Stopping Criteria for Active Learning of Named Entity Recognition

Florian Laws
Institute for NLP
Universität Stuttgart
fl@ifnlp.org

Hinrich Schütze
Institute for NLP
Universität Stuttgart
hs999@ifnlp.org

Abstract

Active learning is a proven method for reducing the cost of creating the training sets that are necessary for statistical NLP. However, there has been little work on stopping criteria for active learning. An operational stopping criterion is necessary to be able to use active learning in NLP applications. We investigate three different stopping criteria for active learning of named entity recognition (NER) and show that one of them, *gradient-based stopping*, (i) reliably stops active learning, (ii) achieves near-optimal NER performance, (iii) and needs only about 20% as much training data as exhaustive labeling.

1 Introduction

Supervised statistical learning methods are important and widely successful tools for natural language processing. These methods learn by estimating a statistical model on labeled training data. Often, these models require a large amount of training data that needs to be hand-annotated by human experts. This is time-consuming and expensive. Active learning (AL) reduces this annotation effort by selecting unlabeled examples that are maximally informative for the statistical learning method and handing them to a human annotator for labeling. The statistical model is then updated with the newly gathered information. In this paper, we adopt the uncertainty sampling approach to AL (Lewis and Gale, 1994). Uncertainty sampling selects those examples in the pool as most informative for which the statistical classifier is least certain in its classification decision.

While AL is an active area of research in NLP, the issue of determining when to stop the AL process has only recently come into focus (Zhu and Hovy, 2007; Vlachos, 2008). This is somewhat surprising because the main purpose of active learning is to save on annotation effort; deciding on the point when enough data is annotated is crucial to fulfilling this goal.

We investigate three different stopping criteria in this paper. First, a user of a classification system may want to set a *minimum absolute performance* for the system to be deployed. The standard way of assessing classifier performance uses a held-out labeled test set. However, labeling a test set of sufficient size is contrary to the goal of minimizing annotation effort and impractical in most real-world settings. We will show that the classifier can estimate its own performance using only an unlabeled reference set and propose to stop active learning if estimated performance reaches the threshold set by the user. The estimation is somewhat inaccurate, however, and we investigate possible reasons for estimation error.

An alternative criterion is based on *maximum possible performance*. We will show that our performance estimation method supports stopping AL at a point where performance is almost optimal.

The third and last criterion is *convergence*. The basic idea here is to stop active learning when more examples from the pool do not contribute more information, indicated either by the fact that the classifier has reached maximum performance or by the fact that the "uncertainty" of the classifier cannot be decreased further. We determine the point where the pool has become uninformative by computing the gradient of either performance or uncer-

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 465–472
Manchester, August 2008

tainty.

This paper is organized as follows. Section 2 shows that three uncertainty measures achieve near-optimal performance for NER at a fraction of the labeling cost of exhaustive labeling of the training set. In Section 3, we introduce a new method for estimating the performance of an actively learned classifier in support of stopping active learning when a certain level of performance has been reached. Section 4 shows that the stopping criterion of reaching peak confidence is not applicable to NER with multiclass logistic regression. Section 5 presents stopping criteria based on convergence. Sections 6 and 7 discuss related work and present our conclusions.

2 Selection Functions

For measuring the uncertainty of a classification decision in uncertainty sampling there exist diverse measures appropriate for different basic classifiers (e.g. margin-based measures for SVMs, and measures based on class probability for classification). Choosing such an uncertainty measure is relatively straightforward for a binary classification problem, but for multiclass problems we need different measures, and it is not obvious which will perform best.

Following Schein (2005), but in the context of NER, we compare several measures of uncertainty for multiclass logistic regression. For a given measure $M_{i,X}$, we select in each iteration the unlabeled example(s) in the pool that have the *smallest* value for $M_{i,X}$ (corresponding to the *maximum* uncertainty).

1-Entropy.

$$M_{i,\textit{1-Entropy}} = 1 - H(\hat{p}(.|x_i)) = 1 + \sum_j \hat{p}(c_j|x_i) \log \hat{p}(c_j|x_i)$$

where $\hat{p}(c_j|x_i)$ is the current estimate of the probability of class c_j given the example x_i.[1] *1-Entropy* favors examples where the classifier assigns similar probabilities to all classes.

Margin. If c and c' are the two most likely classes, the margin is defined as follows:

$$M_{i,\textit{Margin}} = |\hat{p}(c|x_i) - \hat{p}(c'|x_i)|$$

Margin picks examples where the distinction between two likely classes is hard.

