Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Resource Constrained Structured Prediction

Tolga Bolukbasi,* Kai-Wei Chang,+ Joseph Wang,* Venkatesh Saligrama*
*Boston University, Boston, MA
+University of Virginia, Chancellorsville, VA
tolgab@bu.edu, kw@kwchang.net, joewang@bu.edu, and srv@bu.edu

Abstract

We study the problem of structured prediction under test-time budget constraints. We propose a novel approach based on selectively acquiring computationally costly features during test-time in order to reduce the computational cost of prediction with minimal performance degradation. We formulate a novel empirical risk minimization (ERM) for policy learning. We show that policy learning can be reduced to a series of structured learning problems, resulting in efficient training using existing structured learning algorithms. This framework provides theoretical justification for several existing heuristic approaches found in literature. We evaluate our proposed adaptive system on two structured prediction tasks, optical character recognition and dependency parsing and show significant reduction in the feature costs without degrading accuracy.

Introduction

Structured prediction is a powerful and flexible framework for making a joint prediction over mutually dependent output variables. It has been successfully applied to a wide range of computer vision and natural language processing tasks ranging from text classification to human detection. However, the superior performance and flexibility of structured predictors come at the cost of increased computational cost. In order to construct computationally efficient algorithms, a trade-off must be made between the expressiveness and speed of structured models.

The cost of inference in structured prediction can be broken down into three parts: acquiring the features, evaluating the part responses, and solving a combinatorial optimization problem to make a prediction based on part responses. Much of the past research has focused on efficient inference algorithms for specific structures (e.g., Viterbi and CKY parsing algorithms) and general structures (e.g., variational inference (Jordan et al. 1999)). However, these methods overlook feature acquisition and part response, which are bottlenecks when the underlying structure is relatively simple or can be efficiently solved.

Consider the dependency parsing task, where the goal is to create a directed tree that describes semantic relations between words in a sentence. The task can be formulated as a structured prediction problem, where inference concerns finding the maximum spanning trees (MSTs) in a directed graph (McDonald et al. 2005). Each node in the graph represents a word and a directed edge (x_i, x_j) represents how likely x_j is to depend on x_i. Fig. 1 shows an example of our selective feature acquisition scheme for this task. Intuitively, our goal is to learn a system that identifies the parts in each example that are incorrectly parsed using "cheap" features. Acquiring "expensive" features for these parts yields large reduction in error over the entire structure due to improved distinguishability and relationships to other parts.

We consider two forms of the budgeted structured learning problem, prediction under expected budget constraints and anytime prediction. For both cases, we consider the streaming test-time scenario where the system operates on each test example without observation or interaction of other test examples. For both settings, we propose a novel ERM formulation to learn the policy function and reduce this problem to a weighted structured learning problem.

Our proposed approach is general and can be applied to any structured prediction task. Given a structured prediction model, we demonstrate that our approach yields state-of-the-art speedup while maintaining predictive power. We summarize our contributions as follows:

- Novel ERM formulation of structured prediction under expected budget constraints and anytime prediction.
- Reduction of the ERM problem for both these settings to conventional structured prediction problems.
- State of the art speedup of existing structured prediction models on two real-world data sets.

Implementation details and proofs are at https://arxiv.org/abs/1602.08761.

Budgeted Structured Learning

In this section we formulate the problem of budgeted structured prediction for the expected and anytime budget settings. We describe each approach in more detail in the following sections.

Structured Prediction: The goal in structured prediction is to learn a function, F, that maps from an input space, $\mathcal{X}$, to a structure space, $\mathcal{Y}$. In contrast to multi-class classification, the space of outputs $\mathcal{Y}$ is not simply categorical but instead is assumed to be some exponential space of outputs (often of varying size dependent on the feature space) con-

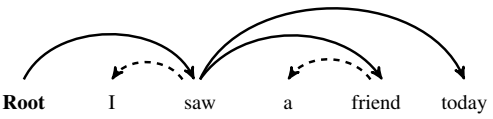

Figure 1: When predicting the dependency tree, some dependencies (e.g., the dashed edges) are easily resolved, and there is less need for expressive features in making a prediction. Our goal is to learn a system that identifies these dependencies to reduce the test-time costs.

taining some underlying structure, generally represented by multiple parts and relationships between parts. For example, in dependency parsing, $x \in \mathcal{X}$ are features representing a sentence (e.g., words, pos tags), and $y \in \mathcal{Y}$ is a parse tree.

In a structured prediction model, the mapping function F is often modeled as $F \equiv \max_{y \in Y} \Psi(x, y)$, where $\Psi : (X, Y) \rightarrow R$ is a scoring function. We assume the score can be broken up into sub-scores across components C, $\Psi(x, y) = \sum_{c \in C} \psi_c(x, y_c)$, where y_c is the output assignment associated with the component c. The number of subcomponents, $|C|$, varies across examples. For the dependency parsing example, each c is an edge in the directed graphs, and y_c is an indicator variable for whether the edge is in the parse tree. The score of a parse tree consists of the scores $\psi_c(x, y_c)$ of all its edges.

