Multi-Criterion Active Learning in Conditional Random Fields

Conference Paper · December 2006
DOI: 10.1109/ICTAI.2006.90 · Source: IEEE Xplore

CITATIONS
11

READS
24

8 authors, including:

Nagiza F Samatova
North Carolina State University
226 PUBLICATIONS  2,214 CITATIONS
SEE PROFILE

Tarik Umar
University of Chicago
2 PUBLICATIONS  11 CITATIONS
SEE PROFILE

David Buttler
Lawrence Livermore National Laboratory
56 PUBLICATIONS  857 CITATIONS
SEE PROFILE

David Hysom
Lawrence Livermore National Laboratory
21 PUBLICATIONS  384 CITATIONS
SEE PROFILE

Available from: David Buttler
Multi-Criterion Active Learning in Conditional Random Fields

Christopher T. Symons, Nagiza F. Samatova, Ramya Krishnamurthy, Byung H. Park, Tarik Umar, David Buttler, Terence Critchlow, and David Hysom

1Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA
2Lawrence Livermore National Laboratory, Livermore, CA 94551 USA

Abstract

Conditional Random Fields (CRFs), which are popular supervised learning models for many Natural Language Processing (NLP) tasks, typically require a large collection of labeled data for training. In practice, however, manual annotation of text documents is quite costly. Furthermore, even large labeled training sets can have arbitrarily limited performance peaks if they are not chosen with care. This paper considers the use of multi-criterion active learning for identification of a small but sufficient set of text samples for training CRFs. Our empirical results demonstrate that our method is capable of reducing the manual annotation costs, while also limiting the retraining costs that are often associated with active learning. In addition, we show that the generalization performance of CRFs can be enhanced through judicious selection of training examples.

1. Introduction

Conditional Random Fields (CRFs) [1], a version of Markov Random Fields [2], have quickly become one of the more popular supervised-learning models in the field of Natural Language Processing (NLP). They are well-suited to handle sequential labeling tasks, such as part-of-speech tagging, shallow parsing, and Named Entity Recognition, outperforming many other popular model types on such problems [1, 3]. In practice, however, there are two major drawbacks to using CRFs for NLP tasks. First and foremost, since CRFs fall under the category of supervised learning, as with all such methodologies, manual annotation costs in terms of building the training data can be prohibitive. This is particularly problematic for NLP tasks, in which applications to new domains, or even just new corpora, almost inevitably require new training data in order to avoid performance drop-offs. The second drawback is that CRFs are well-known to have problems with overfitting [4], a phenomenon in which attempting to reduce training error beyond a certain point can actually lead to reduced generalization ability as well. This particular weakness makes it even more likely that applications to new domains will require retraining on new data.

Active learning, in which the learner selectively chooses the examples that will constitute the hand-labeled training set, can be used to address both of these problems. The objective of active learning is most commonly described as either achievement of similar performance using a substantially reduced amount of training data or improvement of generalization performance on a fixed-size training set. Its viability has been demonstrated repeatedly in the literature via noticeable improvements in generalization performance [5, 6] and/or reductions in annotation requirements [7-9].

In spite of its advantages, active learning can lead to greatly increased computation times due to the costs associated with retraining and active selection. This is an important concern that should not be overlooked. It is particularly problematic for models, such as CRFs, that utilize a global optimization routine for training, since selection based on an updated model requires an expensive retraining procedure. Therefore, such methods often necessitate batch-based selection of examples for retraining. However, even with batch selection, unless the employed active-learning method performs well enough and runs sufficiently quickly, the computational costs for using active learning could easily outweigh the benefit from the reduced manual labor costs.

Thus, the problem becomes one of determining the best criteria for use in sample selection in order to effectively balance the performance of the active learning system with the required processing time. In
this paper, we present a multi-criterion active-learning selection methodology that reduces both the number of manually annotated examples required and the amount of computation time for retraining and selection. Moreover, by demonstrating improved generalization performance when measured against a full-data model, we are able to validate active learning as a viable addition to other known methods [4, 10, 11] for addressing the overfitting problem in CRFs.

The paper is organized as follows. In Section 2, we provide some background material on active learning and CRFs. We then describe our selection methodology in detail in Section 3. Finally, we present experimental results in Section 4, along with conclusions and a discussion of future work in Section 5.

