

Learning Greedy Policies for the Easy-First Framework

Abstract

Easy-first, a search-based structured prediction approach, has been applied to many NLP tasks including dependency parsing and coreference resolution. This approach employs a learned greedy policy (action scoring function) to make easy decisions first, which constrains the remaining decisions and makes them easier. We formulate greedy policy learning in the Easy-first approach as a novel non-convex optimization problem and solve it via an efficient Majorization Minimization (MM) algorithm. Results on within-document coreference and cross-document joint entity and event coreference tasks demonstrate that the proposed approach achieves statistically significant performance improvement over existing training regimes for Easy-first and is less susceptible to overfitting.

Introduction

Easy-first is a general search-based structured prediction framework that has been successfully applied to a variety of natural language processing tasks including POS tagging (Shen, Satta, and Joshi 2007), dependency parsing (Goldberg and Elhadad 2010), and coreference resolution (Stoyanov and Eisner 2012). In this framework, the output is constructed incrementally by making the easiest (most confident) decision at each decision step to gather more evidence for making hard decisions later. Consider the following example from the EECB corpus for the problem of joint entity and event coreference resolution across documents (Lee et al. 2012).

(a) **Hugh Jackman** plays a furry comic-book hero.

(b) **The Australian actor** is playing a super-hero.

The coreference resolution decision for the two verb mentions “plays” and “is playing” is easy, because they share the same lemma. In contrast, the coreference decision for the two noun mentions “Hugh Jackman” and “The Australian actor” is hard based on the lexical, syntactic, semantic and discourse constraints or features (Haghighi and Klein 2010; Ng 2010). Once we establish the fact that the two verbs are coreferent, we have stronger evidence suggesting that the two noun mentions are coreferent because they serve the same semantic role to the same verb cluster.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The Easy-first approach performs greedy search according to a learned policy (scoring function), which plays a critical role in the effectiveness of easy-first. The focus of this paper is to study principled ways of learning policies to ensure the success of the Easy-first framework. In particular, we propose a novel online learning algorithm that learns a linear policy for the Easy-first approach. The contributions of this work are as follows:

- We formulate greedy policy learning as optimizing a non-convex objective consisting of two parts. The first part employs hinge loss to ensure that the learned policy ranks at least one good action higher than all the bad actions. The second part regularizes the weight vector to avoid overly-aggressive updates and over-fitting.
- We develop an efficient majorization-minimization algorithm to optimize the proposed non-convex objective, which iteratively minimizes a convex upper-bound of the objective.
- We evaluate our approach in two NLP domains: within-document entity coreference resolution and cross-document joint entity and event coreference resolution. Our results demonstrate that the proposed approach achieves statistically significant performance improvement over the baseline training approaches for the Easy-first framework and is less prone to overfitting.

Easy-first Framework and Baseline

This section first formally introduces the Easy-first framework and presents a generic online training procedure. We then describe a popular online learning algorithm for Easy-first, which serves as our baseline.

Easy-first: inference and training

Given structured inputs $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$, we assume a task-specific non-negative loss function L . The loss function $L(x, y', y) : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \mapsto \mathcal{R}^+$ associates a loss with labeling an input x with y' when its true output is y . There are two key elements in the Easy-first framework: 1) the *search space* \mathcal{S}_p , whose states correspond to partial structured outputs, and 2) an *action scoring function* f , which is used to construct the complete structured output.

The search space, \mathcal{S}_p is a 2-tuple $\langle I, A \rangle$, where I is the initial state function, and $A(s)$ is a function that gives the

Algorithm 1 Easy-first inference algorithm with learning option. When the flag *learn* is set, the algorithm performs one online training iteration on the given training example.

```

1: input : Structured input  $x$ , parameter-vector  $w$ , learning option
   learn with ground truth output  $y$ 
2:  $s \leftarrow I(x)$ 
3: TERMINATE  $\leftarrow False$ 
4: while not TERMINATE do
5:    $a_p \leftarrow \arg \max_{a \in A(s)} w \cdot \phi(a)$ 
6:   if learn is TRUE then
7:      $G(s) \leftarrow o(s)$ 
8:      $B(s) \leftarrow A(s) - G(s)$ 
9:     if  $a_p \in B(s)$  then
10:      UPDATE( $w, G(s), B(s)$ )
11:     end if
12:      $a_p \leftarrow \text{ChooseAction}(A(s))$ 
13:   end if
14:    $s \leftarrow \text{Apply } a_p \text{ on } s$ 
15:   if Terminal( $s$ ) or  $a_p = HALT$  then
16:     TERMINATE = True
17:   end if
18: end while
19: output: If learn is TRUE return  $w$ ; else return  $s$ .