Structured Prediction Under an Expected Budget

Our goal is to reduce the cost of prediction during test-time (representing computational time, energy consumption, etc.). We consider the case where a variety of scoring functions are available to be used for each component. Additionally, associated with each scoring function is an evaluation cost (such as the time or energy consumption required to extract the features for the scoring function).

For each example, we define a state $S \in \mathcal{S}$, where the space of states is defined $\mathcal{S} = \{0, 1\}^{K \times |C|}$, representing which of the K features is used for the $|C|$ components during prediction. In a state, the element $S(k, c) = 1$ indicates that the k^{th} feature will be used during prediction for component c. For any state S, we define the evaluation cost: $c(S) = \sum_{c \in C} \sum_{k \in K} S(k, c)\delta_k$, where δ_k is the (known) cost of evaluating the k^{th} feature for a single part.

We assume that we are given a structured prediction model $F : \mathcal{X} \times \mathcal{S} \rightarrow \mathcal{Y}$ that maps from a set of features $X \in \mathcal{X}$ and a state $S \in \mathcal{S}$ to a structured label prediction $\hat{Y} \in \mathcal{Y}$. For a predicted label, we have a loss $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ that maps from a predicted and true structured label, $\hat{Y}$ and Y, respectively, to an error cost such as Hamming error, $L(\hat{Y}, Y) = \sum_{i=1}^{k} \mathbb{1}_{\hat{Y}(i) \neq Y(i)}$. For an example (X, Y) and state S. We define a policy $\pi : \mathcal{X} \rightarrow \mathcal{S}$ that maps from the feature space $\mathcal{X}$ and the initial state S_0 to a new state. For ease of reference, we refer to this policy as the feature selection policy. Our goal is to learn a policy π chosen from a family of functions Π that minimzes the expected loss subject to a budget constraint:

$$\min_{\pi \in \Pi} \mathbb{E}_{\mathcal{X}} \left[L \left(F(X, \pi(X)), Y \right) \right], \quad \text{s.t. } \mathbb{E}_{\mathcal{X}} \left[C(\pi(X)) \right] \leq B,$$

where B is a user specified test time budget. Rather an solving this constrained minimization, we define the modified loss

$$C(X, Y, S) = L \left(F(X, S), Y \right) + \lambda c(S) \tag{1}$$

that represents the error induced by predicting a label from X using the sensors in S combined with the cost of acquiring the sensors in S, where λ is a trade-off pattern adjusted according to the budget B[1]. A small value of λ encourages correct classification at the expense of feature cost, whereas a large value of λ penalizes use of costly features, enforcing a tighter budget constraint. We tune this parameter in the training for the target budget B. Our goal is to learn a policy with minimal expected modified loss, $\pi^* = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}} \left[C \left(X, Y, \pi \left(X \right) \right) \right]$. In practice, $\mathcal{D}$ denotes a set of I.I.D training examples:

$$\pi^* = \text{argmin}_{\pi \in \Pi} \sum_{i=1}^{n} C \left(X_i, Y_i, \pi \left(X_i \right) \right). \tag{2}$$

Note that the objective of the minimization can be expanded with respect to the space of states, allowing the optimization problem in (2) to be expressed $\pi^* = \text{argmin}_{\pi \in \Pi} \sum_{i=1}^{n} \sum_{S \in \mathcal{S}} C \left(X_i, Y_i, S \right) \mathbb{1}_{\pi(X_i) = S}$. From this, we can reformulate the problem of learning a policy as a structured learning problem.

Theorem 1. *The minimization in* (2) *is equivalent to the structured learning problem:*

$$\underset{\pi \in \Pi}{\text{argmin}} \sum_{i=1}^{n} \sum_{S \in \mathcal{S}} \left(\max_{\tilde{S} \in \mathcal{S}} C(X_i, Y_i, \tilde{S}) - C \left(X_i, Y_i, S \right) \right) \mathbb{1}_{\pi(X_i) \neq S}.$$

Proofs can be found in Suppl. Material. Theorem 1 maps the policy learning problem in (2) to a weighted structured learning problem. For each example X_i, an example/label pair is created for each state S with an importance weight representing the savings lost by not choosing the state S.

