A Study of Imitation Learning Methods for Semantic Role Labeling

Travis Wolfe **Mark Dredze** **Benjamin Van Durme**
Human Language Technology Center of Excellence
Johns Hopkins University

Abstract

Global features have proven effective in a wide range of structured prediction problems but come with high inference costs. Imitation learning is a common method for training models when exact inference isn't feasible. We study imitation learning for Semantic Role Labeling (SRL) and analyze the effectiveness of the Violation Fixing Perceptron (VFP) (Huang et al., 2012) and Locally Optimal Learning to Search (LOLS) (Chang et al., 2015) frameworks with respect to SRL global features. We describe problems in applying each framework to SRL and evaluate the effectiveness of some solutions. We also show that action ordering, including easy first inference, has a large impact on the quality of greedy global models.

1 Introduction

In structured prediction problems, global features express dependencies between related pieces of a label and make inference non-trivial. In Semantic Role Labeling (SRL) (Gildea and Jurafsky, 2002), global features and constraints have been studied extensively (Punyakanok et al., 2004; Toutanova et al., 2008; Täckström et al., 2015) inter alia. SRL has many phenomenon that relate labels such as syntactic control, role mutual exclusion, and structural constraints like span overlap.

Previous work on inference for models with global features has studied a variety of method including dynamic programming, reranking, and ILP solvers. Greedy search and beam search are relatively understudied areas due to the difficulty in training models which perform well with the weak guarantees provided by greedy search. The Violation Fixing Perceptron (VFP) framework (Huang et al., 2012) is a notable exception which has been used to great effect in a range of structured problems. Learning to Search (L2S) (Daumé III and Marcu, 2005; Chang et al., 2015) is another line of work for training greedy models with no assumptions about features. These training methods are appealing because they decouple the definition of (global) features from the (exact) inference and training procedures. This allows easier specification of models (features not algorithms) and the ability to use inference methods which scale with the difficulty of the problem rather than the type of features used.

In this work, we study VFP and L2S methods for training greedy global SRL models. We find that both methods are far from ideal. VFP is inconsistent and often doesn't perform better than unstructured perceptron training. L2S leads to models which under-predict arguments and do not perform as well as pipeline training. We describe the causes of these problems and offer some solutions.

Finally, we study the effect of the transition system on the usefulness of global features. We find that the order that actions are performed in can be as important as the training method, leading to better models with the same features and computational complexity.

2 Problem Formulation

Semantic role labeling (Gildea and Jurafsky, 2002) (SRL) is the task of locating and labeling (with roles) the semantic arguments to predicates. Adding a step, frame semantic parsing (Das et al., 2010) (FSP), seeks to first disambiguate predicates by labeling them with a frame before performing SRL. Semantic roles abstract over grammatical function and provide information about particular arguments relation to an event, state, or fact. SRL has been shown to be helpful in a variety of NLP tasks including information extraction, question answering, and coreference resolution.

Let x refer to a sentence and its POS tags and dependency parse. For this work, we are given x and a vector of predicate locations $t = [t_1, t_2, ...t_n]$, where each t_i is a span, most often representing a single verb like "love" in the sentence "John loves Mary".

SRL and FSP are defined with respect to a schema which provides a set of frames and roles which will serve as labels for predicates and arguments. We consider two schemas, Propbank (Kingsbury and Palmer, 2002) and FrameNet (Fillmore, 1982; Baker et al., 1998). Propbank frames concern different senses of a lexical unit (a lemma and POS tag), so the correct frame for "love" in the case above is the frame `love-v-1`, as opposed to `love-v-2`, which is only used in modal cases like "I would love to go on vacation". In the FrameNet schema, frames are coarser grain situations which may have many lexical units which map to them. In this case frame would be `Experiencer_focus` which could also be evoked by the *adore.v* or *despise.v* lexical units. These frames will constitute another vector $f = [f_1, f_2, ...f_n]$ of frames for each predicate in t.

