NER Systems that Suit User's Preferences: Adjusting the Recall-Precision Trade-off for Entity Extraction

Einat Minkov, Richard C. Wang
Language Technologies Institute
Carnegie Mellon University
einat,rcwang@cs.cmu.edu

Anthony Tomasic
Inst. for Software Research International
Carnegie Mellon University
tomasic@cs.cmu.edu

William W. Cohen
Machine Learning Dept.
Carnegie Mellon University
wcohen@cs.cmu.edu

Abstract

We describe a method based on "tweaking" an existing learned sequential classifier to change the recall-precision tradeoff, guided by a user-provided performance criterion. This method is evaluated on the task of recognizing personal names in email and newswire text, and proves to be both simple and effective.

1 Introduction

Named entity recognition (NER) is the task of identifying named entities in free text—typically personal names, organizations, gene-protein entities, and so on. Recently, sequential learning methods, such as hidden Markov models (HMMs) and conditional random fields (CRFs), have been used successfully for a number of applications, including NER (Sha and Pereira, 2003; Pinto et al., 2003; Mccallum and Lee, 2003). In practice, these methods provide imperfect performance: precision and recall, even for well-studied problems on clean well-written text, reach at most the mid-90's. While performance of NER systems is often evaluated in terms of $F1$ measure (a harmonic mean of precision and recall), this measure may not match user preferences regarding precision and recall. Furthermore, learned NER models may be sub-optimal also in terms of F1, as they are trained to optimize other measures (e.g., loglikelihood of the training data for CRFs).

Obviously, different applications of NER have different requirements for precision and recall. A system might require high precision if it is designed to extract entities as one stage of fact-extraction, where facts are stored directly into a database. On the other hand, a system that generates candidate extractions which are passed to a semi-automatic curation system might prefer higher recall. In some domains, such as anonymization of medical records, high recall is essential.

One way to manipulate an extractor's precision-recall tradeoff is to assign a confidence score to each extracted entity and then apply a global threshold to confidence level. However, confidence thresholding of this sort cannot increase recall. Also, while confidence scores are straightforward to compute in many classification settings, there is no inherent mechanism for computing confidence of a sequential extractor. Culotta and McCallum (2004) suggest several methods for doing this with CRFs.

In this paper, we suggest an alternative simple method for exploring and optimizing the relationship between precision and recall for NER systems. In particular, we describe and evaluate a technique called "extractor tweaking" that optimizes a learned extractor with respect to a specific evaluation metric. In a nutshell, we directly *tweak* the threashold term that is part of any linear classifier, including sequential extractors. Though simple, this approach has not been empirically evaluated before, to our knowledge. Further, although sequential extractors such as HMMs and CRFs are state-of-the-art methods for tasks like NER, there has been little prior research about tuning these extractors' performance to suit user preferences. The suggested algorithm optimizes the system performance per a user-provided

evaluation criterion, using a linear search procedure. Applying this procedure is not trivial, since the underlying function is not smooth. However, we show that the system's precision-recall rate can indeed be tuned to user preferences given labelled data using this method. Empirical results are presented for a particular NER task—recognizing person names, for three corpora, including email and newswire text.

2 Extractor tweaking

Learning methods such as VP-HMM and CRFs optimize criteria such as margin separation (implicitly maximized by VP-HMMs) or log-likelihood (explicitly maximized by CRFs), which are at best indirectly related to precision and recall. Can such learning methods be modified to more directly reward a user-provided performance metric?

In a non-sequential classifier, a threshold on confidence can be set to alter the precision-recall tradeoff. This is nontrivial to do for VP-HMMs and CRFs. Both learners use dynamic programming to find the label sequence $\mathbf{y} = (y_1, \ldots, y_i, \ldots, y_N)$ for a word sequence $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_N)$ that maximizes the function $\mathbf{W} \cdot \sum_i \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$, where $\mathbf{W}$ is the learned weight vector and $\mathbf{f}$ is a vector of features computed from $\mathbf{x}$, i, the label y_i for x_i, and the previous label y_{i-1}. Dynamic programming finds the most likely state sequence, and does not output probability for a particular sub-sequence. (Culotta and McCallum, 2004) suggest several ways to generate confidence estimation in this framework. We propose a simpler approach for directly manipulating the learned extractor's precision-recall ratio.

We will assume that the labels y include one label O for "outside any named entity", and let w_0 be the weight for the feature f_0, defined as follows:

$$f_0(\mathbf{x}, i, y_{i-1}, y_i) \equiv \begin{cases} 1 & \text{if } y_i = O \\ 0 & \text{else} \end{cases}$$

If no such feature exists, then we will create one. The NER based on $\mathbf{W}$ will be sensitive to the value of w_0: large negative values will force the dynamic programming method to label tokens as inside entities, and large positive values will force it to label fewer entities[1].

[1]We clarify that w_0 will refer to feature f_0 only, and not to other features that may incorporate label information.

We thus propose to "tweak" a learned NER by varying the single parameter w_0 systematically so as to optimize some user-provided performance metric. Specifically, we tune w_0 using a a Gauss-Newton line search, where the objective function is iteratively approximated by quadratics.[2] We terminate the search when two adjacent evaluation results are within a 0.01% difference[3].