2. Background

2.1. Active learning

There are many variations of active learning, but we will be concerned with pool-based active learning, in which the examples that are to be annotated are selected from a fixed, unlabeled data set. It is common practice in this type of active learning to assume that the distribution of the data in the pool roughly approximates the distribution of the data on which the model will be applied, i.e. the true data distribution. For most NLP applications, selection of a pool for which this assumption is credible is rarely problematic, due to the large amount of unlabeled data that is typically available. Pool-based active learning has been utilized successfully in many applications to limit the amount of labeled training data required to reach an acceptable level of performance [8, 9, 12]. Other work in the field has focused more directly on using active learning to increase generalization performance, with good results [5, 6].

In the active-learning literature, many measures have been utilized to actively select the most valuable samples for use in training, including uncertainty [6, 13], variance in prediction from query-by-committee [14], and risk minimization [15]. Uncertainty-based measures typically utilize some means of evaluating model confidence in order to assign significance to a sample. Thus, this type of measure is usually model-dependent. However, using uncertainty alone is not advisable because it often ignores the distribution of the available data. In other words, there is the danger of selecting outliers and degrading generalization performance due to overfitting of data that does not represent the real distribution.

For this reason, several methodologies have emerged that take into account the distribution of the unlabeled data. Freund et al. [14] apply a query-by-committee algorithm to selective sampling on certain concepts that are “dense-in-themselves,” and prove an exponential improvement in the sample complexity. McCallum and Nigam [16] apply active learning with Expectation Maximization on a text classification task. That work employs information concerning the representativeness of a document, which allows consideration of the data distribution.

In many practical cases, active learning is performed in batch. In other words, instead of a single example being chosen at each round of active learning, examples are selected as a batch in order to limit retraining costs. Thus, some methods that attempt to choose a set of diverse examples have been proposed. Brinker [7] and Shen et al. [9], seek to include a diverse sample by using geometrical distance computable in the Euclidean space of Support Vector Machines (SVMs). Sassano [8] reports that diverse batch selection is possible with a two-pool method when it is applied to Japanese word segmentation.

It should be emphasized that efforts to incorporate the data distribution in sample selection procedures have repeatedly been shown to be important for optimally increasing generalization performance, and it has been empirically demonstrated that such efforts result in improved outcomes [5, 6, 14, 15]. In addition, some specific formal problems have been shown to have strong theoretical guarantees in this regard [14, 17].

To the best of our knowledge, there is only one work applying active learning in Conditional Random Fields [18]. This work utilizes model uncertainty alone (i.e. uses a single-criterion approach) to identify potentially useful samples in a structured prediction task. It should be noted that the main goal of that work is not to optimize selection in the same terms that we are concerned with, but rather to reduce the difficulty of annotation, which is highly relevant to structured prediction tasks, as opposed to the more basic sequential learning tasks that we are addressing.

Our work leverages and attempts to improve upon some of the above-mentioned advancements made by others in combining selection criteria [8, 9] in order to fashion a multi-criterion active learning approach that can be practically applied to CRFs.

2.2. Conditional random fields

Conditional Random Fields [1] are log-linear graphical models that typically use a global optimization routine to maximize the conditional log-likelihood of the model parameters given the available
training data. As with other graphical models, they restrict the choice of a probability distribution to one that matches the independence assumptions explicit in the structure of a graph. In such a graph, the nodes are random variables, such as the positions in a sequence to which states (labels) will be assigned, and the edges represent the dependencies between nodes. Thus, the lack of an edge indicates a conditional independence between the concerned nodes.

Since CRFs are undirected graphical models (Markov Random Fields), they allow a factorization of the probability distribution into potential functions defined over cliques (completely connected subgraphs) in the graph. When dealing with sequentially structured data, as we do here, the CRF can take on the form of a linear chain or path. Therefore, the maximal cliques are restricted to being of size 2, which allows efficient calculation of the partition function (normalization factor) for these models. This is important since CRFs use a global, as opposed to per-state, normalization [1].

Thus, in a linear-chain CRF the probability of a state sequence, \( s \), given an observation sequence, \( o \), is defined as

\[
P(s | o) = \frac{1}{Z_o} \exp \left( \sum_{i=1}^{N} \sum_{k} \lambda_k f_k(s_{i-1}, s_i, o, i) \right)
\]  

(1)

where \( f_k \) are feature functions, \( \lambda_k \) are their associated weights in the model, \( i \) is a position in the sequence, and \( Z_o \) is the partition function, which is calculated by summing the scores of every possible state-sequence assignment [10].