Once t and f are known, the schema defines a function mapping a frame to a set of roles $K(f_i)$ which each frame must have filled explicitly (by some mention span in the sentence) or implicitly (by some other discourse entity not directly mentioned in the sentence). For the latter case we say that an unfilled role is filled by a special dummy span called $\emptyset$. For the former case, we could in principle predict any span within the sentence, but to make systems faster and more accurate, a pruning step is often used which picks out only the spans which are plausible arguments to a particular predicate conditioned on a syntactic parse (Xue and Palmer, 2004). We call this set $S(t_i)$[1] and it always includes $\emptyset$. SRL is the task of predicting a matrix $k = \{k_{ij} : i \in [1..n], j \in K(f_i), k_{ij} \in S(t_i)\}$ where k_{ij} is the location of the j^{th} role for frame f_i evoked by the predicate at t_i. For the rest of this paper, we will concern ourselves with the FSP task of predicting both f and k.

Transition System A transition system provides a way to break down an assignment to (f, k) into a sequence of actions. The transition systems we use in this work all use the trivial definition that an action is a variable index and value (i.e. an assignment). A state is comprised of a sequence of actions and constitutes a partial assignment. A state is written $s_t = [a_0, a_1, ...a_{t-1}]$.

Transition systems in this work vary by their ordering over variable indices to fill in.

Other orders will be discussed further in §7, but for now our transition system will predict frames first, in left-to-right sentence order, followed by roles for each frame (in the same order). The roles for a given frame are ordered by how many times they were instantiated in the training and dev data.

3 Global Features

Global features are important for a couple reasons. First, a variety of insights and statistical regularities from previous work (Punyakanok et al., 2004; Toutanova et al., 2008; Täckström et al., 2015) can be described using global features on states and actions. Our definitions will not be fully equivalent to the formulation in previous work, but will draw on the same set of information. Second, global features are by their nature very expressive, and using approximate inference, they will serve as a stress test for various imitation learning methods. In this section we will list our global features and their motivations.

`numArgs` is a global feature template which counts how many arguments a given predicate has realized in a sentence. This is perhaps the simplest type of information which is expressable in a global model but not a local one. This is useful because it serves as a dynamic or contingent intercept. Normally an argument is predicted if its score exceeds 0 (or the score of the action corresponding $\emptyset$), but

[1]Extensions like the one described in Täckström et al. (2015) consider the role during the pruning step, but we gloss over this detail in our notation for simplicity.

with this global feature that threshold also depends on how many arguments have already been labeled.

The remaining global features are pairwise features, meaning they can be expressed as templates of the form $h(a_i, a_t)$ where a_i is any action in the history s_t and a_t is the current action to be scored.

`roleCooc` is a feature template which expresses which roles co-occur with each other in a predicate argument structure. There are some hard role co-occurrence constraints in the annotation guidelines for both Propbank and FrameNet which this feature aims to learn. For Propbank, continuation and reference roles may not appear without their base counterpart. FrameNet does not have this distinction between base, continuation, and reference roles, but instead has some mutual exclusion relationships between frame elements (roles) such as the `Entities`, `Entity_1`, and `Entity_2` roles for the `Similarity` frame. `Entity_1` and `Entity_2` require each other's realization and both are mutually exclusive with the `Entities` role. These roles exist so that there is a sensible way to annotate sentences like "[The two painters]$_{\texttt{Entities}}$ were [alike]$_{\texttt{Similarity}}$" as well as "[Our economy]$_{\texttt{Entity\_1}}$ is [like]$_{\texttt{Similarity}}$ [a healthy plant]$_{\texttt{Entity\_2}}$"

If $R(a_t)$ is a function which returns the role of an action a_t (assuming a_t assigns a value in k), then the pairwise definition of this feature is $h(a_i, a_t) = (R(a_i), R(a_t))$.

`argLoc` is a feature template which describes the linear relationship between argument spans. This relationship $pos(s_1, s_2)$ is the all-pairs relationship between the starts and end indices of the two spans, where two indices are said to be either "left", "left and bordering", "equal", "right and bordering", or "right". If $E(a_t)$ is a function which returns the span of an action a_t (assuming a_t assigns a value in k), then $h(a_i, a_t) = pos(E(a_i), E(a_t))$ This can encode overlap, nesting, or boundary relationships between argument spans.

`roleCoocArgLoc` is the pointwise product of `roleCooc` and `argLoc`. This feature can capture regularities like "a continuation role is to the non-bordering left of the base role" which depend on information from both `argLoc` and `roleCooc`.

Finally `full` refers to all templates together.