```

set of allowed actions from a given state. Given the actions $A(s)$ from a specific state s , we consider any action $a \in A(s)$ that results in a state s' with $L(s') < L(s)$ as a *good action*; otherwise, it is a *bad action*. Within the Easy-first framework, it is typical to encounter states that have more than one good action. We denote the set of all good actions in state s as $G(s)$ and the set of all bad actions as $B(s)$ ($G(s) \cup B(s) = A(s)$).

The second element, the action scoring function f , evaluates all actions in $A(s)$ and guides the search to incrementally produce the structured output. In this work, we consider linear action-scoring functions $f(a) = w \cdot \phi(a)$, where w is the weight vector and $\phi(\cdot)$ is a predefined feature function. Given a structured input $x \in \mathcal{X}$ and weight vector w , the Easy-first inference procedure is illustrated in Algorithm 1 (*learn* = FALSE). The search greedily traverses the search space \mathcal{S}_p . In any state s , the scoring function f is applied to evaluate the quality of each action $a \in A(s)$. The action with the highest score is executed. This process is repeated until a terminal state is reached (for problems with a natural notion of terminal states, e.g., dependency parsing) or a HALT action is chosen (for problems like coreference resolution where we need to learn when to stop) and the predicted output is returned.

The success of the Easy-first framework hinges upon the ability to choose a good action in each decision step. Hence, the *learning goal* within the Easy-first framework is to learn a weight vector w such that the highest scoring action in each step is a *good action*. Toward this goal, a general online training procedure is described in Algorithm 1 (*learn* = TRUE). In any given state s , we assume that there exists an oracle o that can identify $G(s)$ (line 7), the set of all *good actions* given the current state and the ground truth output. If the current highest scoring action a_p is a good action, there is no need to update weights. Otherwise, the weights are up-

dated (lines 9-12). A ChooseAction procedure is then called to select the next action, and we transit to the next search state. This is repeated until the termination condition is met (line 15). Algorithm 1 (*learn* = TRUE) presents the procedure for one training iteration on a single training example (x, y) . This is typically repeated for every training example for multiple iterations, and the updated weights are collected along the way and averaged at the end (which reduces overfitting and improves performance).

There are two elements that need to be specified in this basic training procedure. First, how to perform the update (line 10), which is the main focus of this paper. Second, how to choose the next action, which determines the training trajectories (line 12). Two types of approaches have been pursued in the literature for this purpose: on-trajectory training, which always chooses an action in $G(s)$ (e.g., the highest-scoring action in $G(s)$), and off-trajectory training, which always chooses the highest-scoring action based on the current scoring function even when it is a bad action. In this work, we consider both types of training trajectories.

A baseline update strategy

Now we introduce a popular update strategy that has been widely employed in prior Easy-first work (Goldberg and Elhadad 2010; Stoyanov and Eisner 2012). This strategy aims to update the weights so that one of the good actions will score the highest. To achieve this, it uses a simple heuristic by focusing on the highest scoring good action, referred to as g^* , and the highest scoring bad action, referred to as b^* , and adjusting the weights to increase $f(g^*)$ and decrease $f(b^*)$. For this reason, we refer to this method as Best Good Best Bad (BGBB). The specific update rule for BGBB is given by Equation 1.

$$w \leftarrow w - \eta \cdot (\phi(b^*) - \phi(g^*)), \quad (1)$$

where η is the learning rate. This update is typically done repeatedly for a fixed number of iterations or until a good action is scored the highest. In each iteration, g^* and b^* are re-evaluated. While this heuristic update has been widely applied (Goldberg and Elhadad 2010; Stoyanov and Eisner 2012), it has a number of issues. Although it updates the weights in a direction that promotes a good action and demotes a bad action, there is no guarantee that the final goal (ranking a good action above all bad actions) can actually be achieved. Very frequently, even after a large number of iterations, the updated weights can still choose a bad action. In some cases, there may not exist a weight vector that ranks a good action higher than all bad actions. In such cases, the effect of the heuristic update rule is unclear, because it lacks an explicit optimization objective.