Unfortunately, the expansion of the cost function over the space of states introduces the summation over the combinatorial space of states. To avoid this, we instead introduce an approximation to the objective in (2). Using a single indicator function, we formulate the approximate policy

$$\hat{\pi} = \text{argmin}_{\pi \in \Pi} \sum_{i=1}^{n} \Bigg[W(X_i, Y_i) \mathbb{1}_{\pi(X_i) \neq S^*(X_i, Y_i)} + C \left(X_i, Y_i, S^*(X_i, Y_i) \right) \Bigg], \tag{3}$$

where the pseudo-label is defined:

$$S^*(X_i, Y_i) = \text{argmin}_{S \in \mathcal{S}} C \left(X_i, Y_i, S \right) \tag{4}$$

and the example weight is defined as $W(X_i, Y_i) = \max_{S' \in \mathcal{S}} C \left(X_i, Y_i, S' \right) - C \left(X_i, Y_i, S^*(X_i, Y_i) \right)$.

This formulation reduces the objective from a summation over the combinatorial set of states to a single indicator function for each example and represents an upper-bound on the original risk.

Theorem 2. *The objective in* (3) *is an upper-bound on the objective in* (2).

[1]Our framework does not restrict the type of modified loss, $C(X, Y, S)$, or the state cost, $C(S)$ and extends to general losses

Note that the second term (3) is not dependent on π. Thus, Theorem 2 leads to an efficient algorithm for learning a policy function π by solving an importance-weighted structured learning problem:

$$\hat{\pi} = \text{argmin}_{\pi \in \Pi} \sum_{i=1}^{n} W(X_i, Y_i) \mathbb{1}_{\pi(X_i) \neq S^*(X_i, Y_i)}, \quad (5)$$

where each example X_i having a pseudo-label $S^*(X_i, Y_i)$ and importance weight $W(X_i, Y_i)$.

Combinatorial Search Space: Finding the psuedo-label in Eqn. (4) involves searching over the combinatorially large space of states, $\mathcal{S}$, which is computationally intractable. Note that in limited cases, one can apply greedy approaches as in (Samdani and Roth 2012) for an exact solution. Instead, we present trajectory-based and parsimonious pseudo-labels for approximating S^*.

Trajectory Search: The trajectory-based pseudo-label is a greedy approximation to the optimization in Eqn. (4). To this end, define $\mathcal{S}_i^t$ as the 1-stage feasible transitions: $\mathcal{S}_i^t = \{S \mid d(S, \hat{S}_i^{t-1}) \leq 1, S \wedge \hat{S}_i^{t-1} = \hat{S}_i^{t-1}\}$, where d is the Hamming distance. We define a trajectory of states $\hat{S}_i^t$ where $\hat{S}_i^t = \text{argmin}_{S \in \mathcal{S}_i^t} C(X_i, Y_i, S)$. The initial state is assumed to be $\hat{S}_i^0 = 0^{K \times |C|}$ where none of the K features are evaluated for the C components. For each example i, we obtain a trajectory $\hat{S}_i^0, \hat{S}_i^1, \ldots, \hat{S}_i^T$, where the terminal state $\hat{S}_i^T$ is the all-one state. We choose the pseudo-label from the trajectory: $\hat{S}_i^* = \text{argmin}_{S \in \{\hat{S}_i^0, \ldots, \hat{S}_i^T\}} C(X_i, Y_i, S)$. Note that by restricting the search space of states differing by a single component, the approximation needs to only perform a polynomial search over states as opposed to the exhaustive combinatorial search in Eqn. (4). Observe that the modified loss is not strictly decreasing, as the cost of adding features may outweigh the reduction in loss at any time. Empirically, this approach is computationally tractable and is shown to produce strong results.

Parsimonious Search: Rather than a trajectory search, which requires an inference update as we acquire more features, we consider an alternative one stage update here. The idea is to look for 1-step transitions that can potentially improve the cost. We then simultaneously update all the features that produce improvement. This obviates the need for a trajectory search. In addition we can incorporate a guaranteed loss improvement for our parsimonious search. $\mathcal{S}_i^+ \in \text{argmin}_{S \in \mathcal{S}_i^t} \mathbf{1}_{\{C(X_i, Y_i, S_i^{t-1}) \geq C(X_i, Y_i, S) + \tau\}}$. Note that the potential candidate transitions can be non-unique and thus we generate a collection of potential state transitions, $\mathcal{S}_i^+$. To obtain the final state we take the union over these transitions, namely, $\hat{S}^* = \bigvee_{S \in \mathcal{S}_i^+} S$. Suppose we set the margin $\tau = 0$, replace the cost-function with the loss function then this optimization is relatively simple (assuming that acquiring more features does not increase the loss). This is because the new state is simply the collection of transitions where the sub-components are incorrect. Finding the parsinomious pseudo-label is computationally efficient and empirically shows similar performance to the trajectory-based pseudo-label.

Choosing the pseudo-label requires knowledge of the budget B to set the cost trade-off parameter λ. If the budget is unspecified or varies over time, a system capable of adapting to changing budget demands is necessary. To handle this scenario, we propose an anytime system in the next section.