CRFs are discriminative; i.e. they do not attempt to model the observations, as a generative model would. Therefore, it is not necessary to make the type of independence assumptions about the features used in the model that most generative models would be forced to make to avoid intractability. This allows a great deal of freedom when selecting features for use in model building.

The most popular training routine for CRFs is the limited-memory quasi-Newton BFGS (L-BFGS) algorithm. This numerical method for unconstrained optimization has been shown to converge much more quickly than other known methods when globally setting the parameters (i.e. the \( \lambda_k \) values from equation 1) in CRF training [3]. Convergence to the global maximum of the likelihood function is guaranteed.

Application of a learned model to unlabeled samples is usually performed via dynamic programming in the form of the Viterbi algorithm. This method allows efficient selection of the most probable sequence of labels for assignment to any input sequence. CRFs have been shown to outperform other commonly utilized graphical models, such as Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs), on several sequential natural-language tasks, such as part-of-speech tagging [1].

3. Multi-criterion active learning in CRFs

Our active learning framework takes a multi-criterion approach to sample selection by combining: (a) the uncertainty of the model (section 3.1.1); (b) the representativeness of a sample (section 3.2.2); and (c) the diversity of each batch (section 3.3). It is designed to choose a set of non-redundant unlabeled examples of high model uncertainty, while effectively minimizing the likelihood that problematic or irrelevant examples (outliers) will be included. This section describes our approach to realizing this type of multi-criterion active learning within the context of CRFs.

3.1. Model uncertainty

Intuitively, supplementing the training data with examples that are difficult to classify can potentially help enrich the classifier more than the addition of “easy” examples. We anticipate that such useful examples can strengthen the model’s knowledge concerning ambiguous features. Traditionally, such examples are often defined in terms of the degree of uncertainty given by the classifier or a group of classifiers.

3.1.1. Uncertainty measure

We utilize the probabilistic confidence of the CRF model to assign the degree of uncertainty to an example. Specifically, we begin our approach by basing uncertainty on the score utilized by Culotta and McCallum [18]. This is a confidence measure for CRFs, described in [19], that gives the probability of a labeled field using a constrained version of the forward-backward algorithm. To be more explicit, the method accumulates the weights of all possible label sequences that match a labeled field assigned by the model, and then normalizes this accumulated weight by the full partition function to obtain a probabilistic confidence score. For example, in the Viterbi lattice of state paths shown in Figure 1, the weight of all paths in the upper lattice divided by the weight of all paths (the lower lattice) gives the probability (and thus model confidence) of the field consisting of states \( s_1, s_2, \) and \( s_3 \) at positions \( i_3, i_4, \) and \( i_5 \), respectively.
As a measure for confidence in a field, this constrained forward-backward (CFB) score has been shown to have a high correlation with the correctness of the labeling for a field [19], making this measure more desirable than standard gamma scores [20]. So, in essence, we consider a sample to be useful if it has a much higher than average probability of receiving the wrong label assignment. Furthermore, for such examples, we anticipate that small modifications to the model’s parameters are more likely to change the outcome of the Viterbi algorithm.

We use a sentence as our unit for selection and annotation. Thus, the degree of uncertainty is assigned to a sentence rather than to individual fields. Our assumption is that it is easier for a person to annotate a complete sentence than a stripped out text region with minimal context. Consequently, we assign an uncertainty score to a sentence based on the minimum confidence of the model for any field. Formally, we set the uncertainty score for a sentence, $S$, to be $1 - C_F$, where $C_F$ is the lowest confidence score for any field in the sentence, $\text{Uncertainty}(S) = 1 - C_F$.

### 3.1.2. Efficient computation of uncertainty

A concern with using the measure as described in [19] is the need to calculate the path scores for all possible state assignments passing through a constrained field for each field in a sequence, in addition to calculating the partition function $Z_0$. This would essentially mean performing the dynamic programming algorithm once for each field. This is not necessarily a problem for certain applications in which one only needs the scores for positively labeled fields, which may occur at a relatively low rate. However, since we wish to consider the confidence even for fields that receive no label (i.e. a label of OTHER, which can be thought of as indicating a negative example for the class being trained), the complexity of the required computations is essentially quadratic with respect to both the average sequence length and the number of model states. Therefore, we devised an efficient procedure through which computation of the confidences of all fields is not prohibitively expensive.