Note that easy-first can be viewed as a greedy instantiation of the *LaSO* framework (Daume III and Marcu 2005; Xu, Fern, and Yoon 2009), a search-based structured prediction framework that is applicable to both greedy and non-greedy (beam search) search procedures. Most existing work on application of *LaSO* considers an ordered (non-redundant) search space, but it is equally applicable to an unordered (redundant) search space, which is used by Easy-

first. *LaSO* weight updates are commonly done by promoting an average good action against an average bad action, which we empirically have observed to be inferior to BGBB, thus is not considered in this work.

Proposed Method

In this section, we formulate the learning problem within the Easy-first approach as an optimization objective and introduce a Majorization Minimization algorithm to optimize the proposed objective. Our goal is to *learn a linear scoring function f such that in any given state s a good action is scored higher than any bad action*. This goal can be captured by the following set of constraints:

$$\max_{a \in G(s)} f(a) > f(b) \quad \forall b \in B(s) \quad (2)$$

That is, the score of the highest scoring good action needs to exceed the score of any bad action. If we identify a weight vector w that enables f to satisfy these constraints for a given s , then Easy-first would choose a correct action in state s . Because it is not always possible to find a w that satisfies all the constraints, we introduce the following average hinge loss function to capture them as soft constraints.

$$L_h(w) = \frac{1}{|B(s)|} \sum_{b \in B(s)} [1 - \max_{a \in G(s)} w \cdot \phi(a) + w \cdot \phi(b)]_+ \quad (3)$$

where $[\cdot]_+ = \max(0, \cdot)$, $B(s)$ and $G(s)$ denote the set of bad and good actions in state s , and $\phi(\cdot)$ returns the feature vector representing the input action.

Additionally, the weights should be only updated as much as necessary to satisfy these constraints. This is inspired by passive-aggressive Perceptron training (Crammer et al. 2006) and helps avoid overly aggressive updates that can lead to over-fitting. Combining the two parts, our objective can be described as follows (w_0 represents the current weight vector prior to update):

$$\arg \min_w \lambda \|w - w_0\|^2 + \frac{1}{|B(s)|} \sum_{b \in B(s)} [1 - \max_{a \in G(s)} w \cdot \phi(a) + w \cdot \phi(b)]_+ \quad (4)$$

where λ trades-off the two aspects of the objective.

While the hinge loss is a convex function, the negative max inside the hinge loss makes the objective non-convex. To optimize this objective, we devise an efficient Majorization Minimization (MM) algorithm (Hunter and Lange 2004) to find a local optimal solution. We describe our MM algorithm in Algorithm 2.

Our optimization algorithm works in iterations. In iteration i , we create a convex surrogate objective by replacing $\max_{a \in G(s)} w \cdot \phi(a)$ with $w \cdot \phi(g^*)$, where g^* is the best good action based on the current weights w_i :

$$\arg \min_w \lambda \|w - w_0\|^2 + \frac{1}{|B(s)|} \sum_{b \in B(s)} [1 - w \cdot \phi(g^*) + w \cdot \phi(b)]_+ \quad (5)$$

Algorithm 2 The MM algorithm to solve Equation 4

```

1: input : current parameter-vector  $w_0$ , convergence threshold  $\epsilon$ ,
    $\lambda$ , good actions  $G$ , bad actions  $B$ , max number of iterations  $T$ 
2: output :  $w$ 
3:  $w \leftarrow w_0$ ,  $i \leftarrow 0$ , convergence  $\leftarrow$  false
4:  $pObj \leftarrow -\infty$ 
5: while  $i \leq T$  and !convergence do
6:    $i++$ 
7:    $g^* \leftarrow \arg \max_{a \in G} w \cdot \phi(a)$ 
8:    $w \leftarrow$  solve Equation 5 via gradient descent
9:    $cObj \leftarrow$  evaluate the objective in Equation 4
10:  if  $cObj - pObj \leq \epsilon$  then
11:    convergence  $\leftarrow$  true
12:  end if
13:   $pObj \leftarrow cObj$ 
14: end while