MinMax.

$$M_{i,\textit{MinMax}} = \max_j(\hat{p}(c_j|x_i))$$

The rationale here is that a low probability of the selected class indicates uncertainty. We propose *MinMax* as a measure that is more directly based on the classifier's decision for a particular example. The other two measures also take into account the classifier's assessment of classes that were *not* chosen for the unlabeled example.

2.1 Experiments

We used the newswire section of the ACE 2005 Multilingual Training Corpus (128 documents, 66,015 tokens) for our experiments. A subset of the documents was randomly sampled into an evaluation set that consists of 6301 tokens. We used 30.000 of the remaining tokens as the uncertainty sampling pool. The rest was left aside for future experiments. We use the BBR package (Genkin et al., 2007) for binary logistic regression as our base classifier, with default values for all of BBR's parameters. As our main focus is on AL, we only use basic features like capitalization, puctuation as well as word identity, prefixes and suffixes, each for the classified word itself and for left and right contexts.

We train separate classifiers for each named entity (NE) class and another one for the class "not an NE" (0). For each token we normalize the output probability of the individual classifiers so they sum to 1 and then select for each token the class with the highest probability. Evaluation is performed by comparing individual tokens to the gold standard.[2]

Using all labeled training data as our fully supervised baseline results in a performance of 78.7% F_1 (henceforth: F) and 96.6% accuracy. This is comparable to the accuracy of 96.29% reported by (Daume III, 2007) on the newswire domain. Daumé's work is the only study known to us that uses the ACE dataset, but not the proprietary ACE value score. In the rest of this paper, we report F scores, because we believe that F is a more informative measure for NER than accuracy.

We use AL based on uncertainty sampling. We start with a seed set of ten consecutive tokens randomly selected from the training pool and label it. In each round of AL we select the ten tokens with the smallest value of $M_{i,X}$ (where

[1] We use *1-Entropy* instead of entropy, so all three measures will have lower values for less certain instances.

[2] Chunk-based NER results are not directly comparable with this token-based evaluation.

Selection	Baseline		Peak perf.	
1-Entropy	78.7	2139 (7.1%)	80.8	3460 (11.5%)
MinMax	78.7	2108 (7.0%)	80.8	3650 (12.1%)
Margin	78.7	2019 (6.7%)	81.2	3694 (12.3%)

Table 1: Percentage of data needed by AL to reach baseline or peak performance.

$X \in \{$*1-Entropy*, *Margin*, *MinMax*$\}$) from the remaining pool, including tokens with the label 0. We then label these tokens and add them to the labeled training set. The classifiers are retrained with the new training set and the AL loop repeats. We performed 20 runs of the experiments, each with the same sampling pool, but a different seed set, randomly selected as described above.

Table 1 shows that AL is quite successful for NER. Only 7% of the training data is needed to achieve the same performance as the supervised baseline.

Furthermore we find that after the baseline performance is reached the increase in performance quickly levels off to a point where using more training data does not yield performance improvements anymore. In fact, our experiments show that there is a peak in performance reached at about 12% of the training data and performance decreases again after this point (see Figure 1). The peak is more prominent if the pool is large. On a pool of 30,000 tokens, peak performance is about 2.5% F-Score better than the baseline; on a 6000 token pool, the difference is only about 1.7%. Therefore, once the peak is reached, the AL process should stop, even if the annotation budget is not yet used up.

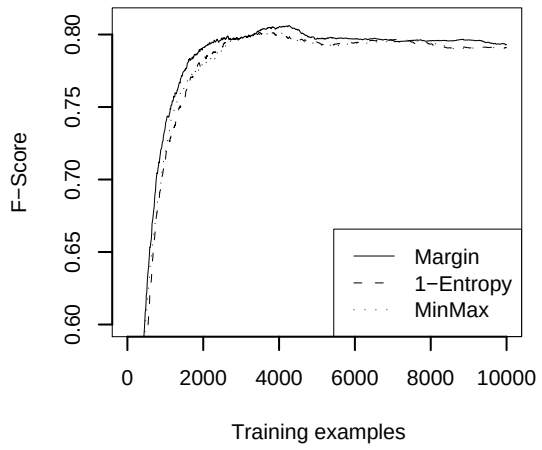

Figure 1: Performance as a function of number of labeled training examples used

Comparing the different selection functions, we found little difference between their performance. *Margin* performs significantly better (Student's t-test, $\alpha = 0.05$), but the difference is small ($< 1\%$ F-Score). If we compare two AL processes (say *Margin* and *1-Entropy*) that were started with the same pool and seed set and stop both processes when they each reach their respective peak performances, *Margin* has a better peak performance of 0.3% F-Score on average (significant at $\alpha = 0.05$).

