Confidence in Structured-Prediction using Confidence-Weighted Models

Avihai Mejer
Department of Computer Science
Technion-Israel Institute of Technology
Haifa 32000, Israel
amejer@tx.technion.ac.il

Koby Crammer
Department of Electrical Engineering
Technion-Israel Institute of Technology
Haifa 32000, Israel
koby@ee.technion.ac.il

Abstract

Confidence-Weighted linear classifiers (CW) and its successors were shown to perform well on binary and multiclass NLP problems. In this paper we extend the CW approach for sequence learning and show that it achieves state-of-the-art performance on four noun phrase chucking and named entity recognition tasks. We then derive few algorithmic approaches to estimate the prediction's correctness of each label in the output sequence. We show that our approach provides a reliable relative correctness information as it outperforms other alternatives in ranking label-predictions according to their error. We also show empirically that our methods output close to absolute estimation of error. Finally, we show how to use this information to improve active learning.

1 Introduction

In the past decade structured classification has seen much interest by the machine learning community. After the introduction of conditional random fields (CRFs) (Lafferty et al., 2001), and maximum margin Markov networks (Taskar et al., 2003), which are batch algorithms, new online method were introduced. For example the passive-aggressive algorithm was adapted to chunking (Shimizu and Haas, 2006), parsing (McDonald et al., 2005b), learning preferences (Wick et al., 2009) and text segmentation (McDonald et al., 2005a). These new online algorithms are fast to train and simple to implement, yet they generate models that output merely a prediction with no additional information, as opposed to probabilistic models like CRFs or HMMs.

In this work we fill this gap proposing few alternatives to compute confidence in the output of discriminative non-probabilistic algorithms. As before, our algorithms output the highest-scoring labeling. However, they also compute additional labelings, that are used to compute the *per word* confidence in its labelings. We build on the recently introduced confidence-weighted learning (Dredze et al., 2008; Crammer et al., 2009b) and induce a distribution over labelings from the distribution maintained over weight-vectors.

We show how to compute confidence estimates in the label predicted per word, such that the confidence reflects the probability that the label is not correct. We then use this confidence information to rank all labeled words (in all sentences). This can be thought of as a retrieval of the erroneous words, which can than be passed to human annotator for an examination, either to correct these mistakes or as a quality control component. Next, we show how to apply our techniques to active learning over sequences. We evaluate our methods on four NP chunking and NER datasets and demonstrate the usefulness of our methods. Finally, we report the performance of obtained by CW-like adapted to sequence prediction, which are comparable with current state-of-the-art algorithms.

2 Confidence-Weighted Learning

Consider the following online binary classification problem that proceeds in rounds. On the ith round the online algorithm receives an input $\boldsymbol{x}_i \in \mathbb{R}^d$ and

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 971–981,
MIT, Massachusetts, USA, 9-11 October 2010.

applies its current rule to make a prediction $\hat{y}_i \in \mathcal{Y}$, for the binary set $\mathcal{Y} = \{-1, +1\}$. It then receives the correct label $y_i \in \mathcal{Y}$ and suffers a loss $\ell(y_i, \hat{y}_i)$. At this point, the algorithm updates its prediction rule with the pair $(\boldsymbol{x}_i, y_i)$ and proceeds to the next round. A summary of online algorithms can be found in (Cesa-Bianchi and Lugosi, 2006).

Online confidence-weighted (CW) learning (Dredze et al., 2008; Crammer et al., 2008), generalized the passive-aggressive (PA) update principle to multivariate Gaussian distributions over the weight vectors - $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ - for binary classification. The mean $\boldsymbol{\mu} \in \mathbb{R}^d$ contains the current estimate for the best weight vector, whereas the Gaussian covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ captures the confidence in this estimate. More precisely, the diagonal elements $\Sigma_{p,p,}$ capture the confidence in the value of the corresponding weight $\boldsymbol{\mu}_p$; the smaller the value of $\Sigma_{p,p,}$ is, the more confident is the model in the value of $\boldsymbol{\mu}_p$. The off-diagonal elements $\Sigma_{p,q}$ for $p \neq q$ capture the correlation between the values of $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$. When the data is of large dimension, such as in natural language processing, a model that maintains a full covariance matrix is not feasible and we back-off to diagonal covariance matrices.

CW classifiers are trained according to a PA rule that is modified to track differences in Gaussian distributions. At each round, the new mean and covariance of the weight vector distribution is chosen to be the solucion of an optimization problem (see (Crammer et al., 2008) for details). This particular CW rule may over-fit by construction. A more recent alternative scheme called AROW (adaptive regularization of weight-vectors) (Crammer et al., 2009b) replaces the guaranteed prediction at each round with the a more relaxed objective (see (Crammer et al., 2009b)). AROW has been shown to perform well in practice, especially for noisy data where CW severely overfits.

The solution for the updates of CW and AROW share the same general form,

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \Sigma_i y_i \boldsymbol{x}_i \; ; \; \Sigma_{i+1}^{-1} = \Sigma_{i+1}^{-1} + \beta_i \boldsymbol{x}_i \boldsymbol{x}_i^\top \; , \quad (1)$$

where the difference between CW and AROW is the specific instance-dependent rule used to set the values of α_i and β_i.