Anytime Structured Prediction

In many applications, the budget constraint is unknown a priori or varies from example to example due to changing resource availability, and an expected budget system as in the previous section is not feasible. We instead consider the problem of learning an anytime system (Grubb and Bagnell 2012). In this setting, a single system is designed such that when a new example arrives during test-time, features are acquired until an arbitrary budget constraint (that may vary over different examples) is met for the particular example. Note that an anytime system is a special case of the expected budget constrained system. Instead of an expected budget, a hard per-example budget is specified in test-time. A single system is applied to all feasible budgets, as opposed to learning unique systems for each budget constraint.

We model the anytime learning problem as sequential state selection. The goal is to select a trajectory of states, starting from an initial state $S_0 = 0^{k \times |C|}$ where all components use features with negligible cost. To select this trajectory of states, we define policy functions $\pi_1, \ldots, \pi_T$, where $\pi_t : \mathcal{X} \times \mathcal{S} \to \mathcal{S}$ is a function that maps from a set of structured features X and current state S to a new state S'.

The sequential selection system is then defined by the policy functions $\pi_1, \ldots, \pi_T$. For an example X, the policy functions produce a trajectory of states $S^1(X) \ldots, S^T(X)$ defined as follows: $S^t(X) = \pi^t(X, S^{t-1}(X)), \qquad S^0(X) = S^0$.

Our goal is to learn a system with small expected loss at any time $t \in [0, T]$. Formally, we define this as the average modified loss of the system over the trajectory of states:

$$\pi_1^*, ..., \pi_T^* = \underset{\pi_1, \ldots, \pi_T \in \Pi}{\text{argmin}} \frac{1}{T} \mathbb{E}_{\mathcal{D}} \left[\sum_{t=1}^{T} C\left(X, Y, S^t(X)\right) \right] \quad (6)$$

where Π is a user-specified family of functions. The problem of learning the policy functions is highly coupled due to the dependence of the state trajectory on all policy functions. We propose a greedy approximation to the policy learning problem by sequentially learning policy functions $\pi_1, \ldots, \pi_T$ that minimize the modified loss:

$$\pi_t = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}} \left[C\left(X, Y, S^t(X)\right) \right] \quad (7)$$

for $t \in \{1, \ldots, T\}$. Note that the π_t selected in (7) does not take into account the future effect on the loss in (6). We consider π_t in (7) to be a greedy approximation as it is instead chosen to minimize the immediate loss at time t.

As in Thm. 1, the problem of learning the sequence of policy functions $\pi_1, \ldots, \pi_T$ can be viewed as a weighted structured learning problem.

Theorem 3. *The optimization problem in* (7) *is equivalent to solving an importance weighted structured learning problem using an indicator risk of the form:*

$$\underset{\pi \in \Pi}{\text{argmin}} \sum_{i=1}^{n} \sum_{s \in \mathcal{S}(S^{t-1}(X_i))} W(X_i, Y_i, s) \mathbb{1}_{\pi(X_i, S^{t-1}(X_i)) \neq s},$$

Algorithm 1 Anytime Policy Learning

input Training set, $\{X_i, Y_i\}_{i=1,\dots,n}$
set $S_i^0 = \mathbf{0}\ \forall i = 1, ..., n, t = 1$
while $t \leq T$ **do**
  Train π_t according to Thm. 3
  for $i \in [n]$ **do**
    Update states: $S_i^t = \pi_t(X_i, S_i^{t-1})$
  $t \leftarrow t + 1$
return $\pi = \{\pi_1, \dots, \pi_T\}$

Algorithm 2 Anytime Structured Prediction

input Policy: $\pi_1, ..., \pi_T$, Example: X, Budget: B
set $S^0 = \mathbf{0}, t = 0$
while $t \leq T$ and $c(S^t) < B$ **do**
  $S^{t+1} = \pi_t(X, S^t), t \leftarrow t + 1$
return $\mathbf{y} = F(X, S^t)$

where the weight is defined:

$$W(X_i, Y_i, s) = \max_{s' \in \mathcal{S}(S^{t-1}(X_i))} C(X_i, Y_i, s') - C\left(X_i, Y_i, s\right).$$

This is equivalent to an importance weighted structured learning problem, where each state s in $\mathcal{S}(S^{t-1}(X_i))$ defines a pseudo-label for the example X_i with an associated importance $\left(\max_{s' \in \mathcal{S}(S^{t-1}(X_i))} C(X_i, Y_i, s') - C\left(X_i, Y_i, s\right)\right)$.