The following methodology allows us to calculate the confidence, $C_F$, for each field (which equates to the majority of all sequence positions) efficiently. Since the confidence is based on a constrained version of the forward-backward algorithm, it simplifies the explanation to note that the Viterbi algorithm is defined as

$$\delta_{i+1}(s_y) = \max_{s'} \delta_i(s') \exp \left( \sum_k \lambda_k f_k(s', s_y, o, i) \right)$$

while the forward algorithm is defined as

$$\alpha_{i+1}(s_y) = \sum_s \alpha_i(s') \exp \left( \sum_k \lambda_k f_k(s', s_y, o, i) \right)$$

[19]. In the above equations, $s_i$ indicates a model state that can be assigned at position $i$, and $s'$ represents any of the states that the model can assume at the previous position. Thus, we only need to add a small number of operations to the Viterbi algorithm in order to calculate and store the values from both procedures at the same time. At that point, we already have the full partition function $Z_\alpha$, which is obtained by summing the forward ($\alpha$) values at the final position in the sequence, and more importantly we have enough information to eliminate the vast majority of the calculations required to find the confidence score for any field. To see this, assume that we have a given start position, $i$, and end position, $j$, for a field. Then, we can apply the following formula to obtain the confidence for a field, $C_F$.

$$\alpha_j(s_y) \left( \prod_{i+1} \frac{\delta_i(s_y)}{\delta_i(s)} \right) \sum_{s'} \frac{\delta_{j+2}(s')}{\delta_j(s)} \exp \left( \sum_k \lambda_k f_k(s', s_{j+2}, o, j + 2) \right)$$

where $Z_0' = \sum_s \alpha_{j+2}(s')$. If $j+2$ does not exist, then the exponentials are not needed and the divisor is summed over the final position in the sequence. Note that all of the $\alpha$ and $\delta$ values were previously stored during the original Viterbi decoding. Furthermore, note that it would also be possible to apply a similar method in which one could perform dynamic programming in both directions once, and then use the backward ($\beta$) values and the final partition function instead of the final exponentials and the sum of $\alpha$ values, respectively.
3.2. Sample representativeness

Now, it is important to keep in mind that given a fixed amount of data, preferably small, we hope to improve the generalization performance of the trained model by optimally selecting the training samples. If we consider this selection problem in the same terms as Cohn et al. [5], then we want to reduce the region of uncertainty of our model as much as possible when selecting new data for manual annotation. Therefore, whenever possible, we hope to avoid the inclusion of irrelevant or redundant examples. Using the uncertainty measure from the previous section should ensure that the examples lie in the current region of model uncertainty and help us avoid the selection of samples that are redundant in terms of the current classifier.

However, the inclusion of unrepresentative data can lead to problems in achieving optimal generalization performance, particularly since CRFs trained via maximum-likelihood methods tend to overfit the parameters to the training data. Therefore, in order to avoid choosing largely irrelevant data, we need to take into account the distribution of the unlabeled data. Thus, we combine a measure for the representativeness of a sentence with the score for its uncertainty in order to select the most useful data. Utilizing representativeness allows us to keep in mind the data distribution, which, as mentioned above, has been shown to be an important consideration in active learning. Our measure for representativeness is defined as the average sample similarity in terms of activated model features, as explained below.

3.2.1. Sample similarity in CRFs

For our purposes, the similarity between two sentences is measured by aligning their predicted label sequences around the lowest confidence fields. The comparisons are performed using these regions of uncertainty in the sentences, rather than the whole sentences, for two main reasons. First, despite returning an entire sentence for ease of labeling, we are choosing sentences based on the usefulness of these identified subregions for training subsequent models. Therefore, even if two sentences have similar regions, if these are not the “useful” regions, then we do not wish to consider them as being similar. Second, comparing entire sentences would be more costly in terms of computation time.