Algorithm 1 Sequence Labeling CW/AROW

Input: Joint feature mapping $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$
Initial variance $a > 0$
Tradeoff Parameter $r > 0$ (AROW)
or Confidence parameter ϕ (CW)
Initialize: $\boldsymbol{\mu}_0 = \mathbf{0}$, $\Sigma_0 = aI$
for $i = 1, 2 \ldots$ **do**
Get $\boldsymbol{x}_i \in \mathcal{X}$
Predict best labeling
$\hat{\boldsymbol{y}}_i = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu}_{i-1} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{z})$
Get correct labeling $\boldsymbol{y}_i \in \mathcal{Y}^{|\boldsymbol{x}_i|}$
Define $\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}_i) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}_i)$
Compute α_i and β_i (Eq. (3) for CW ; Eqs. (4),$\beta_i = 1/r$) for AROW)
Set $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + \alpha_i \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}$
Set $\Sigma_i^{-1} = \Sigma_{i-1}^{-1} + \beta_i \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top$
end for

3 Sequence Labeling

In the sequence labeling setting, instances $\boldsymbol{x}$ belong to a general input space $\mathcal{X}$ and conceptually are composed of a finite number n of components, such as words of a sentence. The number of components $n = |\boldsymbol{x}|$ varies between instances. Each part of an instance is labelled from a finite set $\mathcal{Y}$, $|\mathcal{Y}| = K$. That is, a labeling of an entire instance belongs to the product set $\boldsymbol{y} \in \mathcal{Y} \times \mathcal{Y} \ldots \mathcal{Y}$ (n times).

We employ a general approach (Collins, 2002; Crammer et al., 2009a) to generalize binary classification and use a joined feature mapping of an instance $\boldsymbol{x}$ and a labeling $\boldsymbol{y}$ into a common vector space, $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$.

Given an input instance $\boldsymbol{x}$ and a model $\boldsymbol{\mu} \in \mathbb{R}^d$ we predict the labeling with the highest score, $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$. A brute-force approach evaluates the value of the score $\boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$ for each possible labeling $\boldsymbol{z} \in \mathcal{Y}^n$, which is not feasible for large values of n. Instead, we follow standard factorization and restrict the joint mapping to be of the form, $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{n} \boldsymbol{\Phi}(\boldsymbol{x}, y_p) + \sum_{q=2}^{n} \boldsymbol{\Phi}(\boldsymbol{x}, y_q, y_{q-1})$. That is, the mapping is a sum of mappings, each taking into consideration only a label of a single part, or two consecutive parts. The time required to compute the $\max$ operator is linear in n and quadratic in K using the dynamic-programming Viterbi algorithm.

After the algorithm made a prediction, it uses

the current labeled instance $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to update the model. We now define the update rule both for a version of CW and for AROW for strucutred learning, staring with CW. Given the input parameter ϕ of CW we denote by $\phi' = 1 + \phi^2/2, \phi'' = 1 + \phi^2$. We follow a similar argument as in the single update of (Crammer et al., 2009a, sec. 5.1) to sequence labeling by a reduction to binary classification. We first define the difference between the feature vector associated with the current labeling $\boldsymbol{y}_i$ and the feature vector associated with some labeling $\boldsymbol{z}$ to be, $\boldsymbol{\Delta}_{i,\boldsymbol{y},\boldsymbol{z}} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}_i) - \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$, and in particular, when we use the prediction $\hat{\boldsymbol{y}}_i$ we get, $\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}_i) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}_i)$. The CW update is,

$$\begin{aligned} \boldsymbol{\mu}_i &= \boldsymbol{\mu}_{i-1} + \alpha_i \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \\ \Sigma_i^{-1} &= \Sigma_{i-1}^{-1} + \beta_i \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top , \end{aligned} \quad (2)$$

where the two scalars α_i and β_i are set using the update rule defined by (Crammer et al., 2008) for binary classification,

$$v_i = \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \ , \ m_i = \boldsymbol{\mu}_{i-1} \cdot \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \quad (3)$$

$$\alpha_i = \max \left\{ 0, \frac{1}{v_i \phi''} \left(-m_i \phi' + \sqrt{m_i^2 \frac{\phi^4}{4} + v_i \phi^2 \phi''} \right) \right\}$$

$$\beta_i = \frac{\alpha_i \phi}{\sqrt{v_i^+}} \quad , \quad v_i^+ = \frac{1}{4} \left(-\alpha_i v_i \phi + \sqrt{\alpha_i^2 v_i^2 \phi^2 + 4 v_i} \right)^2$$

We turn our attention and describe a modification of AROW for sequence prediction. Replacing the binary-hinge loss in (Crammer et al., 2009b, Eqs. (1,2)) the first one with the corresponding multi-class hinge loss for structured problems we obtain, $\frac{1}{2} (\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top \Sigma_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) + \frac{1}{2r} (\max \{0, \max_{\boldsymbol{z} \neq \boldsymbol{y}} \{d(\boldsymbol{y}, \boldsymbol{z}) - \boldsymbol{\mu} \cdot (\boldsymbol{\Delta}_{i,\boldsymbol{y},\boldsymbol{z}})\}\})^2$, where, $d(\boldsymbol{y}, \boldsymbol{z}) = \sum_{q=1}^{|\boldsymbol{x}|} 1_{y_q \neq z_q}$, is the hamming distance between the two label sequences $\boldsymbol{y}$ and $\boldsymbol{z}$. The last equation is hard to optimize since the max operator is enumerating over exponential number of possible labellings $\boldsymbol{z}$. We thus approximate the enumeration over all possible $\boldsymbol{z}$ with the predicted label sequence $\hat{y}_i$ and get, $\frac{1}{2} (\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top \Sigma_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) + \frac{1}{2r} \left(\max \left\{ 0, d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) - \boldsymbol{\mu} \cdot \left(\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \right) \right\} \right)^2$. Computing the optimal value of the last equation we get an update of the form of the first equation of Eq. (2) where

$$\alpha_i = \frac{\max \left\{ 0, d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) - \boldsymbol{\mu}_{i-1} \cdot \left(\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \right) \right\}}{r + \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}} . \quad (4)$$

Dataset	Sentences	Words	Features
NP chunking	11K	259K	1.35M
NER English	17.5K	250K	1.76M
NER Spanish	10.2K	317.6K	1.85M
NER Dutch	21K	271.5K	1.76M

Table 1: Properties of datasets.