Theorem 3 reduces the problem of learning a policy to an importance weighted structured learning problem. Replacement of the indicators with upper-bounding convex surrogate functions results in a convex minimization problem to learn the policies $\pi_1, ..., \pi_T$. In particular, use of a hinge-loss surrogate converts this problem to the commonly used structural SVM. Experimental results show significant cost savings by applying this sequential policy.

The training algorithm is presented in Algorithm 1. At time $t = 0$, the policy π_1 is trained to minimize the immediate loss. Given this policy, the states of examples at time $t = 1$ are fixed, and π_2 is trained to minimize the immediate loss given these states. The algorithm continues learning policies until every feature for every example as been acquired. During test-time, the system sequentially applies the trained policy functions until the specified budget is reaches, as shown in Algorithm 2.

Related Work

Multi-class prediction with test-time budget has received significant attention (see e.g., (Viola and Jones 2001; Chen et al. 2012; Busa-Fekete, Benbouzid, and Kégl 2012; Karayev, Fritz, and Darrell 2013; Xu et al. 2013; Trapeznikov and Saligrama 2013; Kusner et al. 2014; Wang et al. 2014; Wang, Trapeznikov, and Saligrama 2014)). Fundamentally, multi-class classification based approaches cannot be directly applied to structured settings for two reasons: (1) Structured Feature Selection Policy: Unlike multi-class prediction, in a structured setting, we have many parts with associated features and costs for each part. This often requires a coupled adaptive part by part feature selection policy applied to varying structures; (2) Structured Inference Costs: In contrast to multi-class prediction, structured prediction requires solving a constrained optimization problem in test-time, which is often computationally expensive and must be taken into account.

Strubell et al. (2015) improve the speed of a parser that operates on search-based structured prediction models, where joint prediction is decomposed to a sequence of decisions. In such a case, resource-constrained multi-class approaches can be applied, however this reduction only applies to search-based models that are fundamentally different from the graph-based models we discussed (with different types of theoretical guarantees and use cases). Applying their policy to the case of graphical models requires repeated inferences, dramatically increasing the computational cost when inference is slow.[2]

Similar observations apply to (Weiss, Sapp, and Taskar 2013; Weiss and Taskar 2013)), who present a scheme for adaptive feature selection assuming the computational costs of policy execution and inference are negligible. Their approach uses reinforcement learning, requiring inference at each step of their policy to estimate rewards. For complex tasks, repeatedly performing inference can negate any computational gains induced by adaptive feature selection (see Fig. 3 in (Weiss and Taskar 2013)).

He et al. (2013) use imitation learning to adaptively select features for dependency parsing. Their approach can be viewed as an approximation of Eqn. (5) with a parsimonious search. Although their policy avoids performing inference to estimate reward, multiple inferences are required for each instance due to the design of action space. Overhead is avoided by exploiting the specific inference structure (maximal spanning tree over fully connected graph), and it is unclear if it can be generalized.

Methods to increase the speed of inference (predicting the given part responses) have been proposed (Weiss and Taskar 2010; Shi, Steinhardt, and Liang 2015). These approaches can be incorporated into our approach to further reduce computational cost and therefore are complementary. More focused research has improved the speed of individual algorithms such as object detection using deformable parts models (Felzenszwalb et al. 2010; Zhu et al. 2014) and dependency parsing (He, Daumé III, and Eisner 2013; Strubell et al. 2015). These methods are specialized, failing to generalize to varying graph size and/or structures and relying on problem-specific and algorithm-specific properties.

Adaptive features approaches have been designed to improve accuracy, including easy-first decoding strategies (Goldberg and Elhadad 2010; Stoyanov and Eisner 2012), however these methods focus on performance as opposed to computational cost.

[2]The equivalent policy of (Strubell et al. 2015) applied to our inference algorithm is marked as the myopic policy in our experiments. Due to the high cost of repeated inference, the resulting policy is computationally intensive.

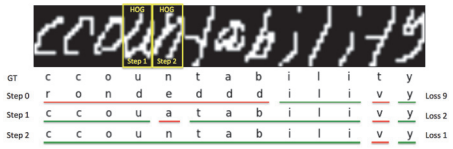

Figure 2: An example word from the OCR test dataset. Note that the word is initially incorrectly identified due to degradation in letters "u" and "n". The letter classification accuracy increases after the policy acquires the HOG features at strategic positions.

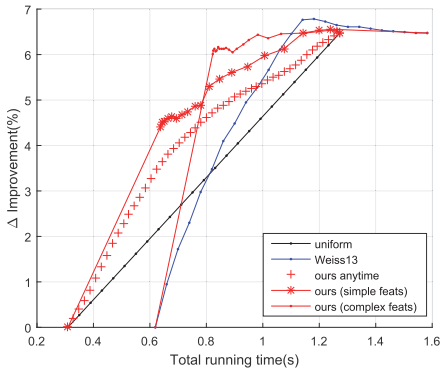

Figure 3: Comparison of our one-shot policy (in red) with uniform strategy (in black) and (Weiss and Taskar 2013)'s policy for the OCR dataset. We obtained a top accuracy of 93%, which is not shown in the plots. Although the rate of accuracy gain is large for the policy that utilizes complex features, we note that policies that utilize simple features achieve lower absolute run-time in the low budget region. This is due to the overhead arising from additional inference required for complex features.