Let \( S_i = <p_{i1}, p_{i2}, ..., p_{im}> \) and \( S_j = <p_{j1}, p_{j2}, ..., p_{jn}> \) be subsequences of sentences \( i \) and \( j \), with \( m \) and \( n \) labeled positions, respectively. These subsequences comprise the lowest confidence fields and their surrounding contexts up to three positions away. This limit still allows us to include the most distant positions that affect the field confidence. Then, we compare the similarities between the positions of \( S_i \) and \( S_j \).

The similarity between any two sequence positions is computed by comparing the features of the CRF that are activated at each position. More specifically, we consider the model’s binary feature vectors at the positions in the sequences, where the values in a vector correspond to the results of the evaluation of the feature functions that are used in the inner summation of equation (1). The similarity between two positions, e.g. \( \text{Sim}(p_1, p_2) \), is then calculated by applying the oft-used cosine-similarity score between the binary feature vectors, \( w_{p_1} \) and \( w_{p_2} \), at each position.

\[
\text{Sim}(p_1, p_2) = \frac{w_{p_1} \cdot w_{p_2}}{\|w_{p_1}\| \cdot \|w_{p_2}\|}
\]

After transforming the positional similarity scores into disparity scores by subtracting them from 1, the best alignment of \( S_i \) and \( S_j \) is then found by using the well-known dynamic time warping algorithm (see Rabiner [21] for details). In other words, dynamic programming is performed to find the alignment that minimizes the cumulative disparity between the two sequences, \( S_i \) and \( S_j \). The cumulative disparity is the sum of the positional disparities in any alignment of the sequences normalized by the length of the longer sequence, i.e. \( \text{max}(m,n) \). If we denote this smallest cumulative disparity value as \( d \), then the similarity between the sentences is defined as

\[
\text{Sim}(S_i, S_j) = 1 - d
\]

A similar approach applied to SVMs [9] ignores all of the sequences and contexts in which the model does not choose to label an entity, even though this may constitute a substantial number of all useful examples. This is potentially problematic in the early stages of model building or for classes of problems that have rarely occurring positive examples. Therefore, rather than using the assumed entities and their contextual regions, we compare the similarity of two sentences based on the lowest confidence fields (which can have the non-entity (OTHER) label) and their surrounding regions.

3.2.2. Representativeness measure

First, we select a subset of candidate sentences that includes the sentences of highest uncertainty, as defined in section 3.1. This set of candidate examples is then used for evaluating the representativeness of each sample in the set. For each sentence, the representativeness is assigned based on the average pair-wise similarity with all other candidates, where pair-wise similarity is calculated as described in section 3.2.1. Thus,
\[
\text{Representativeness}(S) = \frac{1}{N-1} \sum_{i=1 \atop i \neq j}^{N} \text{Sim}(S_i, S_j)
\]

Then, the sentences are sorted based on their combined scores, which are calculated using the following weighted formula (as in [9]):
\[0.6*\text{Uncertainty}(S) + 0.4*\text{Representativeness}(S)\]

### 3.3. Diversity measure

Since the expense of retraining can be prohibitive when using L-BFGS or any other method for globally maximizing the likelihood of the parameters, we need to use batch selection of training samples to limit the retraining costs. For this, we wish to optimize the utility of each new batch added to the training set by avoiding the addition of redundant examples in a single round. Therefore, we also incorporate a diversity measure into our batch selection.

Once the sentences with the highest combined scores are identified, as described in Section 3.2.2, we then attempt to select a set of the most representative sentences that covers the widest possible region of uncertainty. For this diversity consideration, we utilize the local diversity measurement described by Shen et al. [9]. Specifically, from the list of candidate sentences, which have been sorted according to their combined scores, we add selections to our batch in order as long as the pair-wise similarity with any of the sentences currently contained in the batch is no greater than a threshold value. The pair-wise similarity is calculated using the same procedure as the one used for sequence similarity in the calculation of representativeness, see section 3.2.1.

One suggested measure for this threshold value is the average similarity of all the samples in the batch [9]. However, this omits inclusion of some of the most useful samples, even when a large portion of their activated features are distinct from those of any of the examples in the current batch. Therefore, we use a higher threshold, \((1+\text{avgSim})/2\), for our diversity consideration, where \(\text{avgSim}\) is the average pair-wise similarity among all samples previously added to the batch. Thus, the samples are added to the batch from the candidate set in decreasing order of their combined scores as long as the pair-wise similarity with each member of the current batch is less than or equal to this threshold.