A variety of performance metrics might be imagined: for instance, one might wish to optimize recall, after applying some sort of penalty for precision below some fixed threshold. In this paper we will experiment with performance metrics based on the (complete) F-measure formula, which combines precision and recall into a single numeric value based on a user-provided parameter β:

$$F(\beta, P, R) = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

A value of $\beta > 1$ assigns higher importance to recall. In particular, F_2 weights recall twice as much as precision. Similarly, $F_{0.5}$ weights precision twice as much as recall.

We consider optimizing both token- and entity-level F_β – awarding partial credit for partially extracted entities and no credit for incorrect entity boundaries, respectively. Performance is optimized over the dataset on which $\mathbf{W}$ was trained, and tested on a separate set. A key question our evaluation should address is whether the values optimized for the training examples transfer well to unseen test examples, using the suggested approximate procedure.

3 Experiments

3.1 Experimental Settings

We experiment with three datasets, of both email and newswire text. Table 1 gives summary statistics for all datasets. The widely-used *MUC-6* dataset includes news articles drawn from the Wall Street Journal. The *Enron* dataset is a collection of emails extracted from the Enron corpus (Klimt and Yang, 2004), where we use a subcollection of the messages located in folders named "meetings" or "calendar". The *Mgmt-Groups* dataset is a second email

[2]from http://billharlan.com/pub/code/inv.

[3]In the experiments, this is usually within around 10 iterations. Each iteration requires evaluating a "tweaked" extractor on a training set.

collection, extracted from the CSpace email corpus, which contains email messages sent by MBA students taking a management course conducted at Carnegie Mellon University in 1997. This data was split such that its test set contains a different mix of entity names comparing to training exmaples. Further details about these datasets are available elsewhere (Minkov et al., 2005).

	# documents			# names
	Train	Test	# tokens	per doc.
MUC-6	347	30	204,071	6.8
Enron	833	143	204,423	3.0
Mgmt-Groups	631	128	104,662	3.7

Table 1: Summary of the corpora used in the experiments

We used an implementation of Collins' voted-percepton method for discriminatively training HMMs (henceforth, VP-HMM) (Collins, 2002) as well as CRF (Lafferty et al., 2001) to learn a NER. Both VP-HMM and CRF were trained for 20 epochs on every dataset, using a simple set of features such as word identity and capitalization patterns for a window of three words around each word being classified. Each word is classified as either inside or outside a person name.[4]

3.2 Extractor tweaking Results

Figure 1 evaluates the effectiveness of the optimization process used by "extractor tweaking" on the Enron dataset. We optimized models for F_β with different values of β, and also evaluated each optimized model with different F_β metrics. The top graph shows the results for token-level F_β, and the bottom graph shows entity-level F_β behavior. The graph illustates that the optimized model does indeed roughly maximize performance for the target β value: for example, the token-level F_β curve for the model optimized for $\beta = 0.5$ indeed peaks at $\beta = 0.5$ on the test set data. The optimization is only roughly accurate[5] for several possible reasons: first, there are differences between train and test sets; in addition, the line search assumes that the performance metric is smooth and convex, which need not be true. Note that evaluation-metric optimization is less successful for entity-level performance,

[4]This problem encoding is basic. However, in the context of this paper we focus on precision-recall trade-off in the general case, avoiding settings' optimization.

[5]E.g, the token-level F_2 curve peaks at $\beta = 5$.

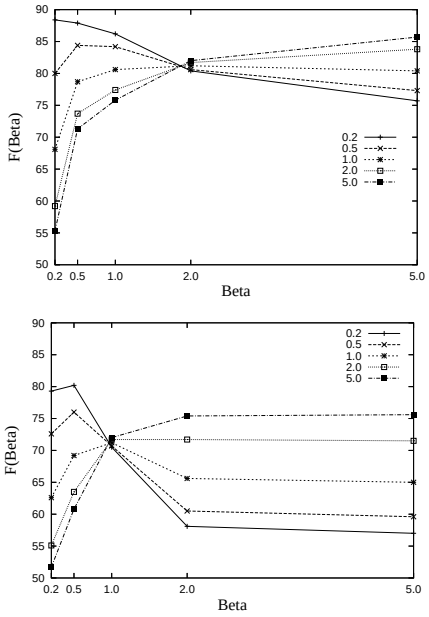

Figure 1: Results of token-level (top) and entity-level (bottom) optimization for varying values of β, for the Enron dataset, VP-HMM. The y-axis gives F in terms of β. β (x-axis) is given in a logarithmic scale.

which behaves less smoothly than token-level performance.

	Token		Entity	
β	Prec	Recall	Prec	Recall
Baseline	*93.3*	*76.0*	*93.6*	*70.6*
0.2	100	53.2	98.2	57.0
0.5	95.3	71.1	94.4	67.9
1.0	88.6	79.4	89.2	70.9
2.0	81.0	83.9	81.8	70.9
5.0	65.8	91.3	69.4	71.4

Table 2: Sample optimized CRF results, for the MUC-6 dataset and entity-level optimization.

