improvement for CRF, but very big improvement for local models, probably because such features help to alleviate the *label bias problem* [7].

5 Conclusions

We have presented the experiments comparing the three common state-of-the-art feature-rich probabilistic sentence classifiers inside a single system, using completely identical feature sets. The experiments show that both CRF and RRM significantly outperform MEMM, while themselves performing roughly similarly. Thus, it shows that the comparatively poor performance of CRF in the CoNLL 2003 NER task [16] is due to suboptimal feature selection and not to any inherent flaw in the algorithm itself.

Also, we demonstrated that the Laplacian prior performs just as well and sometimes better than Gaussian prior, contrary to the results of some of the previous researches.

On the other hand, the much simpler RRM classifier performed just as well as CRF and even outperformed it on some of the datasets. The reason of such surprisingly good performance invites further investigation.

References

1. Aitken, J. S.: Learning Information Extraction Rules: An Inductive Logic Programming approach. 15th European Conference on Artificial Intelligence. IOS Press. (2002)
2. Bikel, D. M., Schwartz, R., Weischedel, R.M.: An Algorithm that Learns What's in a Name. Machine Learning. (34): (1999) 211-231.
3. Chieu, H.L., Tou Ng, H.: Named Entity Recognition: A Maximum Entropy Approach Using Global Information. Proceedings of the 17th International Conference on Computational Linguistics. (2002)
4. McCallum, A., Freitag, D., Pereira, F.: Maximum Entropy Markov Models for Information Extraction and Segmentation. Proceedings of the 17th International Conference on Machine Learning. (2000)
5. Sun A., et al.: Using Support Vector Machine for Terrorism Information Extraction. 1st NSF/NIJ Symposium on Intelligence and Security Informatics. (2003)

6. Kushmerick, N., Johnston, E., McGuinness, S.: Information extraction by text classification. IJCAI-01 Workshop on Adaptive Text Extraction and Mining. Seattle, WA. (2001)
7. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proc. 18th International Conf. on Machine Learning. (2001)
8. Rosenfeld, B., Feldman, R., Fresko, M., Schler, J., Aumann, Y.: TEG - A Hybrid Approach to Information Extraction. Proc. of the 13th ACM. (2004)
9. Berger, A., della Pietra, S., della Pietra, V.: A maximum entropy approach to natural language processing. Computational Linguistics 22(1), (1996) 39-71.
10. Darroch, J.N., Ratcliff, D.: Generalized iterative scaling for log-linear models. Annals of Mathematical Statistics (1972). 43(5): 1470-1480.
11. Borthwick, A., Sterling, J., Agichtein, E., Grishman, R.: Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In the proceedings of the 6th Workshop on Very Large Corpora. (1998)
12. Kim Sang, T., Erik, F., De Meulder, F.: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. Edmonton, Canada (2003)
13. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named Entity Recognition through Classifier Combination. In: Proceedings of CoNLL-2003, Edmonton, Canada, (2003), pp. 168-171.
14. Zhang, T., Johnson, D.: A Robust Risk Minimization based Named Entity Recognition System. In: Proceedings of CoNLL-2003, Edmonton, Canada, (2003), pp. 204-207.
15. Chieu, H.L., Tou Ng, H.: Named Entity Recognition with a Maximum Entropy Approach. In: Proceedings of CoNLL-2003, Edmonton, Canada, (2003), pp. 160-163.
16. McCallum, A., Li, W.: Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In: Proceedings of CoNLL-2003, Edmonton, Canada, (2003), pp. 188-191.
17. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Dissertation, Kluwer, (2002)
18. Nigam, K., Lafferty, J., McCallum, A.: Using maximum entropy for text classification. In IJCAI-99 Workshop on Machine Learning for Information (1999) 61—67.
19. Sha, F., Pereira, F.: Shallow parsing with conditional random fields, Technical Report CIS TR MS-CIS-02-35, University of Pennsylvania, (2003)
20. Zhang, T., Damerau, F., Johnson, D.: Text Chunking using Regularized Winnow. Meeting of the Association for Computational Linguistics. (2001) 539-546
21. Zhang, T., Regularized Winnow Methods. NIPS, (2000) 703-709
22. Peng, F., McCallum, A.: Accurate Information Extraction from Research Papers Using Conditional Random Fields. (1997)
23. Chinchor, N. MUC-7 Named Entity Task Definition Dry Run Version, Version 3.5. Proceedings of the Seventh Message Understanding Conference. (1998)
24. Borthwick, A., Sterling, J., Agichtein, E., Grishman, R. Description of the MENE Named Entity System as Used in MUC-7. Proceedings of the Seventh Message Understanding Conference. (1998)