```

This convex objective is then optimized via gradient descent (line 8) to obtain a new weight vector w_{i+1} , which is then used to identify the g^* for the convex surrogate in the next iteration. This repeats until convergence (lines 10-12) or for a fixed number of iterations (T).

It is easy to verify that equation 5 is an upper bound to the original objective and is tight at the current weights w_i , which guarantees that our algorithm will monotonically decrease the objective until converging to a local minimum.

We omit the details of the gradient descent subroutine for solving Equation 5. However, it is worth noting that each step of the gradient descent procedure corresponds nicely to a single update step of the Best Good Best Bad approach (Equation 1). In particular, each gradient descent step performs the following:

$$w \leftarrow w - \eta \left[\lambda(w - w_0) + \frac{1}{|B(s)|} \sum_{a \in V} \phi(a) - \phi(g^*) \right], \quad (6)$$

where η is the learning rate and V is a subset of $B(s)$ that contains all the bad actions that scored higher than g^* .

Comparing to Equation 1, we note two key differences. First, our update rule does not solely focus on the best bad action. Instead, it tries to suppress all the bad actions that incur constraint violations. We argue that by considering all violated bad actions at once, we avoid the jumpy behavior of BGBB from one iteration to the next and increase learning stability. The second key difference is that our update rule has the added flexibility for explicit control of aggressiveness and overfitting in updates. By tuning parameter λ , we can achieve a trade-off between aggressively satisfying the given constraints and conservatively staying close to the current weight. Hence, we refer to our update rule as Regularized Best Good Violated Bad (RBGVVB).

Note that we can capture the easy-first learning goal with the following constraint as an alternative to Equation 2.

$$\max_{a \in G(s)} f(a) > \max_{b \in B(s)} f(b)$$

This naturally leads to the following alternative objective:

$$\arg \min_w \lambda \|w - w_0\|^2 + [1 - \max_{a \in G(s)} w \cdot \phi(a) + \max_{b \in B(s)} w \cdot \phi(b)]_+ \quad (7)$$

A similar MM procedure can be derived to solve this objective by introducing the following convex objective (fixing $g^* = \arg \max_{a \in G(s)} w_0 \cdot \phi(a)$) in each MM iteration:

$$\arg \min_w \lambda \|w - w_0\|^2 + [1 - w \cdot \phi(g^*) + \max_{b \in B(s)} w \cdot \phi(b)]_+ \quad (8)$$

And the gradient descent update rule for this objective is:

$$w \leftarrow w - \eta \left[\lambda(w - w_0) + \max_{b \in B(s)} \phi(b) - \phi(g^*) \right] \quad (9)$$

In other words, in each MM iteration, we will fix the best good action g^* , and update it against the best bad action in each gradient descent iteration. This update rule bears a strong similarity with BGG as we focus on only the best bad in each update, but with passive-aggressive regularization included. We refer to this variant as RBGG and consider it as an additional baseline in our experiments.

Experimental Evaluation

We conduct our evaluation on two NLP problem domains: within-document entity coreference resolution and joint entity and event coreference resolution across documents.

Baseline Methods

We compare our proposed Regularized Best Good Violated Bad (**RBGV**) approach with the commonly used Best Good Best Bad (**BGG**) update rule (Stoyanov and Eisner 2012). As discussed previously, RBGV differs from BGG in two key ways: first, each update of RBGV considers all bad actions that incur constraint violations; second, it incorporates passive-aggressive regularization. To understand the impact of these two factors, we also include in our comparison two additional methods: Regularized BGG (RBGG) as described at the end of the previous section, and RBGV with no regularization (BGV). Below we separately present our experimental results for each problem domain.