### 3.4. Two-pool selection for efficiency

Trying to keep the data distribution in perspective while limiting the number of similarity comparisons can be problematic. Therefore, since the calculation of all pair-wise similarity scores is computationally intensive, we employ a two-pool method for sample selection [8]. That is to say, rather than performing evaluation of all possible unlabeled examples, we randomly select a smaller pool at each iteration from which we make our ultimate selections. We assume that these samples are selected independently and identically distributed (i.i.d.), thus allowing us to maintain the same distribution. This method is particularly relevant with the availability of extremely large unlabeled corpora for NLP tasks. It also offers an intuitive appeal in that selection of outliers becomes less likely and the diversity of a batch can also be positively affected.

In summary, at each iteration of active learning we perform the following tasks: (1) Independently select 1000 samples at random from the unlabeled data set; (2) Determine the uncertainty scores of these 1000 samples using the latest model; (3) Choose the 250 most uncertain samples for our candidate set; (4) Compute the representativeness score for all selected samples; (5) Calculate the combined score for each sample; and (6) Select a batch of disparate samples from the candidate set by utilizing the combined scores and the diversity threshold.

### 4. Experimental results

We evaluate our methodology on a Named Entity Recognition (NER) task using data from the CoNLL 2003 shared task [22].

#### 4.1. Model construction

We train separate models for each entity type, person, location, and organization, utilizing most of the features that are commonly applied for NER tasks in a window of size 5 around each label position. These include standard regular expression features, such as ALL\_CAPS and CONTAINS\_HYPHEN, word features, edge (label-transition) features, several small lexicons, and a few specialized features for each type. However, we do not use any of the common grammatical features, such as part-of-speech tags and phrasal chunk tags, in this evaluation, and no form of feature selection is employed. Furthermore, these results do not include performance evaluation after combining the results of the individual models, through which we would typically expect to see improved performance, due to the common occurrence of overlap among difficult to classify examples for different types.
4.2. Overall performance

Figure 2 is a plot showing a performance comparison of random sampling, uncertainty-based active learning (single-criterion selection using the uncertainty measure and a single pool), and our multi-criterion approach described above. The results represent the average of the F1 scores across all three of the entity models. The initial random sample used to begin training is 100 sentences, and the batch size (number of sentences added at each iteration) is 50.

![Figure 2](image)

Figure 2. Comparative results for all entities using a batch size of 50 (F-Measure is the average score of all 3 entity types).

After 3 rounds of active learning (and up to 22 rounds (1150 sentences), at which point the averaged performance begins to plateau), the F-scores for the models obtained through our two-pool, multi-criterion approach exceed those of the single-criterion approach (using model uncertainty) by 1.69 points on average, and exceed those of random selection by 5.12 points on average.

The results displayed in Figure 3 (for the person entity) demonstrate that, in addition to reducing the manual annotation requirements, we are also able to limit the overall computation time by utilizing a multi-criterion approach that includes two-pool selection. Specifically, the F-Measure for the person entity when using all of the data (8,646 sentences) is 77.56. This same performance can be reached with a much smaller amount of data by applying active learning or even random sampling. However, while random sampling requires 1,750 sentences to reach this level of performance, uncertainty-based active learning (i.e. the single-criterion approach) only requires 1100 sentences (1000 when the single-criterion selection is combined with two-pool selection). Furthermore, by means of our two-pool, multi-criterion sample selection, we are able to reduce this requirement to only 750 example sentences.

![Figure 3](image)

Figure 3. Comparative results for achieving full-data performance (F1-Score =77.56) for person models. (a) Number of annotated sentences required. (b) Total computation time (on a single 1.7GHz Pentium 4 processor) including all selection and re-training.

The single-pool, multi-criterion results shown in Figure 3 require a larger candidate set size (500) in order to allow a meaningful measure of representativeness for the samples. While it is possible to outperform uncertainty-based selection in terms of the annotation requirements when employing this method, the computational costs become too great. In fact, it is infeasible to increase the candidate set size much beyond this level and still supply a usable method. However, the complete multi-criterion method that includes two-pool selection is able to reduce both the annotation requirements and the computational costs, only requiring 61.1 hours of total computation time.