Similar results were obtained optimizing baseline CRF classifiers. Sample results (for MUC-6 only, due to space limitations) are given in Table 2, optimizing a CRF baseline for entity-level F_β. Note that as β increases, recall monotonically increases and precision monotonically falls.

The graphs in Figure 2 present another set of results with a more traditional recall-precision curves. The top three graphs are for token-level F_β optimization, and the bottom three are for entity-level optimization. The solid lines show the token-level and entity-level precision-recall tradeoff obtained by

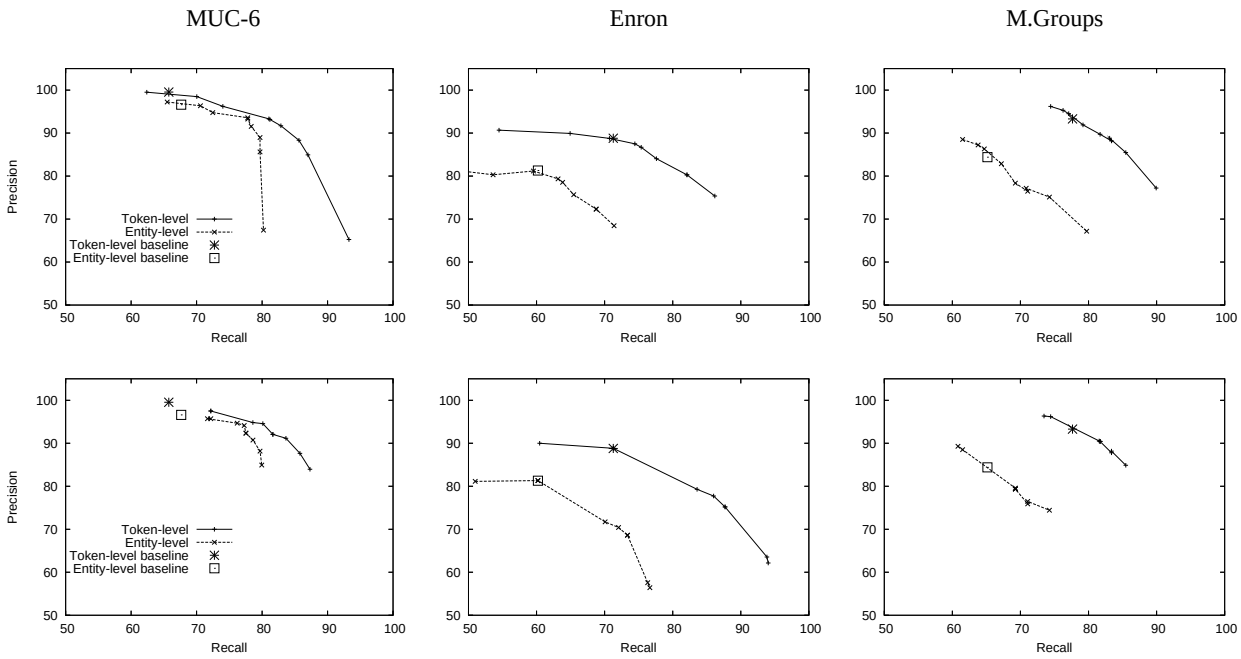

Figure 2: Results for the evaluation-metric model optimization. The top three graphs are for token-level $F(\beta)$ optimization, and the bottom three are for entity-level optimization. Each graph shows the baseline learned VP-HMM and evaluation-metric optimization for different values of β, in terms of both token-level and entity-level performance.

varying[6] β and optimizing the relevant measure for F_β; the points labeled "baseline" show the precision and recall in token and entity level of the baseline model, learned by VP-HMM. These graphs demonstrate that extractor "tweaking" gives approximately smooth precision-recall curves, as desired. Again, we note that the resulting recall-precision tradeoff for entity-level optimization is generally less smooth.

4 Conclusion

We described an approach that is based on modifying an existing learned sequential classifier to change the recall-precision tradeoff, guided by a user-provided performance criterion. This approach not only allows one to explore a recall-precision tradeoff, but actually allows the user to specify a performance metric to optimize, and optimizes a learned NER system for that metric. We showed that using a single free parameter and a Gauss-Newton line search (where the objective is iteratively approximated by quadratics), effectively optimizes two plausible performance measures, token-level F_β and entity-level F_β. This approach is in fact general, as it is applicable for sequential and/or structured learning applications other than NER.

[6] We varied β over the values 0.2, 0.5, 0.8, 1, 1.2, 1.5, 2, 3 and 5

References

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. In *HLT-NAACL*.

B. Klimt and Y. Yang. 2004. Introducing the Enron corpus. In *CEAS*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

A. Mccallum and W. Lee. 2003. early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CONLL*.

E. Minkov, R. C. Wang, and W. W. Cohen. 2005. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT-EMNLP*.

D. Pinto, A. Mccallum, X. Wei, and W. B. Croft. 2003. table extraction using conditional random fields. In *ACM SIGIR*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*.