	AROW	CW	5-best PA	Perceptron
NP chunking	0.946	0.947	0.946	**0.944
NER English	0.878	0.877	* 0.870	* 0.862
NER Dutch	0.791	0.787	0.784	* 0.761
NER Spanish	0.775	0.774	0.773	* 0.756

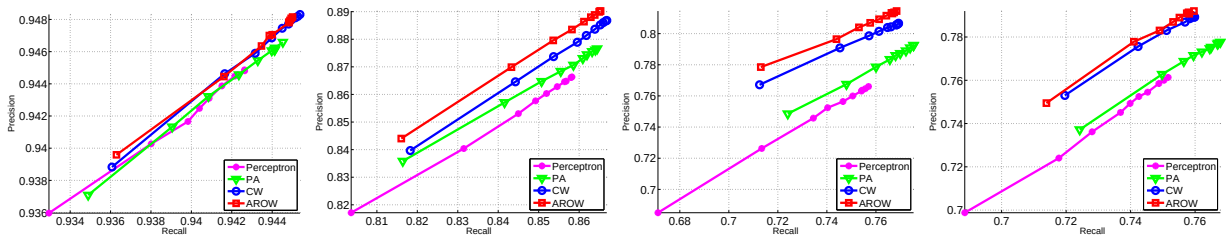
Table 2: Averaged F-measure of methods. Statistical significance (t-test) are with respect to AROW, where * indicates 0.001 and ** indicates 0.01

We proceed with the confidence paramters in (Crammer et al., 2009b, Eqs. (1,2)), which takes into considiration the change of confidence due to the update. The effective features vector that is used to update the mean parameters is $\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}$, and thus the structured update is, $\frac{1}{2} \log \left(\frac{\det \Sigma_i}{\det \Sigma} \right) + \frac{1}{2} \mathrm{Tr} \left(\Sigma_{i-1}^{-1} \Sigma \right) + \frac{1}{2r} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top \Sigma \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}$. Solving the above equation we get an update of the form of the second term of Eq. (2) where $\beta_i = \frac{1}{r}$. The pseudo-code of CW and AROW for sequence problems appears in Alg. 1.

4 Evaluation

For the experiments described in this paper we used four large sequential classification datasets taken from the CoNLL-2000, 2002 and 2003 shared tasks: noun-phrase (NP) chunking (Kim et al., 2000), and named-entity recognition (NER) in Spanish, Dutch (Tjong and Sang, 2002) and English (Tjong et al., 2003). The properties of the four datasets are summarized in Table 1. We followed the feature generation process of (Sha and Pereira, 2003).

Although our primary goal is estimating confidence in prediction and not the actual performance itself, we first report the results of using AROW and CW for sequence learning. We compared the performance CW and AROW of Alg. 1 with two standard online baseline algorithms: Averaged-Perceptron algorithm and 5-best PA (the value of five was shown to be optimal for various tasks (Crammer et al., 2005)). The update rule described in Alg. 1 assumes a full covariance matrix, which is not feasible in our



(a) NP Chunking (b) NER English (c) NER Dutch (d) NER Spanish

Figure 1: Precision and Recall on four datasets (four panels). Each connected set of ten points corresponds to the performance of a specific algorithm after each of the 10 iterations, increasing from bottom-left to top-right.

		Prec	Recall	F-meas	% Err
NP chunking	CW	0.945	0.942	0.943	2.34%
	CRF	0.938	0.934	0.936	2.66%
NER English	CW	0.838	0.826	0.832	3.38%
	CRF	0.823	0.820	0.822	3.53%
NER Dutch	CW	0.803	0.755	0.778	2.05%
	CRF	0.775	0.753	0.764	2.09%
NER Spanish	CW	0.738	0.720	0.729	4.09%
	CRF	0.751	0.730	0.740	2.05%

Table 3: Precision, Recall, F-measure and percentage of mislabeled words results of CW vs. CRF

setting. Three options are possible: compute a full Σ and then take its diagonal elements; compute a full inverse Σ, take its diagonal elements and then compute its inverse; assume that Σ is diagonal and compute the optimal update for this choice. We found the first method to work best, and thus employ it from now on.

The hyper parameters (r for AROW, ϕ for CW, C for PA) were tuned for each task by a single run over a random split of the data into a three-fourths training set and a one-fourth test set. We used parameter averaging with all methods.

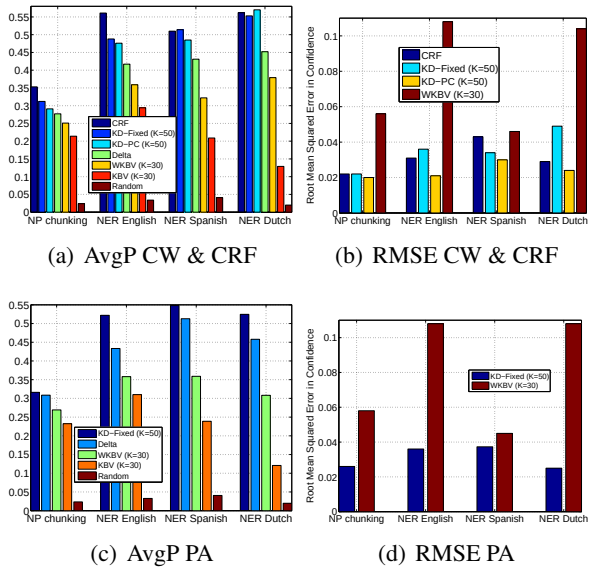
For each of the four datasets we used 10-fold cross validation. All algorithms (Perceptron, PA, CW and AROW) are online, and as mentioned above work in rounds. For each of the ten folds, each of the four algorithm performed ten (10) iterations over the training set and the performance (Recall, Precision and F-measure) was evaluated on the test set after each iteration.

The F-measure of the four algorithms after 10 iterations over the four datasets is summarized in Table 2. The general trend is that AROW slightly outperforms CW, which is better than PA that is better than the Perceptron. The difference between AROW and the Perceptron is significant, and between AROW and PA is significant in two datasets. The difference between AROW and CW is not significant although it is consistent.