Refinements We designed the features in a way as to be overly general. For example, consider `numArgs` and its effects for various frames. A value like 4 may be very unlikely for a frame like `see-v-3` which was instantiated with exactly one argument in each of the 24 times it appeared in Propbank. But, a value of 4 is below average for a frame like `afford-v-1` which was observed 43 times with an average of 4.2 realized arguments.

While `numArgs` seems like it should depend on the frame, there are other cases like the FrameNet role exclusion and requires relationships which should hold regardless of frame. For example, the frames `Amalgamation`, `Becoming_separated`, `Cause_to_amalgamate`, and `Separating` all have the same pattern concerning the `Parts`, `Part_1`, and `Part_2` roles. These frames were seen only 2, 2, 9, and 12 times in training data respectively, so generalizing this rule by pooling training data is crucial.

To choose the right granularity for the global feature templates, we consider multiple refinements. A refinement of a template is the result of taking the pointwise product of the template with one or two label features templates. The label feature templates we consider are constant (a backoff feature), frame, role, and frame-role. For each global feature template, we try each refinement and use the one with the best dev set F-measure when trained with LOLS.

4 Experimental Design

We measure performance on two data sets, the Propbank annotations (Kingsbury and Palmer, 2002) available in the Ontonotes 5.0 corpus (Pradhan et al., 2012) and FrameNet 1.5 (Baker et al., 1998).

For all learning methods we average the weights across all iterations of training (Freund and Schapire, 1999). This is explicitly called for as a part of LOLS and is also a standard trick used with the structured perceptron.

We use the local features described in Hermann et al. (2014) for argument and frame identification, but we did not use their feature embedding method since it performed about as well as the sparse feature method and was slower. We use the best refinements using the process described in §3.

We are studying the fully greedy case of inference in this work (i.e. a beam size of 1). As far as we know, efficient greedy and easy first inference are mutually exclusive goals, and we focus on the latter. Our implementation uses a heap to store actions in a manner similar to Goldberg and Elhadad (2010). This way actions can be generated once, instead of once per transition, and global features perform sparse updates to the actions on the heap. For beam search, states cannot share a heap (since their histories, and thus global features, would be different), so actions generation, global features, and action sorting would have to occur at every transition.

All performance values shown here are measured for the task of frame semantic parsing (FSP), meaning that we measure precision, recall, and F-measure where every index in f and k are considered predictions. Predictions in k are not correct unless the frame that they correspond to are also correct. We show two scenarios: gold f refers to the case where the frame labels are given and auto f refers to when they are predicted by the model. All figures and plots are on dev set performance.

Unless specified, we set the loop order over roles by how frequently they occur in the dev data, which will be described as **freq** in §7.

5 Violation Fixing Perceptron

Violation Fixing Perceptron (VFP) (Huang et al., 2012) is a family of perceptron updates which are intended to train machines which operate using beam search. The beam holds states, and at every step an action is appended to each state to reach a successor state which is put on the next beam.

In VFP, the core concept is a violation. A tuple (x, y, z), where x is a sentence as defined earlier and y is a string of correct actions (having zero cost/loss), and z is a string of predicted actions, is a *violation* if $\theta \cdot f(x, z) > \theta \cdot f(x, y)$ and z is "incorrect". There are multiple ways of defining incorrect which yield different algorithms in the VFP family. In all variants y and z must be the same length and if there is more than one incorrect (x, y, z), the one with the largest difference in score is chosen. In the early update variant, first described by Collins and Roark (2004), z is incorrect if it differs from y *only* in the *last* position. In max violation z is incorrect if it differs *any* position. In latest update z is incorrect if it differs in the *last* position (but can include other differences, unlike early update).

	Gold f		Auto f	
Global Feature	PB $\Delta\ell$	FN $\Delta\ell$	PB $\Delta\ell$	FN $\Delta\ell$
`numArgs`	-0.4	-0.1	-1.3	+0.3
`roleCooc`	-0.4	-0.3	-0.1	+0.6
`argLoc`	-1.2	-0.4	-1.9	+0.2
`roleCoocArgLoc`	-2.0	-0.2	0.0	+0.2
`full`	-1.5	-0.7	-2.0	+0.2

Figure 1: Global model advantage using max violation VFP and **freq**.