The differences between *1-Entropy* and *MinMax* are not statistically significant, except for a short start-up phase (see Figure 1).

3 Performance Estimation

In practical applications, classifiers can only be reliably deployed when they attain a predefined *minimum absolute performance* level. Thus, we would like to determine if this level has been reached and then stop the annotation process. However, this is not a simple task, because in these settings there is no labeled test set available to evaluate performance. Creating this test set would mean a substantial annotation effort, which is what we want to avoid by using AL in the first place. Therefore, we will try to estimate the classifier's performance on unlabeled data.

Following Lewis (1995), we estimate the F-Score based on the current estimates of the class probabilities. Based on the F measure's definition as the harmonic mean of precision (P) and recall (R), we can write F as a function of true positives (TP), false positives (TP) and false negatives (FN):

$$F = \frac{2 \cdot P \cdot R}{P + R} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Similar to Lewis, we estimate $\hat{\text{TP}}$, $\hat{\text{FP}}$, $\hat{\text{FN}}$, but we need to extend their work from binary classification to 1-vs-all multiclass classification:

$$\hat{\text{TP}} = \sum_i^n \sum_j^E \hat{p}(c_j|x_i) d_{i,j} \quad (1)$$

$$\hat{\text{FP}} = \sum_i^n \sum_j^E (1 - \hat{p}(c_j|x_i)) d_{i,j} \quad (2)$$

$$\hat{\text{FN}} = \sum_i^n \sum_j^E \hat{p}(c_j|x_i)(1 - d_{i,j}) \quad (3)$$

where n is the number of examples, E is the number of named entity classes, excluding the "not

an NE" class. $\hat{p}(c_j|x_i)$ is the estimated probability that example x_i has class c_j. The flag $d_{i,j}$ indicates "is winning class": $d_{i,j} = 1$ if $j = \text{argmax}_j \hat{p}(c_j|x_i)$ and $d_{i,j} = 0$ else.

Like standard NER evaluation schemes, e.g. (Tjong Kim Sang and De Meulder, 2003), we consider only those decisions to be TPs where (i) the reference class matches the selected class and (ii) this class is not "not an NE". When estimating TP, we assume that the probability of a match equals the probability of the selected class (which is $\hat{p}(c_j|x_i) \cdot d_{i,j}$). The probability of making an FP error is just the remaining probability mass. For FN, we can calculate the estimated probability by summing up the class probabilities of the non-selected named entity classes.

3.1 Evaluation of Performance Estimation

To evaluate the performance estimation method, we ran it on an *unlabeled reference set*. The reference set is a set of unlabeled data distinct from the sampling pool. In our experiments, we use the tokens in the test set from 2.1, but with the labels stripped off.

We compare the true performance on the test set (reported as "True" in Table 2) with the estimate (reported as "Lewis"). The Δ columns report the difference of the named method to "True". We also tested leave-one-out (LOO) estimation of F, P and R using the data of the selected training set.

	True	Lewis	Δ Lewis	LOO	Δ LOO
F	79	92	+13	85	+6
P	81	92	+11	86	+5
R	77	92	+15	84	+7

Table 2: Performance estimation. LOO and Lewis overestimate true F by 6% and 13%, respectively.

We find that both methods overestimate precision and recall by a large margin. We also note that the peak in performance at about 4200 training examples that we found when evaluating on held-out data (see Figure 1) does not occur when evaluating performance using the Lewis method. Instead, the estimate of F grows monotonically. This means that we cannot use a peak of estimated F as a criterion for stopping. When setting an absolute threshold of $F = 80\%$ for stopping, active learning stops at about 1000 iterations, yielding a true performance of only $F = 73\%$ (selection by *Margin*, 20 trials). This indicates that we cannot directly use Lewis estimates for stopping.

3.2 Error Analysis

The reason for the overestimation is that the logistic regression classifier is too confident in its own decision. For positive decisions, the class probability very often is close to 1, for negative decisions, it is close to 0. As a result, the estimator gives very little score for FN (Equation 3) or FP (Equation 2) in most instances, which leads to the high overestimation of performance.