We further investigate the convergence properties of the algorithms in Fig. 1. The figure shows the recall and precision results after each training round averaged across the 10 folds. Each panel summarizes the results on a single dataset, and in each panel a single set of connected points corresponds to one algorithm. Points in the left-bottom of the plot correspond to early iterations and points in the right-top correspond to later iterations. Long segments indicate a big improvement in performance between two consecutive iterations.

Few points are in order. First, high (in the y-axis) values indicate better precision and right (in the x-axis) values indicate better recall. Second, the performance of all algorithms is converging in about 10 iterations as indicated by the fact the points in the top-right of the plot are close to each other. Third, the long segments in the bottom-left for the Perceptron algorithm indicate that this algorithm benefits more from more than one pass compared with the other. Fourth, on the three NER datasets after 10 iterations AROW gets slightly higher precision values than CW, while CW gets slightly higher recall values than AROW. This is indicated by the fact that the top-right red square is left and above to the top-right blue circle. Finally, in two datasets, PA get slightly better recall than CW and AROW, but paying in terms of precision and overall F-measure performance.

In addition to online algorithms we also compared the performance of CW with the CRF algo-



(a) AvgP CW & CRF (b) RMSE CW & CRF

(c) AvgP PA (d) RMSE PA

Figure 2: Two left panels: average precision of rankings of the words of the test-set according to confidence in the prediction of seven methods (left to right bars in each group): CRF, KD-Fixed, KD-PC, Delta, WKBV, KBV and random ordering, when training with the CW algorithm (top) and the PA algorithm (bottom). Two right panels: The root-mean-squared-error of four methods that output absolute valued confidence: CRF, KD-Fixed, KD-PC and WKBV.

rithm which is a batch algorithm. We used Mallet toolkit (McCallum, 2002) for CRF implementation. For feature generation we used a combination of standard methods provided with Mallet toolkit (called pipes). We chose a combination yielding a feature set that is close as possible to the feature set we used in our system but it was not a perfect match, CRF generated about 20% fewer features in all datasets. Nevertheless, any other combination of pipes we tried only hurt CRF performance. The precision, recall, F-measure and percentage of mislabeled words of CW algorithm compared with CRF measured over a single split of the data into a three-fourths training set and a one-fourth test set is summarized in Table 3. We see that in three of the four datasets CW outperforms CRF and in one dataset CRF performs better. Some of the performance differences may be due to the differences in features.

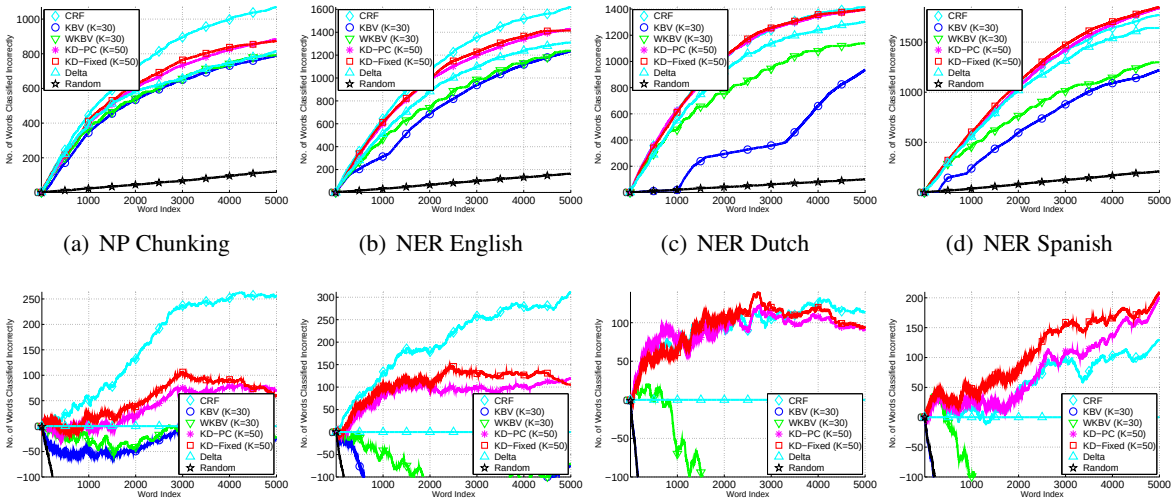
5 Confidence in the Prediction

Most large-margin-based training algorithms output models that their prediction is a single labeling of the input, with no additional confidence information about the correctness of that prediction. This situation is acceptable when the output of the system is used anyway, irrespectively of its quality. This situation is not acceptable when the output of the system is used as an input of another system that is sensitive to correctness of the specific prediction or that integrates various input sources. In such cases, additional confidence information about the correctness of these feeds for specific input can be used to improve the total output quality. Another case where such information is useful, is when there is additional agent that is validating the output of the system. The confidence information can be used to direct the check into small number of suspected predictions as opposed to random check, which may miss errors if their rate is small.

Some methods only provide *relative* confidence information. This information can be used to *rank* all predictions according to their confidence score, which can be used to direct a quality control component to detect errors in the prediction. Note, the confidence score is meaningless by itself and in fact, any monotonic transformation of the confidence scores yield equivalent confidence information. Other methods are providing confidence in the predicted output as an *absolute* information, that is, the probability of a prediction to be correct. We refer to these probabilistic outputs in a frequentists approach. When taking a large set of events (predictions) with similar probability confidence value ν of being correct, we expect that about ν fraction of the predictions in the group will be correct.