Entity coreference resolution within documents

We first consider the problem of entity coreference resolution within documents, which groups noun phrase mentions into clusters corresponding to entities. Within-document entity coreference resolution has been widely studied, and there exist many successful systems, including the Easy-first system (Stoyanov and Eisner 2012).

Data. For this problem, we conduct experiments on two entity coreference resolution corpora.

- **ACE04:** (ACE 2004) We employ the same training/testing partition as ACE2004-CULOTTA-TEST (Culotta et al. 2007; Bengtson and Roth 2008). There are 443 documents in total, among which 268 documents are used for training, 68 documents for validating, and 107 documents for testing.

- **OntoNotes-5.0:** (Pradhan et al. 2012) We employ the official split for training, validation, and testing. There are 2802 documents for training; 343 documents for validation; and 345 documents for testing.

We perform coreference resolution on predicted mentions extracted by the UIUC mention detector (Chang et al. 2012). The evaluation of the coreference results is carried out using the updated version¹ (7.0) of the coreference scorer. Experiments on gold mentions show similar results and are not included in the paper.

Experimental setup. We implemented our learning algorithm based upon the Easy-first coreference system (Stoyanov and Eisner 2012). We employ the same set of features, which includes cluster features capturing cluster-level global agreement and mention-pair features capturing local configurations signifying coreferences. We also follow the same protocol for handling the “HALT” action (which serves as the terminal state when selected) as (Stoyanov and Eisner 2012). In particular, we represent the HALT action by a feature vector of all zeros except for a halt feature that is set to 1. For all other actions, we set the halt feature to zero. Note that the learned weight of this halt feature operates as a threshold on action scores. In the inference stage, if no merge action scores higher than this threshold, the search procedure terminates.

We followed the experimental setup of (Stoyanov and Eisner 2012) with a few small changes. To ensure fair comparison of the different update rules, we initialize all the methods with the zero weight vector. Another difference is that we employ five-fold cross-validation for parameter tuning for all the methods. For BGG, we tune the learning rate ($\eta \in \{10^{-1}, \dots, 10^{-5}\}$) and the maximum number of repeated perceptron updates ($k \in \{1, 5, 10, 20, 50\}$) for each mistake step. For RBGV and RBGG, we tune the regularization parameter ($\lambda \in \{10^{-4}, 10^{-3}, \dots, 10^3\}$). For MM-based method including BGV, RBGV, RBGG, we tune the maximum number of MM iterations ($T \in \{1, 5, 10, 20, 50\}$) and the maximum number of gradient descent steps ($t \in \{1, 5, 10, 20, 50\}$). Note that for gradient descent, our method sets the learning rate to be one over the number of iterations. Finally, (Stoyanov and Eisner 2012) uses off-trajectory training. To remove the impact of the training trajectories, our experiments consider both off-trajectory and on-trajectory training.

Results. Table 1 show the results on the OntoNotes 5.0 and ACE2004 corpora with the predicted mentions. For evaluation, we compute MUC (Vilain et al. 1995), B^3 (Bagga and Baldwin 1998), CEAF (Luo 2005), and CoNLL F1 (Pradhan et al. 2011), all of which have been employed in the CoNLL Shared-task 2011. Note that CoNLL F1 is simply the average F1 values of the other three metrics.

From the results, we can see that our proposed approach (RBGV) consistently outperforms BGG for both corpora. In addition, we also observe that the unregularized version of our method (BGV) also outperforms BGG, although with a slightly smaller margin. Similarly, our proposed method RBGV also outperforms regularized BGG

¹<http://code.google.com/p/reference-coreference-scorers/>

ALGORITHM	MUC	BCUB	CEAF _e	CoNLL
Results on OntoNotes 5.0 corpus.				
On-Traj-RBGVB	67.51	52.05	52.46	57.34
On-Traj-BGVB	67.12	52.11	49.79	56.34
On-Traj-RBGBB	66.93	51.71	47.92	55.52
On-Traj-BGGB	66.72	50.33	47.32	54.79
Off-Traj-RBGVB	69.71	52.75	52.05	58.17
Off-Traj-BGVB	69.35	51.31	50.07	56.91
Off-Traj-RBGBB	67.55	50.97	48.82	55.78
Off-Traj-BGGB	66.02	50.76	48.61	55.13