To verify this, we grouped the empirical probability of a selected class being the correct class in bins according to the estimated probability of the logistic classifier. Table 3 shows this empirical probability given a class and its estimate. The table is split into two halves, such that the empirical probabilities for positive decisions (the class got chosen as the best class) and negative decisions are shown separately. The top value in each cell ("emp") shows the empirical probability as opposed to the estimated probability, which is the value below ("est"). The product of the difference of these two probabilities and the number of instances that were counted into this bin ("cnt"), gives an estimate of how much the probability estimates in the bin contribute to the error (absolute value) of the performance estimation.

The table shows that class probabilities are in fact estimated too optimistically. For many of the entries in the positives table, the estimated probabilities are greater than the empirical probabilities. In the negatives table, the estimated probabilities are smaller. In both cases, the estimates are closer to the respective extreme values 1 or 0, which means they are overconfident. Note that for positive decisions, the estimation error of the values in a single bin contributes to the overall estimation error in two ways: overestimating TPs and underestimating FPs. For example, the estimation error for the cell in bold is 29.2, contributing -29.2 for FP (underestimation) and $+29.2$ for TP (overestimation). Also note that due to the high number of non-NE tokens in the text, there is a large number of negative decisions for each entity-class classifier; thus, small differences in the probabilities make large contributions to error.

We ran a separate experiment in which we trained a classifier on the entire labeled pool. The Lewis estimator overestimated F by 12% in this case. This indicates that the estimation error does not primarily come from the biased selection of training examples inherent in the selective sam-

		negative decisions			positive decisions			
		0-.2	.2-.4	.4-.6	.2-.4	.4-.6	.6-.8	.8-1
O	emp	0.0643	0.269	0.25	0.0	0.25	0.233	0.991
	est	0.00825	0.295	0.438	0.394	0.537	0.714	0.999
	cnt	607	26	12	1	16	30	5609
	err	34	-0.67	-2.25	0.394(tn)	4.6 (tn)	14.4 (tn)	45.4 (tn)
GPE	emp	0.00384	0.391	0.5	0.0	0.333	0.571	**0.875**
	est	0.000812	0.296	0.435	0.357	0.535	0.687	**0.989**
	cnt	5985	23	6	1	9	21	**256**
	err	18.1 (fn)	2.19 (fn)	0.388 (fn)	0.357 (fp)	1.82 (fp)	2.42 (fp)	**29.2 (fp)**
ORG	emp	0.00853	0.393	0.667		0.5	0.615	0.828
	est	0.000847	0.283	0.441		0.545	0.71	0.968
	cnt	6093	28	12		14	26	128
	err	46.8 (fn)	3.06 (fn)	2.7 (fn)		0.631 (fp)	2.46 (fp)	17.9 (fp)
PER	emp	0.0041	0.455	0.5		0.273	0.5	0.93
	est	0.000748	0.283	0.48		0.563	0.718	0.98
	cnt	6102	22	6		11	18	142
	err	20.4 (fn)	3.78 (fn)	0.121 (fn)		3.2 (fp)	3.93 (fp)	7.19 (fp)

Table 3: Empirical probabilities and contribution to estimation errors. (We omit small classes and empty columns.) Example (cell in bold): 256 tokens were estimated to be a GPE by the classifier with estimated probabilities between 0.8 and 1.0. The average estimate was 0.989. In reality, only 224 of these tokens (87.5%) were GPEs. The contribution of this cell to the overall FP count is $(0.989 - 0.875) \cdot 256 \approx 29.2$.

pling method, but from bias inherent in either the whole pool of training data or the base classifier.

3.3 Towards a Better Estimate

Over-optimistic estimates for precision and recall stem from the classifier's over-optimistic probability estimates. We try to correct the estimates by replacing the predicted class probabilities with the appropriate value in an empirical probability table like the one shown in Table 3. However, since in practice we do not have labels for the test set, we cannot compute the empirical probabilities directly. Instead, we use leave-one-out estimation to bootstrap the adjustment table from the selected training data. The adjusted estimation shows a marked increase in the estimates for FP and FN, leading to a quite accurate estimate for precision ($+5$ absolute error), but the now pessimistic estimate for recall (-16) leads to underestimation of F-Score overall (-8) (see Table 4).