Algorithms: All of our methods to evaluate confidence, except two (Delta and CRF below), share the same conceptual approach and work in two stages. First, a method generates a set of K possible labelings for the input sentence (instead of a single prediction). Then, the confidence in a predicted labeling for a specific word is defined to be the proportion of labelings which are consistent with the predicted label. Formally, let $\boldsymbol{z}^{(i)}$ for $i = 1 \ldots K$ be the K labelings for some input $\boldsymbol{x}$, and let $\hat{\boldsymbol{y}}$ be the actual prediction for the input. (We do not assume that $\hat{\boldsymbol{y}} = \boldsymbol{z}^{(i)}$ for some i). The confidence in the label $\hat{y}_p$ of word $p = 1 \ldots |\boldsymbol{x}|$ is defined to be

$$\nu_p = |\{i \ : \ \hat{y}_p = z_p^{(i)}\}|/K \ . \qquad (5)$$



(a) NP Chunking (b) NER English (c) NER Dutch (d) NER Spanish

(e) NP Chunking (f) NER English (g) NER Dutch (h) NER Spanish

Figure 3: Total number of detected erroneous words vs. the number of ranked words (top panels), and relative to the Delta method (bottom panels). In other words, the lines in the bottom panels are the number of *additional* erroneous words detected compared to Delta method. All methods builds on the same weight-vector except CRF (see text).

We tried four approaches to generate the set of K possible labelings. The first method is valid only for methods that induce a probability distribution over predicted labels. In this case, we draw K labelings from this distribution. Specifically, we exploit the Gaussian distribution over weight vectors $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ maintained by AROW and CW, by inducing a distribution over labelings given an input. The algorithm samples K weight vectors according to this Gaussian distribution and outputs the best labeling with respect to each weight vector. Formally, we define the set $Z = \{\boldsymbol{z}^{(i)} : \boldsymbol{z}^{(i)} = \arg\max_{\boldsymbol{z}} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z}) \text{ where } \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)\}$

The predictions of algorithms that use the mean weight vector $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$ are invariant to the value of the input Σ (as noted by (Crammer et al., 2008)). However for the purpose of confidence estimation the specific value of Σ has a huge affect. Small eigenvalue of Σ yield that all the elements of Z will be the same, while large values yield random elements in the set, ignoring the input.

One possible simple option is to run the algorithm few times, with few possible initializations of Σ and choose one using the training set. However since the actual predictions of all these versions is the same (invariance to scaling, see (Crammer et al., 2008)) in practice we run the algorithm once initializing $\Sigma = I$. Then, after the training is completed, we try few scalings of the final covariance $s\Sigma$ for some positive scalar s, and choose the best value s using the training set. We refer to this method as `KD-PC` for K-Draws by Parameters Confidence.

The second method to estimate confidence follows the same conceptual steps, except that we used an isotropic covariance matrix, $\Sigma = sI$ for some positive scale information s. As before, the value of s was tuned on the training set. We denote this method `KD-Fixed` for K Draws by Fixed Standard Deviation. This method is especially appealing, since it can be used in combination with training algorithms that do *not* maintain confidence information, such as the Perceptron or PA.

Our third and fourth methods are deterministic and do not involve a stochastic process. We modified the Viterbi algorithm to output the K distinct labelings with highest score (computed using the mean weight vector in case of CW or AROW). The third method assigns uniform importance to each of the K labelings ignoring the actual score values. We call this method `KBV`, for K-best Viterbi. We thus propose the fourth method in which we define an importance weight ω_i to each labeling $\boldsymbol{z}^{(i)}$ and evaluate confidence using the weights, $\nu_p = \left(\sum_{i \text{ s.t. } \hat{y}_p = z_p^{(i)}} \omega_i\right) / \left(\sum_i \omega_i\right)$, where we set the weights to be their score value clipped at zero from below $\omega_i = \max\{0, \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z}^{(i)})\}$. (In practice,

top score was always positive.) We call this method WKBV for weighted K-best Viterbi.

In addition to these four methods we propose a fifth method that is based on the margin and does not share the same conceptual structure of the previous methods. This method provide confidence score that is only relative and not absolute, namely its output can be used to compare the confidence in two labelings, yet there is no semantics defined over the scores. Given an input sentence to be labeled $\boldsymbol{x}$ and a model we define the confidence in the prediction associated with the pth word to be the difference in the highest score and the closest score, where we set the label of that word to anything but the label with the highest score. Formally, as before we define the best labeling $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$, then the score of word p is defined to be, $\boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) - \max_{u \neq \hat{y}_p} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z}_{|z_p=u})$, where we define the labeling $\boldsymbol{z}_{|z_p=u}$ to be the labeling that agrees with $\boldsymbol{z}$ on all words, except the pth word, where we define its label to be u. We refer to this method as Delta where the confidence information is a difference, aka as delta, between two score values.

Finally, as an additional baseline, we used a sixth method based on the confidence values for single words produced by CRF model. We considered the marginal probability of the word p to be assigned the predicted label $\hat{y}_p$ to be the confide value, this probability is calculated using the forward-backwards algorithm. This method is close in spirit to the Delta method as the later can be thought of computing marginals (in score, rather than probability). It also close to the K-Draws methods, as both CRF and K-Draws induce a distribution over labels. For CRF we can compute the marginals explicitly, while for the Gaussian models generated by CW (or AROW) the marginals can not be computed expliclity, and thus a sample based estimation (K-Draws) is used.

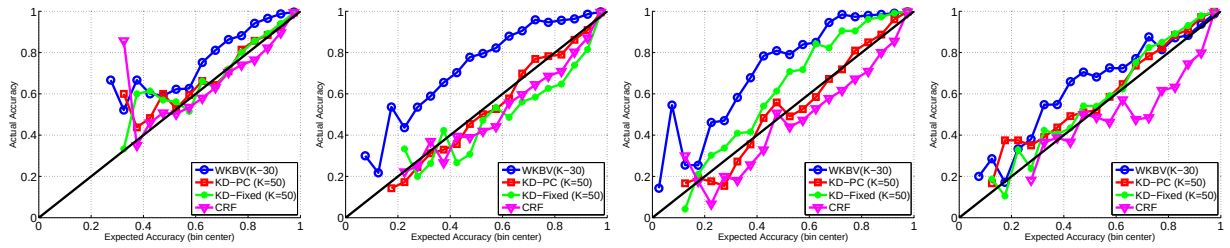
Experimental Setting: We evaluate the above methods as follows. We trained a classifier using the CW algorithm running for ten (10) iterations on three-fourth of the data and applied it to the remaining one-fourth to get a labeling of the test set. There are between $49K - 54K$ words to be labeled in all tasks, except NER Dutch where there are about $74K$ words. The fraction of words for which the trained model makes a mistake ranges between 2% (for NER Dutch) to 4.1% for NER Spanish.