Results on ACE 2004 corpus.				
On-Traj-RBGVB	51.09	69.97	48.08	56.38
On-Traj-BGVB	49.65	66.87	45.99	54.17
On-Traj-RBGBB	47.87	67.21	45.45	53.51
On-Traj-BGGB	47.71	64.67	41.55	51.31
Off-Traj-RBGVB	51.25	72.61	48.40	57.42
Off-Traj-BGVB	50.79	71.43	47.70	56.64
Off-Traj-RBGBB	49.23	70.14	46.77	55.38
Off-Traj-BGGB	48.91	68.21	46.47	54.53

Table 1: Results on within-document coreference corpora with predicted mentions. Metric values reflect version 7 of the CoNLL scorer.

ALGORITHM	MUC	BCUB	CEAF _e	CoNLL
On-Traj-RBGVB	66.91	50.85	49.76	55.84
On-Traj-BGVB	65.58	49.78	49.13	54.83
On-Traj-RBGBB	64.49	49.04	47.58	53.70
On-Traj-BGGB	62.45	45.71	44.87	51.01
Off-Traj-RBGVB	66.67	51.8	51.21	56.56
Off-Traj-BGVB	66.19	50.87	50.43	55.83
Off-Traj-RBGBB	65.97	50.73	48.92	55.20
Off-Traj-BGGB	61.9	44.55	43.04	49.83
Lee et al.	66.4	50.99	49.83	55.74

Table 2: Results for cross-document joint entity and event coreference resolution with predicted mentions.

(RBGBB). For these three comparisons, statistical significance tests using the Paired bootstrap resampling approach (KoeHN 2004) indicate that the performance differences we observe are statistically significant in all four measures.

These results collectively indicate that the performance gain of our proposed method over baseline BGGB is not dominated by a single factor of using regularization or considering all violated bad actions. Rather, both factors contribute to the performance improvement we observe. In the next section, we further explore the performance differences between RBGVB and BGGB.

Note that the best performance we achieved on the OntoNotes corpus (58.17 as measured by CoNLL F1) is approximately 3% lower than the current state-of-the-art on this corpus (Durrett and Klein 2013). We would like to emphasize that our results are not directly comparable to that of (Durrett and Klein 2013), which solves coreference resolution in a left-to-right order and uses very different feature designs. By incorporating more advanced features, we expect to see further improvement of our method.

Joint entity and event coreference across documents

Cross-document joint entity and event coreference resolution is a challenging problem that involves resolving the coreferences for entities (noun phrases) and events (verbs) across multiple documents simultaneously.

Data. We employ the benchmark EECB corpus (Lee et al. 2012) for our experiments. EECB contains 482 documents clustered into 43 topics. We use the same split for training, validation, and testing as Lee et al. (2012). Out of 43 topics, 12 topics are used for training, 3 topics for validation, and 28 topics for testing. We conduct experiments on predicted mentions. The predicted mentions are extracted using the same mention extraction system as Lee et al. (2012).

Features. We employ the same set of features as Lee et al. (2012) with two minor distinctions. First, in addition to the merge actions, we introduce the HALT action to serve the role of a terminal state for Easy-first search, following Stoyanov and Eisner (2012). Another minor distinction is that due to non-availability we employed a different semantic role labeling (SRL) system, which is trained on both NomBank and PropBank (Johansson and Nugues 2008).

Comparison to state-of-the-art. In addition to the three baselines (BGGB, RBGBB, BGVB) mentioned before, we compare our RBGVB approach with the current state-of-the-art cross-document joint entity and event coreference system by Lee et al. (2012).