	True	Lewis	adj. Lewis	Δ adj. Lewis
F	78	91	70	-8
P	81	93	86	+5
R	76	89	60	-16
TP	520	596	555	+35
FP	125	48	90	-35
FN	163	70	379	+216

Table 4: Lewis estimation with adjusted probabilities

As we see, the adjustment overshoots for recall, indicating that the new estimated probabilities are still off. There could be several reasons for this. The first reason is that the bin width is quite coarse, as there are only five bins for the entire probability interval, each bin covering a range of 0.2. However, using finer bin widths can lead to data sparsity problems.

Another reason might be the estimation errors within individual bins that compound to a quite large overall error especially in the negative case. Finally, differences in the distributions of training set and reference set could cause unreliable estimates. The empirical probabilities for the adjustment table are estimated with leave-one-out on the training set. However, since the training set is created by selective sampling, it will be biased.

4 Confidence-based Stopping

We have found that performance estimation is not yet reliable enough to stop when a desired performance level is reached. However, since there is a maximum performance that can be reached on any given sampling pool, the annotation process still should stop at this point regardless of whether a target performance level has been reached or not. We therefore seek a stopping criterion that finds the *maximum possible performance* when the classifier is iteratively trained on a given sampling pool. Again, in practice we do not have a labeled test set to evaluate against, so we have to try to find the stopping point from either the remaining pool, or the separate unlabeled reference set.

Vlachos (2008) proposes to calculate the *confidence* of the classifier by using the *average uncer-*

tainty on the unlabeled reference set. For multiclass problems, he uses SVM classifiers with the SVM margin size as the uncertainty measure. Using this measure, Vlachos reports finding, albeit distorted by fluctuations, a peak pattern in this confidence measure that coincides with reaching maximal performance in his experiments. He then suggests to use this *peak confidence* as the stopping criterion.

However, in our experiments with multiclass logistic regression, we could not find this peak pattern when calculating the confidence using the three uncertainty measures introduced above: *1-Entropy*, *Margin* and *MinMax*.

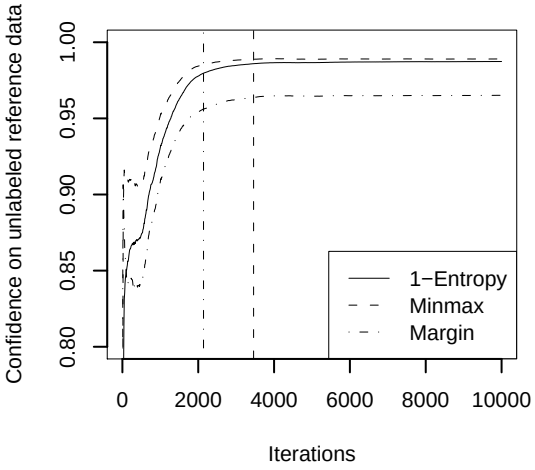

Figure 2: Confidence on unlabeled reference set (selection: *1-Entropy*). The vertical lines indicate when baseline and optimal performance are reached. There is no peak pattern in the curves, so reaching peak confidence cannot be used as a stopping criterion.

In Figure 2, we show the three measures, averaged over 20 trials as described in section 2.1. Due to instability of AL during start-up, there are some fluctuations in the first 100 iterations. After 500 iterations the confidence curves stabilize and at about 4000 iterations approach asymptotes, without exhibiting peak patterns. Thus, the proposed criterion of *peak confidence* based on average reference uncertainty does not seem applicable for controlling AL with multiclass logistic regression.

5 Gradient-based Stopping

Since we cannot use peaks for stopping, we propose to stop when a base measurement characterizing the progress of active learning has converged. We identify the point of convergence by computing gradients. We find that the rise of the performance estimation slows to an almost horizontal slope at about the time when the true performance reaches its peak. We therefore propose the following new stopping criterion: Estimate the gradient of the curve and stop when it approaches 0. Since we do not need an accurate estimation of absolute performance here, we can use the unadjusted Lewis estimate for this method. We call this stopping criterion (estimated) *performance convergence*.