We set the value of the hyper parameter ϕ to its optimal value obtained in the experiments reported in the previous section. The size of K of the number of labelings used in the four first methods (KD-PC, KD-Fixed, KBV, WKBV) and the weighting scalar s used in KD-PC and KD-Fixed were tuned for each dataset on a single evaluation on subset of the training set according to the best measured average precision. For the parameter s we tried about 20 values in the range 0.01 to 1.0, and for the number of labels K we tried the values in $10, 20 \ldots 80$. The optimal values are $K = 50$ for KD-PC and KD-Fixed, and $K = 30$ for KBV and WKBV. We noticed that KD-PC and KD-Fixed were robust to larger values of K, while the performance of KBV and WKBV was degraded significantly for large values of K.

We also trained CRF on the same training sets and applied it to label and assign confidence values to all the words in the test sets. The fraction of mislabeled words produced by the CRF model and the CW model is summarized in Table 3.

Relative Confidence: For each of the datasets, we first trained a model using the CW algorithm and applied each of the confidence methods on the output, ranking from low to high all the words of the test set according to the confidence in the prediction associated with them. Ideally, the top ranked words are the ones for which the classifier made a mistake on. This task can be thought of as a retrieval task of the erroneous words.

The average precision is the average of the precision values computed at all ranks of erroneous words. The average precision for ranking the words of the test-set according the confidence in the prediction of seven methods appears in the top-left panel of Fig. 2. (left to right bars in each group : CRF, KD-Fixed, KD-PC, Delta, WKBV, KBV and random ordering.) We see that when ordering the words randomly, the average precision is about the frequency of erroneous word, which is the lowest average precision. Next are the two methods based on the best Viterbi labelings, where the weighted approach outperforming the non-weighted version. Thus, taking the actual score value into consideration improves the ability to detect erroneous words. Next in performance is Delta, the margin-induced method. The



(a) NP Chunking (b) NER English (c) NER Dutch (d) NER Spanish

Figure 4: Predicted error in each bin vs. the actual frequency of mistakes in each bin. Best performance is obtained by methods close to the line $y = x$ (black line) for four tasks. Four methods are compared: weighted K-Viterbi (WKBV), K-draws PC (KD-PC) and K-draws fixed covariance (KD-Fixed) and CRF.

two best performing among the CW based methods are KD-Fixed and KD-PC, where the former is better in three out of four datasets. When compared to CRF we see that in two cases CRF outperforms the K-Draws based methods and in the other two cases it performs equally. We found the relative success of KD-Fixed compared to KD-PC surprising, as KD-Fixed does not take into consideration the actual uncertainty in the parameters learned by CW, and in fact replaced it with a *fixed* value across all features. Since this method does not need to assume a confidence-based learning approach we repeated the experiment, training a model with the passive-aggressive algorithm, rather than CW. All confidence estimation methods can be used except the KD-PC, which does take the confidence information into consideration. The results appear in the bottom-left panel of Fig. 2, and basically tell the same story, KD-Fixed outperform the margin based method (Delta), and the Viterbi based methods (KBV, WKBV).

To better understand the behavior of the various methods we plot the total number of detected erroneous words vs. the number of ranked words (first $5,000$ ranked words) in the top panels of Fig. 3. The bottom panels show the relative additional number of words each methods detects on top of the margin-based Delta method. Clearly, KD-Fixed and KD-PC detect erroneous words better than the other CW based methods, finding about 100 more words than Delta (when ranking $5,000$ words) which is about 8% of the total number of erroneous words.

Regarding CRF, it outperforms the K-Draws methods in NER English and NP chunking datasets, finding about 150 more words, CRF performed equally for NER Dutch, and performed worse for NER Spanish finding about 80 less words. We emphasize that all methods *except CRF* were based on the same exact weight vector, ranking the same predations, while CRF used an alternative weight vector that yields different number of erroneous words.

In details, we observe some correlation between the percentage or erroneous words in the entire set and the number of erroneous words detected among the first $5,000$ ranked words. For NP chunking and NER English datasets, CRF has more erroneous words compared to CW and it detects more erroneous words compared to K-Draws. For NER Dutch dataset CRF and CW have almost same number of erroneous words and almost same number of erroneous words detected, and finally in NER Spanish dataset CRF has fewer erroneous words and it detected less erroneous words. In other words, where there are more erroneous words to find (e.g. CRF in NP chunking), the task of ranking erroneous words is easier, and vice-versa.

We hypothesize that part of the performance differences we see between the K-Draws and CRF methods is due to the difference in the number of erroneous words in the ranked set.

This ranking view can be thought of marking suspected words to be evaluated manually by a human annotator. Although in general it may be hard for a human to annotate a single word with no need to annotate its close neighbor, this is not the case here. As the neighbor words are already labeled, and pretty reliably, as mentioned above.

Absolute Confidence: Our next goal is to evaluate how reliable are the *absolute* confidence values output by the proposed methods. As before, the confidence estimation methods (KD-PC, KD-Fixed,

KBV, WKBV and CRF) were applied on the entire set of predicted labels. (Delta method is omitted as the confidence score it produces is not in $[0, 1]$).