Experiments

In this section, we demonstrate the effectiveness of the proposed algorithm on two structured prediction tasks in different domains: dependency parsing and OCR. We report the results on both anytime and expected case policies and refer to the latter one as one-shot policy.

Cost Computation: At a high-level, policies for resource constrained structured prediction must manage & trade-off benefits of three resources, namely, feature acquisition costs, intermediate inference costs, and policy overhead costs that decides between feature acquisition and inference. Some methods as described earlier account for feature costs but not inference and overhead costs. Other methods incorporate inference into their policy (meta-features) for selecting new features but do not account for the resulting policy overhead. Our approach poses policy optimization as a structured learning problem and jointly optimizes these resources.

Comparisons to Existing Methods: We compare our system to the Q-learning approach in (Weiss and Taskar 2013) and two baselines: a uniform policy and a myopic policy. The uniform policy takes random part level actions. The uniform policy will help us show that the performance of our policy does not come from removing redundant features, but clever allocation of them among samples. As a second baseline, we adapt the myopic policy used by (Trapeznikov and Saligrama 2013) to the structured prediction case. The myopic policy runs the structured predictor initially on all cheap features, then looks at the total confidence of the classifier normalized by the sample size (e.g. sentence length). If the confidence is below a threshold, it chooses to acquire expensive features for all positions. Finally, we compare against the Q-learning method proposed by (Weiss and Taskar 2013). This method requires global features for structures with varying size. From now on we will refer to features that require access to more than one part as complex features and part level features as simple features. In their case, they use confidence feedback from the structured predictor which induces additional inference overhead for the policy. In addition to this, it is not straightforward to apply this approach to do part by part feature selection on structures with varying sizes.

Policy Overhead: Recall that policy evaluation must be factored into test-time costs. Since our policy solves a structured inference problem this can be high when rich features and structure are used. Our approach allows maximum flexibility in terms of designing the best policy overhead versus performance trade-off. Experiments show that complex features indeed benefit the policy, but simple features (e.g., w/o transition features) perform better for cases where the inference time and feature costs are comparable and the additional overhead is unwanted. In our OCR and DP experiments, we use simple features and model the policy with a weighted SVM.

One shot results are obtained by sweeping λ in Equation 1. Anytime results show the average performance if all test examples terminate at particular budget. We discuss the implementation details in Suppl. Material.

Optical Character Recognition We tested our algorithm on a sequence-label problem, the OCR dataset (Taskar, Guestrin, and Koller 2003) composed of 6,877 handwritten words, where each word is represented as a sequence of 16×8 letter images. We use a linear-chain Markov model, and similar to the setup in (Weiss, Sapp, and Taskar 2013; Wang et al. 2014), use raw pixel values and HOG features with 3×3 cell size as our feature templates. We split the data such that 90% is used for training and 10% is used for test.

Fig. 3 shows the average letter accuracy vs. total running time. Note that (Weiss and Taskar 2013) can not operate on part by part level when the graph structure is varying. We see that using clever part by part selection has significant advantage over using uniform feature templates. The one-shot policy performs better than the any-time policy because it has more information (the test-time budget) during training as well as being less constrained (samples can use varying budgets compared to every sample using the same budget). Finally, Fig 2 shows the behavior of the policy on an individual example for the anytime model, significant gains in accuracy are made in first several steps by correctly identifying the noisy letters.

The significant speedup for our approach for OCR (and

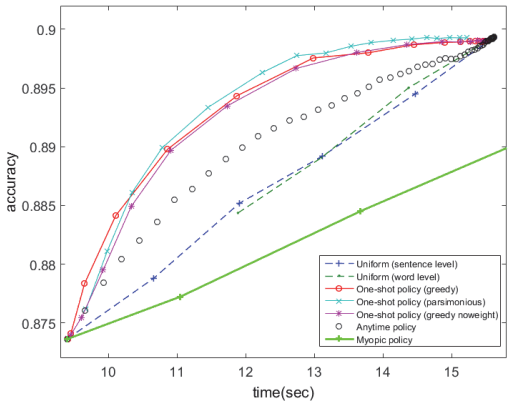

Figure 4: Performance of various adaptive policies for varying budget levels (dependency tree accuracy vs. total execution time), is compared to a uniform strategy on word and sentence level, and myopic policy for the 23 section of PTB dataset.