Results In figure 1 we plot the difference in performance between a model which includes a particular global feature type and the baseline model which only uses local features. Almost across the board the values are negative, indicating that the global model performs worse, even though the local model is nested within the global model (i.e. there exists a parameter setting in the global model such that it is equivalent to the local model). This result is at odds with previous results which have successfully used max violation perceptron to train models with non-local features. We hypothesize that the reason performance goes down is due to the expressivity of our global features and the inconsistency problem described in Chang et al. (2015).

Briefly, the inconsistency comes from the fact that the weights derived from VFP training simultaneously, and ambiguously, reflect what to do conditioned on being in a state arrived at by the *oracle* or the *predictor*. These two distributions over states are different if the predictor cannot perfectly mimic the oracle (the beam separability assumption). At test time, all of the states will be reached from the *predictor*'s actions, so the contribution of what to do by possibly incorrectly assuming the state/history was created by the *oracle* is misleading. This can be very bad when the global features are expressive and the predictor makes a significant number of mistakes.

Inconsistency To validate that inconsistency is responsible for this poor performance, we setup another experiment where we artificially make the task easier. If the model is more accurate, then the predictor will necessarily be closer to the oracle, meaning that the inconsistency will shrink towards 0. To

	Gold f		Auto f	
Global Feature	PB $\Delta\ell$	FN $\Delta\ell$	PB $\Delta\ell$	FN $\Delta\ell$
`numArgs`	0.0	0.0	+0.2	+0.7
`roleCooc`	-0.6	-0.3	-0.1	+0.5
`argLoc`	-0.4	+0.1	-0.1	-0.4
`roleCoocArgLoc`	+0.4	+0.4	+0.1	-0.1
`full`	+0.6	+0.4	-0.1	+0.3

Figure 2: Global model advantage using LOLS and **freq**.

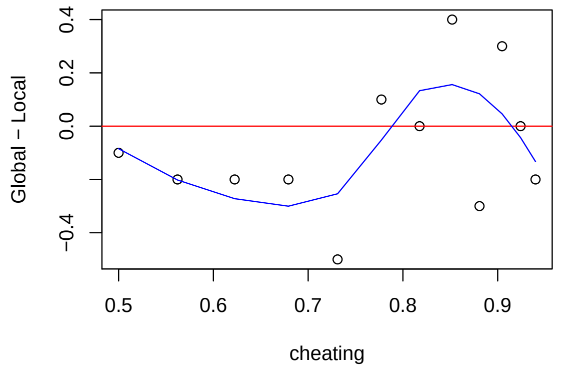

Figure 3: Benefit of `roleCooc` global features as a function of inconsistency in the model.

make the task easier, we added a binary feature to the local features which was either 1 or -1 based on whether the action has cost 0. We flip the sign of this feature with probability $1 - \alpha$. A model with $\alpha = 1$ should get perfect accuracy and $\alpha = \frac{1}{2}$ offers no extra information.

Figure 3 shows the difference between a global model using `roleCooc` and a local model (both receiving the "cheating" feature) for various values of α. This experiment used FrameNet data and max violation VFP. The local model does better than the global model (below the red line) where the inconsistency is high ($\alpha < 0.75$) and worse where it is low. Though the plot is noisy, when $\alpha = 1$ the two models have the same performance.

This result explains why max violation training has been shown to be successful in tasks like POS tagging and shift-reduce parsing, where the accuracy of the model is in the 90s. VFP with global features improves over local models on these tasks because the inconsistency is small, and the benefit from global features is great.

6 Learning to Search

Learning to search (L2S) is a family of imitation learning algorithms including early update perceptron (Collins and Roark, 2004), LaSO (Daumé III and Marcu, 2005), SEARN (Daumé III et al., 2009), DAgger (Ross et al., 2011), and LOLS (Chang et al., 2015). The unifying feature of these algorithms is that they all are a reduction of training transition based models to a cost-sensitive classification problem over (s_t, a_t) pairs.

Chang et al. (2015) showed that when the reference (oracle) policy is optimal, which we can guarantee in our case,[2] the cost estimates may be derived from reference roll-outs, which can be easily computed in constant time. Given reference cost estimates, the only way to distinguish within this family is with respect to the roll-in distribution. The LOLS algorithm prescribes using the current policy for rolling-in, which does not always work well, which we will return to in §8.2.