For each of the four datasets and the five algorithms we grouped the words according to the value of their confidence. Specifically, we used twenty (20) bins dividing uniformly the confidence range into intervals of size 0.05. For each bin, we computed the fraction of words predicted correctly from the words assigned to that bin. Ultimately, the value of the computed frequency should be about the center value of the interval of the bin. Formally, bin indexed j contains words with confidence value in the range $[(j-1)/20, j/20)$ for $j = 1 \ldots 20$. Let b_j be the center value of bin j, that is $b_j = j/20 - 1/40$. The frequency of correct words in bin j, denoted by c_j is the fraction of words with confidence $\nu \in [(j-1)/20, j/20)$ that their assigned label is correct. Ultimately, these two values should be the same, $b_j = c_j$, meaning that the confidence information is a good estimator of the frequency of correct labels. Methods for which $c_j > b_j$ are too pessimistic, predicting too high frequency of erroneous labels, while methods for which $c_j < b_j$ are too optimistic, predicting too low frequency of erroneous words.

The results are summarized in Fig 4, one panel per dataset, where we plot the value of the center-of-bin b_j vs. the frequency of correct prediction c_j, connecting the points associated with a single algorithm. Four algorithms are shown: KD-PC, KD-Fixed, WKBV and CRF. We omit the results of the KBV approach - they were substantially inferior to all other methods. Best performance is obtained when the resulting line is close to the line $y = x$.

From the plots we observe that WKBV is too pessimistic as its corresponding line (blue square) is above the line $y = x$. CRF method is too optimistic, its corresponding line is below the line $y = x$. The KD-Fixed method is too pessimistic on NER-Dutch and too optimistic on NER-English. The best method is KD-PC which, surprisingly, tracks the line $x = y$ pretty closely. We hypothesis that its superiority is because it makes use of the uncertainty information captured in the covariance matrix Σ which is part of the Gaussian distribution.

Finally, these bins plots does not reflect the fact that different bins were not populated uniformly, the bins with higher values were more heavily populated. We thus plot in the top-right of Fig. 2 the root mean-square error in predicting the bin center value given by $\sqrt{\left(\sum_j n_j (b_j - c_j)^2\right) / \left(\sum_j n_j\right)}$, where n_j is the number of words in the jth bin. We observed a similar trend to the one appeared in the previous figure. WKBV is the least-performing method, then KD-Fixed and CRF, and then KD-PC which achieved lowest RMSE in all four datasets. Similar plot but when using PA for training appear in the bottom-right panel of Fig. 2. In this case we also see that KD-Fixed is better than WKBV, even though both methods were not trained with an algorithm that takes uncertainty information into consideration, like CW.

The success of KD-PC and KD-Fixed in evaluating confidence led us to experiment with using similar techniques for inference. Given an input sentence, the inference algorithm samples K times from the Gaussian distribution and output the best labeling according to each sampled weight vector. Then the algorithm predicts for each word the most frequent label. We found this method inferior to inference with the mean parameters. This approach differs from the one used by (Crammer et al., 2009a), as they output the most frequent labeling in a set, while the predicted label of our algorithm may not even belong to the set of predictions.

6 Active Learning

Encouraged by the success of the KD-PC and KD-Fixed algorithms in estimating the confidence in the prediction we apply these methods to the task of active learning. In active learning, the algorithm is given a large set of unlabeled data and a small set of labeled data and works in iterations. On each iteration, the overall labeled data at this point is used to build a model, which is then used to choose new subset of examples to be annotated.

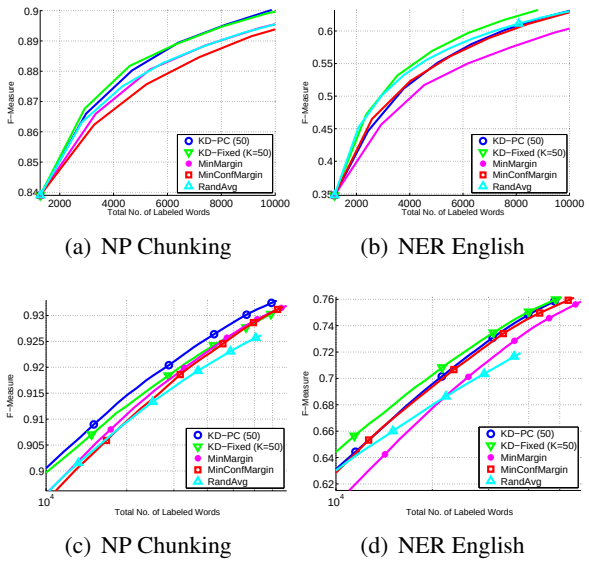
In our setting, we have a large set of unlabeled sentences and start with a small set of 50 annotated sentences. The active learning algorithm is then using the CW algorithm to build a model, which in turn is used to rank sentences. The new data items are then annotated and accumulated to the set of labeled data points, ready for the next round. Many active learning algorithms are first computing a prediction for each of the unlabeled-data examples, which is

then used to choose new examples to be labeled. In our case the goal is to label sentences, which are expensive to label. We thus applied the following setting. First, we chose a subset of 9K sentences as unlabeled training set, and another subset of size 3K for evaluation. After obtaining a model, the algorithm labels random $1,000$ sentences and chose a subset of 10 sentences using the active learning rule, which we will define shortly. After repeating this process 10 times we then evaluate the current model using the test data and proceed to choose new unlabeled examples to be labeled. Each method was applied to pick $5,000$ sentences to be labeled.

In the previous section, we used the confidence estimation algorithms to choose individual *words* to be annotated by a human. This setting is realistic since most words in each sentence were already classified (correctly). However, when moving to active learning, the situation changes. Now, all the words in a sentence are not labeled, thus a human may need to label additional words than the one in target, in order to label the target word. We thus experimented with the following protocol. On each iteration, the algorithm defines the score of an entire sentence to be the score of the least confident word in the sentence. Then the algorithm chooses the least confident sentence, breaking ties by favoring shorter sentences (assuming they contain relatively more informative words to be labeled than long sentences).