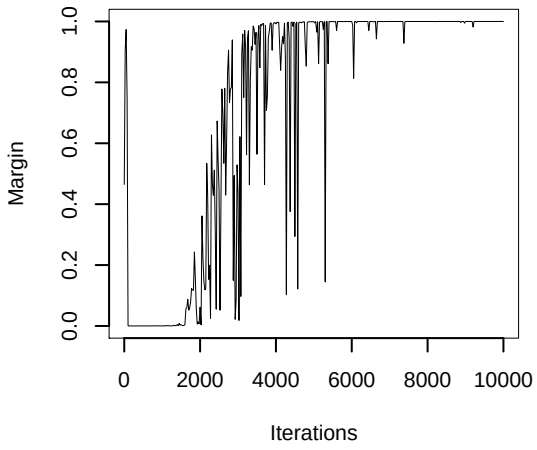

Figure 3: *Margin* uncertainty of selected instance (single run). The graph demonstrates that without smoothing this criterion is too noisy.

In a similar way, we can use the gradient of the uncertainty of the last selected instance. The instance that was selected last is always the one with maximum uncertainty, and thus the most informative for training. When the uncertainty measure comes close to the extreme value of 1, we decide that there are no informative examples left in the pool and we stop the AL process. (Unfortunately, 1 is minimum uncertainty and 0 is maximum uncertainty according to our definitions of the three measures.) The gradient of the uncertainty measure approaches 0 at this point (see Figure 3), so we can again use a gradient criterion for implementing this idea. We call this stopping criterion *uncertainty convergence*.

In Figure 3, which shows a graph of the *Margin* uncertainty of the selected instance, we can also see that it is quite noisy. The value drops sharply when some examples are encountered but quickly returns to the previous level after a few iterations. The performance estimation measure is

slightly noisy as well, so we need a robust way of computing the gradient. We achieve this with a moving median approach. At each step, we compute the median of $w_2 = \{a_{n-k}, \ldots, a_n\}$ (the last n values) and of $w_1 = \{a_{n-k-1}, \ldots, a_{n-1}\}$ (the previous last n values). Each value a_i is the performance at iteration i (for the performance gradient) or the uncertainty of the instance selected in iteration i (for the uncertainty gradient).

We then estimate the gradient using the medians of the two windows:

$$g = (\text{median}(w_2) - \text{median}(w_1))/1 \qquad (4)$$

For the performance estimate, which is less noisy, we can also use the arithmetic mean instead of the median. In this case, we simply replace "median" with "mean" in Equation 4.

We found that a window of size $k = 100$ yields good results in mitigating the noise while still reacting fast enough to the changes in the gradient. We combine this criterion with a maximum criterion and only stop if the last value a_n is a new maximum. We stop the AL process when (i) the current certainty or estimated performance is a new maximum and (ii) the newly calculated gradient g is positive and (iii) g falls below a predefined level ϵ.

5.1 Evaluation

We show the results of gradient stopping applied to each of the three uncertainty measures and the Lewis estimate. For comparison, we also include results with a threshold-based criterion, where AL stops when the uncertainty measure of the selected instance reaches a threshold of $1-\epsilon$. This is similar to (Zhu and Hovy, 2007), but extended by us to all three uncertainty measures.

Table 5 shows results for each criterion. The "Stop" value indicates number of tokens at which the stopping criterion stopped AL. "ΔBl" indicates the difference between baseline performance and performance at the stopping point, "ΔPk" the difference to peak performance. The "sd" columns show the respective standard deviations.

We find that all stopping criteria stop before 20% of the pool is used, providing a large reduction in annotation effort. While the point of peak performance can not be precisely found by the criteria, all criteria reliably stop at a performance level that surpasses the fully supervised baseline. The threshold criteria seem to be a bit better in finding a stopping point closer to optimal performance. Not unsurprisingly, the stopping function that matches the selection function performs best. The gradient methods, however, seem to be providing better-than-baseline performance more consistently (less variation) and might require less tuning of the threshold parameter when other factors (e.g., the batch size) change. If lower noise allows it, as for the Lewis estimate, moving averages should be used in place of moving medians.

6 Related Work

Schütze et al. (2006) studied a Lewis-based performance estimation method in a binary text classification setting. They attribute difficulties in estimating recall to a "missed cluster effect", meaning that the active sampling procedure is failing to select some clusters of relevant training examples in the pool that are too dissimilar to the relevant examples already known. Diversity measures as proposed by (Shen et al., 2004) might help in mitigating this effect, but our experiments show that there are fundamental differences between text classification and NER. Since missed clusters of relevant examples in the training data would eventually be used as we exhaustively label the entire pool, we should see improvements in recall when the missed clusters get used. Instead, we observed in section 2.1, that there are no further performance gains after a certain portion of the pool is labeled. Thus, all examples that the classifier can make use of must have been taken into account, and there appear to be no missed clusters.