Results In figure 2 we plot global model advantage using the **freq** action orderings and LOLS training. There are mixed results; some global features are actually improving over the local model (something which was not achieved by VFP training). We will return to why this is in §8.2, but first we will analyze an orthogonal aspect of the model.

7 Action Ordering

So far our transition system considers actions sorted by frequency of a role, which may not be optimal. Here we measure the effect of other orderings.

Easy First The first motivation is related to easy first inference (Shen et al., 2007; Raghunathan et al., 2010) inter alia. The idea is that the "easiest" decisions should be made first because there is less risk that they are wrong and may be more safely conditioned on in making future decisions than any other action. To implement this heuristic, we define two variants of the **easyfirst** meta action ordering. **easyfirst-dynamic** chooses the variable index corresponding to the highest scoring action. **easyfirst-static** chooses variable indices sorted by

[2]Every action fills in a label and we can say whether it is right or wrong, thus the reference policy is the one which always fills in a correct label.

the dev set F-measure of the local model (most accurate visited first).[3]

Baselines The **freq** ordering sorts actions by how frequently their role appears in the training set, most frequent first. This be seen as a very naive version of **easyfirst**, but with the nice property that it is independent of the local model.

From a model (estimator) bias and variance point of view, we should expect dynamic orderings to have higher variance (whether they have lower bias is a somewhat related but empirical question). In our case, we could track this variance by proxy and look at the number of nonzero global features, as is common in the sparsity-inducing regularization literature. Consider training a model with the `roleCooc` global feature on single example, a frame with K roles. With **easyfirst-dynamic**, there are K^2 possible `roleCooc` nonzero features, whereas with **easyfirst-static** and **freq** the maximum is $\frac{K(K-1)}{2}$ since the order is fixed at training time.

To see if increased variance is responsible for potential differences in the **easyfirst** variants, we construct a parallel situation with random orderings: **rand-static** and **rand-dynamic**. The first chooses a random ordering over roles which is used throughout training and testing, and the second chooses a random ordering every time inference is run.

Results In figure 4 we have plotted models trained with each global feature type and each action ordering. The first thing to notice is the variance across different action orderings is generally larger than the variance across different global features (for the best action ordering). This indicates that action ordering is important, perhaps more so than the global features used. This is an important result considering that most previous work on transition based inference has not addressed automatic ordering.

Next, there is little consistency between Propbank and FrameNet. We believe the major reason for this is the amount of training data (Propbank has 20.7 times more instances and 1.58 more instances per type), causing overall accuracy to be higher and **easyfirst** inference to work better.

Looking at the number of non-zero global features, we see virtually no correlation between that measure of capacity and performance, on either data set. While this metric is often used in static (local) models to describe capacity, we believe this metric is less meaningful with global features.

Note that **rand-dynamic** works well on FrameNet, only losing to a non-random ordering once (**easyfirst-static** on `argLoc`). Given the overall worse performance of our model on FrameNet, and the dearth of training data, we hypothesize that **rand-dynamic** is actually providing a regularizing effect similar to dropout (Hinton et al., 2012). Since both **rand-static** and **rand-dynamic** are random, they offer no real signal they could differ on (bias is the same), and using the standard bias-variance argument we should expect **rand-static** to do no worse since **rand-dynamic** introduces additional variance into the model estimate. Our only explanation for the results is that **rand-static** is overfitting in a way which **rand-dynamic** isn't capable of.

Consistent with overfitting, we see that on both data sets **easyfirst-static** usually does as well or better than **easyfirst-dynamic**. In the opposite fashion of the random orderings, here the dynamic version is more expressive and likely to overfit.

8 Error Analysis

Neither VFP nor LOLS worked for our transition transition systems out of the box. Here we discuss problems encountered with each algorithm and offer some solutions for fixing them. We do not claim these solutions are robust, but hopefully offer insight into potential difficulties in training models like this.

8.1 Violation Fixing Perceptron

The max violation version of VFP dictates that the violation to be corrected is the solution to $\text{argmin}_{(x,y,z)\in C, z\in \bigcup_i \{\mathcal{B}_i[0]\}} w_t \cdot \Delta\Phi(x,y,z)$ Where $\mathcal{B}_i$ is the beam holding actions at step i^{th} and C is the beam confusion set as defined in (Huang et al., 2012). With only local features, Φ and $\Delta\Phi$ decompose into a sum over actions and and the argmin can be pushed inside that sum. This is equivalent to an (unstructured) perceptron update for every step in the trajectory. When global features are added, the update to the local features ceases to match the un-

[3]F-measure is computed from MAP estimates of precision and recall under a $\beta(1, \frac{5}{4})$ prior, slightly rewarding frequency.