We evaluated five methods, KD-PC and KD-Fixed mentioned above. The method that ranks a sentence by the difference in score between the top- and second-best labeling, averaged over the length of sentence, denoted by MinMargin (Tong and Koller, 2001). A similar approach, motivated by (Dredze and Crammer, 2008), normalizes MinMargin score using the confidence information extracted from the Gaussian covariance matrix, we call this method MinConfMargin. Finally, We also evaluated an approach that picks random sentences to be labeled, denoted by RandAvg (averaged 5 times).

The averaged cumulative F-measure vs. number of words labeled is presented in Figs. 5,6. We can see that for short horizon (small number of sentences) the MinMargin is worse (in three out of four data sets), while MinConfMargin is worse in NP Chunking. Then there is no clear winner, but the KD-Fixed seems to be the best most of the time. The



(a) NP Chunking (b) NER English

(c) NP Chunking (d) NER English

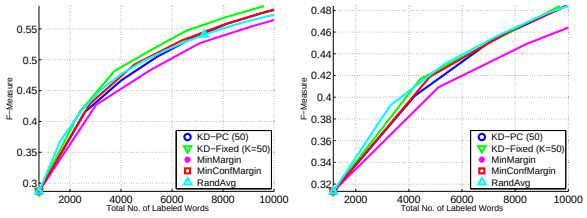
Figure 5: Averaged cumulative F-score vs. total number of words labeled. The top panels show the results for up to $10,000$ labeled words, while the bottom panels show the results for more than $10k$ labeled words.

bottom panels show the results for more than $10k$ training words. Here, the random method performing the worst, while KD-PC and KD-Fixed are the best, and as shown in (Dredze and Crammer, 2008), MinConfMargin outperforming MinMargin.

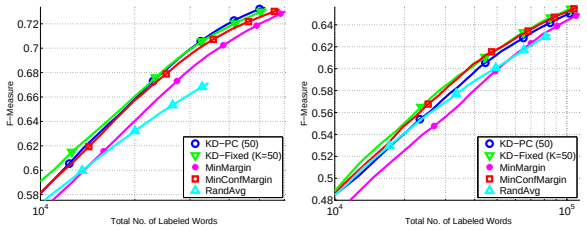
Related Work: Most previous work has focused on confidence estimation for an entire example or some fields of an entry (Culotta and McCallum, 2004) using CRFs. (Kristjansson et al., 2004) show the utility of confidence estimation is extracted fields of an interactive information extraction system by high-lighting low confidence fields for the user. (Scheffer et al., 2001) estimate confidence of single token label in HMM based information extraction system by a method similar to the Delta method we used. (Ueffing and Ney, 2007) propose several methods for word level confidence estimation for the task of machine translation. One of the methods they use is very similar to the weighted and non-weighted K-best Viterbi methods we used with the proper adjustments to the machine translation task.

Acknowledgments

The resrach is supported in part by German-Israeli Foundation grant GIF-2209-1912. KC is a Horev Fellow, supported by the Taub Foundations. The reviewers thanked for their constructive comments.



(a) NER Dutch (b) NER Spanish



(c) NER Dutch (d) NER Spanish

Figure 6: See Fig. 5

References

[Cesa-Bianchi and Lugosi2006] N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

[Collins2002] M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

[Crammer et al.2005] K. Crammer, R. Mcdonald, and F. Pereira. 2005. Scalable large-margin online learning for structured classification. Tech. report, Dept. of CIS, U. of Penn.

[Crammer et al.2008] K. Crammer, M. Dredze, and F. Pereira. 2008. Exact confidence-weighted learning. In *NIPS 22*.

[Crammer et al.2009a] K. Crammer, M. Dredze, and A. Kulesza. 2009a. Multi-class confidence weighted algorithms. In *EMNLP*.

[Crammer et al.2009b] K. Crammer, A. Kulesza, and M. Dredze. 2009b. Adaptive regularization of weighted vectors. In *NIPS 23*.

[Culotta and McCallum2004] A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. In *HLT-NAACL*, pages 109–112.

[Dredze and Crammer2008] M. Dredze and K. Crammer. 2008. Active learning with confidence. In *ACL*.

[Dredze et al.2008] M. Dredze, K. Crammer, and F. Pereira. 2008. Confidence-weighted linear classification. In *ICML*.

[Kim et al.2000] E.F. Tjong Kim, S. Buchholz, and K. Sang. 2000. Introduction to the conll-2000 shared task: Chunking.

[Kristjansson et al.2004] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *AAAI*, pages 412–418.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[McCallum2002] Andrew McCallum. 2002. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

[McDonald et al.2005a] R.T. McDonald, K. Crammer, and F. Pereira. 2005a. Flexible text segmentation with structured multilabel classification. In *HLT/EMNLP*.

[McDonald et al.2005b] Ryan T. McDonald, Koby Crammer, and Fernando C. N. Pereira. 2005b. Online large-margin training of dependency parsers. In *ACL*.

[Scheffer et al.2001] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *IDA*, pages 309–318, London, UK. Springer-Verlag.

[Sha and Pereira2003] Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, pages 213–220.

[Shimizu and Haas2006] N. Shimizu and A. Haas. 2006. Exact decoding for jointly labeling and chunking sequences. In *COLING/ACL*, pages 763–770.

[Taskar et al.2003] B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *nips*.

[Tjong and Sang2002] Erik F. Tjong and K. Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *CoNLL*.

[Tjong et al.2003] E.F. Tjong, K. Sang, and F. De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*, pages 142–147.

[Tong and Koller2001] S. Tong and D. Koller. 2001. Support vector machine active learning with applications to text classification. In *JMLR*, pages 999–1006.

[Ueffing and Ney2007] Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Comput. Linguist.*, 33(1):9–40.

[Wick et al.2009] M. Wick, K. Rohanimanesh, A. Culotta, and A. McCallum. 2009. Samplerank: Learning preferences from atomic gradients. In *NIPS Workshop on Advances in Ranking*.