Tomanek et al. (2007) present a stopping criterion for query-by-committee-based AL that is based on the rate of disagreement of the classifiers in the committee. While our uncertainty convergence criterion can only be applied to uncertainty sampling, the performance convergence criterion can be used in a committee-based setting.

Li and Sethi (2006) estimate the conditional error as a measure of uncertainty in selection (instead of using it for stopping as we do), using a variable-bin histogram for improving the error estimates. They do not evaluate the quality of the probability estimates. As with our stopping criterion, we expect this selection criterion to be the more effective the more accurate the probability estimates are. We therefore believe that our method of improving probability estimates based on LOO bins could improve their selection criterion.

Stop crit.	ϵ	Peak	Stop		Δ Bl	sd	Δ Pk	sd
1-Entropy threshold	0.01	80.8	3645	12.0%	1.44	0.7	−0.68	0.4
MinMax threshold	0.01	80.8	3133	10.3%	0.11	1.0	−2.0	0.8
Margin threshold	0.01	80.8	3158	10.4%	1.1	0.8	−1.0	0.8
1-Entropy gradient	0.00005	80.8	4572	15.0%	0.97	0.4	−1.1	0.5
MinMax gradient	0.00005	80.8	4397	14.5%	1.02	0.4	−1.1	0.5
Margin gradient	0.00005	80.8	5292	17.5%	0.81	0.3	−1.32	0.4
Lewis grd. (Median)	0.00005	80.8	2791	9.2%	0.8	1.4	−1.3	1.4
Lewis grd. (Mean)	0.00005	80.8	3999	13.1%	1.1	0.8	−0.95	0.6

Table 5: Performance at stopping points (baseline perf. 78.7, Selection: *1-Entropy*)

7 Conclusion and Future Work

In this paper, we presented several criteria to stop the AL process. For stopping the training at a user-defined performance level, we proposed a method for estimating classifier performance in a multiclass classification setting. While we could achieve acceptable accuracy in estimation of precision, we find that recall estimation is hard. Estimation is not accurate enough to assist in making a reliable decision if the performance of the classifier is acceptable for practical use. In the future, we plan to improve on performance estimation quality, e.g., by using the variable-bin approach suggested by Li and Sethi (2006).

Nevertheless, we showed that the gradient of the performance estimate can successfully be used as a stopping criterion relative to the optimal performance that is attainable on a given pool. We also describe stopping criteria based on the gradient of the uncertainty measure of the instances selected for training. The criteria reliably determine stopping points that result in a performance that is better than the supervised baseline and close to the optimal performance. We believe that these criteria can be applied to any AL setting based on uncertainty sampling, not just NER.

If it turns out that the maximum possible performance does not meet a user's expectations, the user needs to acquire fresh data and refill the pool. This might lead to an approach to reduce the computational cost of AL we want to evaluate in future work: Subdivide a large sampling pool into smaller sub-pools, run AL sequentially on the sub-pools. When the stopping criterion is reached, switch to the next sub-pool.

We also found that uncertainty curves of the selected examples are quite noisy. We would like to investigate which properties of the training examples cause these drops in the uncertainty curve.

References

Daume III, Hal. 2007. Frustratingly easy domain adaptation. In *ACL-07*, pages 256–263.

Genkin, A., D.D. Lewis, and D. Madigan. 2007. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304.

Lewis, D.D. and W.A. Gale. 1994. A sequential algorithm for training text classifiers. *ACM SIGIR*.

Lewis, D.D. 1995. Evaluating and optimizing autonomous text classification systems. *ACM SIGIR*.

Li, M. and I.K. Sethi. 2006. Confidence-Based Active Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1251–1261.

Schein, Andrew I. 2005. *Active Learning for Logistic Regression*. Ph.D. thesis, University of Pennsylvania.

Schütze, H., E. Velipasaoglu, and J.O. Pedersen. 2006. Performance thresholding in practical text classification. In *CIKM*, pages 662–671.

Shen, Dan, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *ACL '04*.

Tjong Kim Sang, Erik F. and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Tomanek, Katrin, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *EMNLP-CoNLL*.

Vlachos, Andreas. 2008. A stopping criterion for active learning. *Computer Speech and Language*, 22(3):295–312.

Zhu, J. and E. Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *EMNLP-CoNLL*.