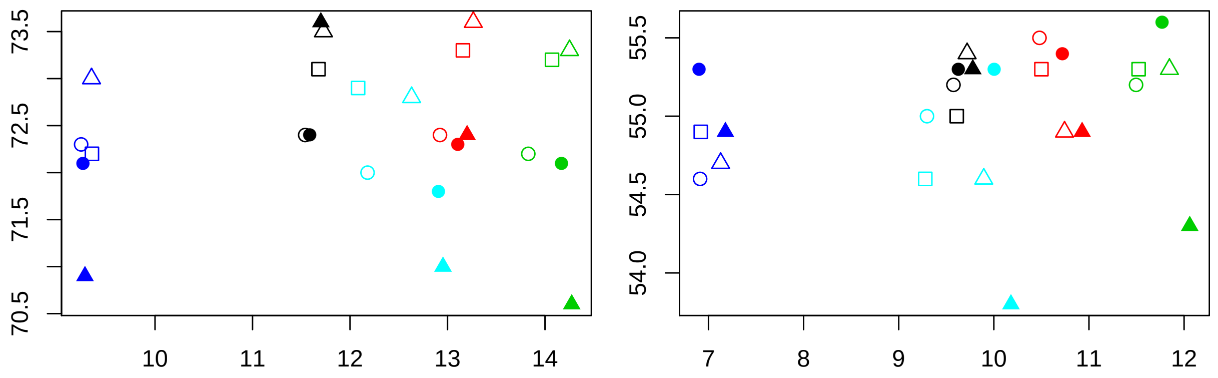

Figure 4: Model performance (y) by log number of non-zero global features (x). Propbank (left) and FrameNet (right). Global feature type by color: numArgs, roleCooc, argLoc, argLocRoleCooc, and full. **easyfirst** is triangle, **freq** is square, **rand** is circle. Filled in means dynamic, hollow is static.

structured perceptron update and both global and local features are only updated with respect to a prefix of the oracle and predicted trajectory.

This prefix update may mean that mistakes at the end of the trajectory will not be corrected until the mistakes at the beginning are fixed.[4] Skipping training data puts global models at a disadvantage over local ones, and we attribute the poor performance of the global models to this issue.

This problem was one of the motivations of max violation over the early update strategy introduced by Collins and Roark (2004). Huang et al. (2012) described an update called "latest update" which chooses the longest prefix which was still a violator, presumably to address the problem of skipping training data. While this may help, it is still possible to construct examples where a classification update would be made but a "latest update" would not.

For example, let $s(y_i) = w \cdot \phi(x, y_i)$ and $s(y_{[1:i]}) = w \cdot \phi(x, y_{[1:i]})$, such that $w \cdot \Delta\Phi(x, y_{[1:i]}, z_{[1:i]}) = s(y_{[1:i]}) - s(z_{[1:i]})$. Assume local scores $s(y_i)$ are derived from one-hot vectors indexed by (i, y_i). Assume a global model with the form: $s(y_{[1:i]}) = \sum_{k<j} w \cdot f(y_k, y_j) + \sum_j s(y_j)$ Take a sequence of binary decisions over the alphabet $\{a, b\}$ with mistakes at indices i and j such that $i < j$. Assume greedy search.

$$\begin{aligned} y_i = a, y_j = b, z_i = b, z_j = a \\ w \cdot f(b, a) = -3, w \cdot f(x, y) = 0 \; \forall (x, y) \neq (b, a) \\ s(y_i) = 0, s(y_j) = 0 \\ s(y_{[1:i]}) = 0, s(y_{[1:j]}) = 0 \\ s(z_i) = 1, s(z_j) = 1 \\ s(z_{[1:i]}) = 1, s(z_{[1:j]}) = 1 + 1 + -3 = -1 \\ s(y_{[1:j]}) > s(z_{[1:j]}) \Leftrightarrow \Delta\Phi(x, y_{[1:i]}, z_{[1:i]}) < 0 \end{aligned}$$

$(x, y_{[1:j]}, z_{[1:j]})$ is in the confusion set, but is not a violator, even though $y_j \neq z_j$, and the classification update would change $s(y_j)$ and $s(z_j)$.