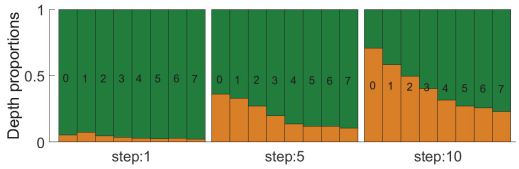

Figure 5: Distribution of parse-tree depth for words that use cheap (green) or expensive features (orange) for anytime policy. Time increases from left to right. Each group of columns show the distribution of depths from 0(root) to 7. The policy is concentrated on acquiring features for lower depth words. A sentence example also shows this effect. It is easy to identify parents of the adjectives and determiner. However, additional features(orange) are required for the root(verb), subject and object.

dependency parsing) can be partially attributed to the fact that feature acquisition cost is relatively close to inference costs. Therefore, methods such as (Weiss and Taskar 2013) that require inference feedback on acquired features can perform poorly.

Dependency Parsing We follow the setting in (He, Daumé III, and Eisner 2013) and conduct experiments on English Penn Treebank (PTB) corpus (Marcus, Marcinkiewicz, and Santorini 1993). All algorithms are implemented based on the graph-based dependency parser (McDonald et al. 2005) in Illinois-SL library (Chang et al. 2015), where the code is optimized for speed. Two sets of feature templates are considered for the parser. [3] The first (ψ^{Full}) considers the part-of-speech (POS) tags and lexicons of x_i, x_j, and their surrounding words (see (McDonald et al. 2005)). The other (ψ^{POS}) only considers the POS features. The policy assigns one of these two feature templates to each word in the sentence, such that all the directed edges (x_i, x_j) corresponding to the word x_i share the same feature templates. The first feature template, ψ^{POS}, takes 165 μs per word and the second feature template, ψ^{Full}, takes 275 μs per word to extract the features and compute edge scores. The decoding by ChuLiu-Edmonds algorithm is 75 μs per word, supporting our hypothesis that feature extraction makes a significant portion of the total running time yet the inference time is not negligible. Due to the space limit, we present further details of the experiment setting in the appendix.

Fig. 4 shows the test performance (unlabeled attachment accuracy) along with inference time. We see that all one-shot policies perform similarly, and the anytime policy is below one-shot policy for all budget levels. As discussed in Anytime Structured Prediction, the anytime policy is more constrained in that it has to achieve a fixed budget for all examples. The naive myopic policy performs worse than uniform since it has to run inference on samples with low confidence two times, adding approximately 4.5 seconds of extra time for the full test dataset. The effect of importance weights for the greedy policy seems small. We hypothesize that this is due to the policy functional complexity being the limiting factor.

When we apply the length dictionary filtering heuristic in (He, Daumé III, and Eisner 2013; Rush and Petrov 2012), our parser achieves 89.7% UAS on PTB section 23 with overall running time merely 7.5 seconds (I/O excluded, 10s with I/O) and obtains 2.9X total speed-up while losing only 1% UAS compared to the baseline. [4] This significant speed-up over an efficient implementation is remarkable.

Fig. 4 shows the distribution of depth for the words that use expensive and cheap features in the ground truth dependency tree. This result is particularly informative, because any uniform feature selection policy would uniformly distribute the features among all bins. Hence, the result clearly shows the advantage of using a part-by-part selection policy.

Conclusion

We presented a novel method for reducing feature acquisition cost in structured learning while maintaining prediction performance. Our method jointly incorporates feature costs with prediction error in its objective. We propose two new formulations for two test-time settings (one-shot and

[3] Complex features often contribute to small performance improvement. Adding redundant features can easily yield arbitrarily large speedups, and comparing speedups of different systems with different accuracy levels is not meaningful (see Fig. 3 in (He, Daumé III, and Eisner 2013)). In addition, greedy-style parser such as (Strubell et al. 2015) might be faster by nature. On the other hand, several recently developed neural dependency parsers can achieve higher accuracy, but are slower (Andor et al. 2016). Discussing different architectures is outside the scope of this paper.

[4] This heuristic only works for parsing. Therefore, we exclude it when presenting Figure 4 as it does not reflect the performance of policies in general.In contrast, the baseline system in (He, Daumé III, and Eisner 2013) is slower than us by about three times when operating at 90% accuracy, Figure 3 in (He, Daumé III, and Eisner 2013) shows that their final system takes about 20s. We acknowledge that (He, Daumé III, and Eisner 2013) use different features, policy settings, and hardware from ours; therefore these numbers might not be comparable.

anytime) and show that both can be reduced to instances of structured learning problems. Our policy leads to understanding the utility of simple and complex features in different regimes (low-budget vs. high-budget) providing the user with the flexibility to seek the suitable tradeoffs. Our method can utilize any structured inference algorithm and available feature choices. Experimentally, we showed significant speedup for problems that require structured inference, where both feature acquisition cost and structured inference cost must both be taken into account.