Experimental setup. We build our experiments on top of the Stanford multi-pass sieve system (RaghuNathan et al. 2010). For all methods, we employ the same set of features and the same initial processing step to the noun-phrase mentions as described by Lee et al. (2012). We set up our experiments to closely resemble the experiments by Lee et al. (2012). The parameters of RBGVB, BGVB, and RBGBB (λ , MM iterations T , and gradient descent iterations t) and BGGB (learning rate η and maximum updates per iteration k) are tuned with five-fold cross-validation within the training set using the same range of values specified for the within-document coreference tasks. For all methods, we tune the halt feature using the validation set to determine the stopping condition for inference. For the method of Lee et al. (2012), we employ the implementation provided by the authors and follow the parameter setup suggested in the original paper.

Results. Our experiments consider both on-trajectory and off-trajectory training. Lee et al. (2012) performs offline training, where the training examples can be viewed as collected in an off-trajectory fashion (not restricted to taking good actions during training). Thus it is omitted from comparison for the on-trajectory setting. We measure the performances using the same set of metrics as for within-document entity coreference (MUC, B^3 , CEAF, CoNLL F1). We evaluate the results by employing the most up-to-date (Version 7) CoNLL evaluation software².

We present the results for predicted mentions in Table 2

²Lee et al. (2012) used V4 of the evaluation software. Thus our results do not exactly match their reported numbers. We also used V4 to evaluate the results from our implementation of their system and were able to closely replicate their reported results.

by evaluating both entity and event clusters jointly. First, we observe that our proposed method (RBGVB) outperforms the baseline BGBB on all measures for both on-trajectory and off-trajectory training. We also observe that RBGVB is consistently better than regularized BGBB (RBGBB). Comparing unregularized BGVB and BGBB, we again see a clear win for BGVB. These differences are consistent with what we have observed for the within-document tasks and are statistically significant according to paired bootstrap sampling test. Finally, comparing the performance of RBGVB to that of Lee et al. (2012), we see a very small improvement and statistical testing indicates that the difference is only significant for the CEF measure. This suggests that our method is comparable to the current state-of-the-art on this corpus.

Approach	Total Steps	Mistakes	Recoveries	Percentage
RBGVB	50195	16228	4255	0.262
BGBB	50195	11625	4075	0.351

Table 3: Training statistics on the ACE 2004 corpus.

Summary of results

Our results on both domains demonstrate that our proposed method RBGVB achieves statistically significant improvements over both BGBB and Regularized BGBB (RBGBB) across the board. The performance difference is more pronounced for the cross-document joint coreference problem for which our method consistently outperforms BGBB sometimes by a large margin. Additionally, removing the impact of the regularizer, we observe that BGVB also achieves statistically significant improvements over the BGBB method.

Discussion

There are two key distinctions between RBGVB and the BGBB update rule. First, BGBB considers only the best-scoring bad action in its update, whereas our method considers all bad actions that are causing constraint violations (in Equation 2) in each update. Second, our method follows a passive-aggressive strategy to discourage overly-aggressive updates. Our hypothesis is that these distinctions allow us to introduce more stability in learning and help avoid overfitting to specific bad actions encountered during training.

Approach	Mean	Variance	STDEV
RBGVB	0.87	0.0047	0.069
BGBB	0.82	0.0064	0.080

Table 4: Global performance of the learned weights on the ACE 2004 training corpus.

To shed some light on this hypothesis, we take a closer look at the effect of the updates during training. To avoid the impact of noisy mentions, we conduct this set of experiments on gold mentions and focus on within-document coreference and on-trajectory training. We collected some interesting statistics during five iterations of training for

RBGVB and BGBB on the ACE2004 corpus, which are presented in Table 3. The second column of the table records the total number of search steps, which are the same for both methods because we are using on-trajectory training. The third column shows the number of mistakes (bad actions chosen during search) encountered in the process of training, each incurring a round of updates. The next two columns show the number of times that the update is successful (“recoveries”, where the highest-scoring action is good after update) and its corresponding percentage.

We were surprised to note that our method (RBGVB) is significantly less successful at correcting mistakes (26% for BGVB vs. 35% for BGBB). How could it be less effective in correcting the mistakes but more effective overall? The explanation lies in the passive-aggressive element of our objective, which explicitly encourages small changes to the weights, sometimes at the expense of not satisfying all the constraints. This explains why our method tends to fail more at correcting the mistakes, but does not provide an answer to why its overall performance tends to be better.