Our experimental results also confirmed our intuition regarding the diversity threshold. In other words, utilizing average similarity to restrict additions to a batch gives poor results in our CRF framework,
likely due to the exclusion of many high-uncertainty samples that would have affected features not influenced by the supposedly more diverse batch. Thus, while the higher threshold described above allows us to outperform a single-criterion method, the use of the lower threshold (i.e. average similarity of all samples currently in the batch) does not.

In addition to allowing a reduction in labeling effort, we are able to increase the generalization performance as compared with a model built using all of the data. The results in Table 1 indicate that improvements in F-Measure can be achieved through this methodology, and this can be accomplished while still significantly reducing the annotation requirements.

Table 1: F-Score improvement achieved via active learning over using all data for model.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Method</th>
<th>No. of Sentences</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>All Data</td>
<td>8,646</td>
<td>77.56</td>
</tr>
<tr>
<td></td>
<td>Multi-Criterion</td>
<td>1,150</td>
<td>80.03</td>
</tr>
<tr>
<td></td>
<td>Active Learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>All Data</td>
<td>8,646</td>
<td>88.19</td>
</tr>
<tr>
<td></td>
<td>Multi-Criterion</td>
<td>3,050</td>
<td>88.51</td>
</tr>
<tr>
<td></td>
<td>Active Learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>All Data</td>
<td>8,646</td>
<td>59.52</td>
</tr>
<tr>
<td></td>
<td>Multi-Criterion</td>
<td>2,500</td>
<td>60.05</td>
</tr>
<tr>
<td></td>
<td>Active Learning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The results in Table 1 simply help to confirm that large data sets often contain subsets that are more ideally suited to increasing generalization performance when used for training. In other words, it is less dangerous to tightly fit the model parameters to a collection of examples that contains fewer rare or unlikely samples. Of course, it is also possible to surpass the performance of the full-data model using some subset selected via single-criterion or random selection, but the multi-criterion approach can identify such a subset more quickly by taking a more principled approach. In fact, as demonstrated by the results in Figure 2, the performance increases in a much more monotonic fashion when the data distribution is considered.

5. Conclusions and future work

We described a multi-criterion approach to active learning in CRFs and established that this method has the potential to allow reductions in annotation requirements, over and above what can be achieved when using model uncertainty alone. This is particularly encouraging given the difficulty of obtaining new training data when applying models to new concepts and domains in NLP tasks. Moreover, we were able to demonstrate that we can obtain this improvement while providing reductions in the overall computational costs involved.

In addition, we showed that it is possible to improve the generalization performance of trained CRF models by choosing the training data set selectively. Thus, active learning offers another potential method for dealing with overfitting in CRFs that should be able to enhance the results of other known methods for doing so. It remains to be seen how effective active learning can be when combined with other methods for coping with overfitting in CRFs.

Furthermore, while it was beyond the scope of this work to exhaustively study convergence criteria for CRF training, the lack of an ideal stopping criterion tempers the results we received to some degree. Specifically, it should be noted that convergence criteria for L-BFGS optimization of CRF models are not well defined. For example, the norm of the gradient is often utilized, but it is known to be unreliable due to a tendency to oscillate as it approaches the target optimum [23]. Further exploration of convergence criteria for CRFs in general, and an investigation of ideal stopping criteria during active learning in particular, would be welcome.

Finally, we should mention that while feature selection is an important task in model building, integration of such techniques was outside of the focus of this paper. Since we did not incorporate any form of feature selection in our experiments, our performance numbers are negatively affected as a result. While we do not believe this alters the legitimacy of our conclusions, it is unclear whether various feature selection techniques would be more or less effective when applied with a training set that was actively selected.

6. Acknowledgements

This research was sponsored by the U.S. Department of Homeland Security’s Advanced Scientific Computing Program. This work was performed under the auspices of the U.S. Department of Energy by Oak Ridge National Laboratory, which is managed by the University of Tennessee-Battelle, L.L.C. for the Department of Energy under contract DOE-AC05-00OR22725, and by the University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. UCRL-CONF-222475. In addition, a portion of this research was performed under contract number DE-AC05-
00OR22750 between the U.S. Department of Energy and Oak Ridge Associated Universities.

Our CRF implementation was built upon the open-source SourceForge CRF project codes (http://crf.sourceforge.net). We would like to thank the authors for making this implementation publicly available.

References