Acknowledgment. This material is based upon work supported in part by NSF Grants CCF: 1320566, CNS: 1330008, CCF: 1527618, DHS 2013-ST-061-ED0001, ONR Grant 50202168 and US AF contract FA8650-14-C-1728.

References

Andor, D.; Alberti, C.; Weiss, D.; Severyn, A.; Presta, A.; Ganchev, K.; Petrov, S.; and Collins, M. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Busa-Fekete, R.; Benbouzid, D.; and Kégl, B. 2012. Fast classification using sparse decision DAGs. In *29th International Conference on Machine Learning (ICML 2012)*, 951–958. Omnipress.

Chang, K.-W.; Upadhyay, S.; Chang, M.-W.; Srikumar, V.; and Roth, D. 2015. IllinoisSL: A JAVA Library for Structured Prediction. *arXiv preprint arXiv:1509.07179*.

Chen, M.; Weinberger, K. Q.; Chapelle, O.; Kedem, D.; and Xu, Z. 2012. Classifier cascade for minimizing feature evaluation cost. In *International Conference on Artificial Intelligence and Statistics*, 218–226.

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; and Ramanan, D. 2010. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32(9):1627–1645.

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*, 742–750. Association for Computational Linguistics.

Grubb, A., and Bagnell, D. 2012. Speedboost: Anytime prediction with uniform near-optimality. In *International Conference on Artificial Intelligence and Statistics*, 458–466.

He, H.; Daumé III, H.; and Eisner, J. 2013. Dynamic Feature Selection for Dependency Parsing. In *EMNLP*, 1455–1464.

Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S.; and Saul, L. K. 1999. An introduction to variational methods for graphical models. *Machine learning* 37(2):183–233.

Karayev, S.; Fritz, M.; and Darrell, T. 2013. Dynamic feature selection for classification on a budget. In *International Conference on Machine Learning (ICML): Workshop on Prediction with Sequential Models*.

Kusner, M. J.; Chen, W.; Zhou, Q.; Xu, Z. E.; Weinberger, K. Q.; and Chen, Y. 2014. Feature-Cost Sensitive Learning with Submodular Trees of Classifiers. In *AAAI*, 1939–1945.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics* 19(2):313–330.

McDonald, R.; Pereira, F.; Ribarov, K.; and Hajič, J. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 523–530. Association for Computational Linguistics.

Rush, A., and Petrov, S. 2012. Vine pruning for efficient multi-pass dependency parsing learned prioritization for trading off accuracy and speed. In *NAACL*.

Samdani, R., and Roth, D. 2012. Efficient decomposed learning for structured prediction. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 217–224.

Shi, T.; Steinhardt, J.; and Liang, P. 2015. Learning Where to Sample in Structured Prediction. In *AISTATS*.

Stoyanov, V., and Eisner, J. 2012. Easy-first Coreference Resolution. In *COLING*, 2519–2534. Citeseer.

Strubell, E.; Vilnis, L.; Silverstein, K.; and McCallum, A. 2015. Learning Dynamic Feature Selection for Fast Sequential Prediction. *arXiv preprint arXiv:1505.06169*.

Taskar, B.; Guestrin, C.; and Koller, D. 2003. Max-Margin Markov Networks. In *Advances in Neural Information Processing Systems*, None.

Trapeznikov, K., and Saligrama, V. 2013. Supervised sequential classification under budget constraints. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 581–589.

Viola, P., and Jones, M. 2001. Robust real-time object detection. *International Journal of Computer Vision* 4.

Wang, J.; Bolukbasi, T.; Trapeznikov, K.; and Saligrama, V. 2014. Model selection by linear programming. In *Computer Vision–ECCV 2014*. Springer. 647–662.

Wang, J.; Trapeznikov, K.; and Saligrama, V. 2014. An LP for Sequential Learning Under Budgets. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, 987–995.

Weiss, D., and Taskar, B. 2010. Structured Prediction Cascades. In *International Conference on Artificial Intelligence and Statistics*, 916–923.

Weiss, D. J., and Taskar, B. 2013. Learning adaptive value of information for structured prediction. In *Advances in Neural Information Processing Systems*, 953–961.

Weiss, D.; Sapp, B.; and Taskar, B. 2013. Dynamic structured model selection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2656–2663.

Xu, Z.; Kusner, M.; Chen, M.; and Weinberger, K. Q. 2013. Cost-Sensitive Tree of Classifiers. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 133–141.

Zhu, M.; Atanasov, N.; Pappas, G. J.; and Daniilidis, K. 2014. Active deformable part models inference. In *Computer Vision–ECCV 2014*. Springer. 281–296.