To answer this question, we further examine the quality of the weights obtained by both methods in a global setting. That is, we recorded all the weights that are produced by the two learning algorithms and evaluated how well each weight can guide the greedy search on the training set. To do this, we generated some sample search trajectories on the training data, and at each search step evaluated the learned weights to choose actions. If a weight-vector chose a good action, it was considered a correct decision. For each method, we computed the percentage of correct decisions made by all of the learned weight-vectors, averaged across five randomly-generated search trajectories. The mean and variance are reported in Table 4. The results show that the weights learned by RBGVB have consistently better global performance and smaller variance. This suggests that by satisfying more local constraints, BGBB is indeed suffering from overfitting, to which our method is less prone.

Conclusions and Future Work

We proposed a novel online learning algorithm for the Easy-first framework. By formulating the learning problem as an optimization objective, we capture the essence of the learning goal for the Easy-first framework: select the best scoring action at each search state while avoiding overly-aggressive updates. Experiments on two NLP domains, within-document coreference resolution and cross-document joint entity and event coreference resolution, showed that our greedy learning method outperforms an existing Easy-first training method and is competitive with the current state-of-the-art for cross-document joint entity and event coreference resolution. Easy-first makes a series of greedy local decisions, some of which can be hard without additional context information, and errors on those decisions can propagate to downstream decisions. One solution we intend to explore is to perform a search in the Limited Discrepancy Search (LDS) space (Doppa, Fern, and Tadepalli 2014) induced by the learned greedy policy to further improve the performance.

References

2004. *The ACE Evaluation Plan*.
- Bagga, A., and Baldwin, B. 1998. Algorithms for scoring coreference chains. In *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, 563–566.
- Bengtson, E., and Roth, D. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, 294–303.
- Chang, K.-W.; Samdani, R.; Rozovskaya, A.; Sammons, M.; and Roth, D. 2012. Illinois-coref: The ui system in the conll-2012 shared task. In *CoNLL Shared Task*.
- Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.* 7:551–585.
- Culotta, A.; Wick, M.; Hall, R.; and McCallum, A. 2007. First-order probabilistic models for coreference resolution. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)*, 81–88.
- Daume III, H., and Marcu, D. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*.
- Doppa, J. R.; Fern, A.; and Tadepalli, P. 2014. Structured prediction via output space search. *Journal of Machine Learning Research (JMLR)* 15:1317–1350.
- Durrett, G., and Klein, D. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington: Association for Computational Linguistics.
- Goldberg, Y., and Elhadad, M. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, 742–750.
- Haghighi, A., and Klein, D. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, 385–393.
- Hunter, D. R., and Lange, K. 2004. A tutorial on mm algorithms. *The American Statistician* 58(1):30–37.
- Johansson, R., and Nugues, P. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, 183–187.
- Koehn, P. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, 388–395.
- Lee, H.; Recasens, M.; Chang, A.; Surdeanu, M.; and Jurafsky, D. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, 489–500.
- Luo, X. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, 25–32.
- Ng, V. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, 1396–1411.
- Pradhan, S.; Ramshaw, L.; Marcus, M.; Palmer, M.; Weischedel, R.; and Xue, N. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, 1–27.
- Pradhan, S.; Moschitti, A.; Xue, N.; Uryupina, O.; and Zhang, Y. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of the Joint Conference on EMNLP and CoNLL: Shared Task*, 1–40.
- Raghunathan, K.; Lee, H.; Rangarajan, S.; Chambers, N.; Surdeanu, M.; Jurafsky, D.; and Manning, C. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, 492–501.
- Shen, L.; Satta, G.; and Joshi, A. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 760–767.
- Stoyanov, V., and Eisner, J. 2012. Easy-first coreference resolution. In *COLING*, 2519–2534.
- Vilain, M.; Burger, J.; Aberdeen, J.; Connolly, D.; and Hirschman, L. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding*, 45–52.
- Xu, Y.; Fern, A.; and Yoon, S. W. 2009. Learning linear ranking functions for beam search with application to planning. *Journal of Machine Learning Research* 10:1571–1610.