Both max violation and latest update would choose to update on $(x, y_{[1:i]}, z_{[1:i]})$ in hopes of fixing it before moving on to the mistake at j. This happens consistently in our experiments (on the FrameNet data with `roleCooc`, by the end of training more than 10% of violators contain a mistake in the suffix not chosen by max violation).

Results In figure 5 we show the performance of max violation and latest update variants of VFP along with an augmentation (+CLASS) intended to fix the issue of missing suffix mistakes. Global models were trained with the `roleCooc` feature template and **easyfirst-dynamic** action ordering. +CLASS adds an unstructured perceptron update for every index in the trajectory. This modification always helps on FrameNet, leading to global models which outperform local models, but consistently hurts on Propbank. Remember that all of these deltas are measured against a local only model,

[4]This is the intended behavior under the beam separability assumption, but this may lead to very poor performance in general.

	Gold f		Auto f	
Training	PB $\Delta\ell$	FN $\Delta\ell$	PB $\Delta\ell$	FN $\Delta\ell$
max violation	-3.5	-0.9	-1.3	-0.4
latest update	-1.4	-0.7	-1.4	-0.3
max violation +CLASS	-3.0	+1.8	-2.2	+2.4
latest update +CLASS	-2.4	+1.2	-2.4	+2.4

Figure 5: Global model advantage using `roleCooc` and **easyfirst-dynamic** across VFP variations and +CLASS.

which is a pure CLASS update, so you can think of the +CLASS variants as a linear interpolation between a global and local objective.

8.2 LOLS

LOLS performs a roll-in with the current policy. This causes many updates which are derived from mistakes during frame identification. Once the wrong frame is predicted, in argument identification the model's cost incentives flip towards trying to predict $\emptyset$ for all roles so as not to incur false positives. The roles in FrameNet are defined based on the frame[5] and in Propbank they are not consistent across frames.[6] This is arguably a pathological property of a transition system: action costs strongly depend on state.

Using LOLS (model roll-in), there is a strong bias towards choosing $\emptyset$ for all roles, leading to high precision, low recall, and overall sub-optimal models. We found that when training the argument identification parameters of the model it was better to perform a hybrid model/oracle roll-in whereby the frame identification actions were chosen by the oracle. This may not be the fault of LOLS, but the of Hamming loss for action costs, which is a bad surrogate for F-measure.

Another important component of LOLs is the choice of cost in the cost-sensitive classification reduction. We found that defining costs based on the Hamming loss of an action performed very poorly. We found much better results with the multiclass hinge encoding described in Lee et al. (2004). In figure 6 we show performance with various choices of roll-in and cost definitions. The best LOLS global models consistently improve over local models.

[5]If you label a span as the Cognizer role for the frame Opinion and that span was the Cognizer role for the Judgment frame, then the label is wrong.

[6]with the exception of `ARG0` and `ARG1` which typically correspond to proto-Agent and proto-Patient roles.

		Gold f		Auto f	
Roll-in	Cost	PB $\Delta\ell$	FN $\Delta\ell$	PB $\Delta\ell$	FN $\Delta\ell$
model	Hamming	-24.5	-15.5	-10.1	-4.9
model	Hinge	-1.7	-1.1	-0.4	+0.2
hybrid	Hamming	-22.1	-12.9	-8.9	-1.0
hybrid	Hinge	+0.8	+1.0	+0.9	+1.1

Figure 6: Global model advantage using `roleCooc` and **easyfirst-dynamic** across LOLS variations: roll-in and cost function.

8.3 Absolute Performance

Throughout the paper we have listed relative performance. Our absolute performance is 73.0 for Propbank (dev) and 55.3 for FrameNet (dev). This falls significantly short of the work of Zhou and Xu (2015) at 81.1 (PB dev), FitzGerald et al. (2015) at 79.2 (PB dev), and 72.0 (FN). Those works used non-linear neural models with multi-task distributed representations, which are not comparable to our results. However, the models of Pradhan et al. (2013) at 77.5 (PB test) and Das et al. (2012) at 64.6 (FN test) are roughly comparable, and the performance gap is still significant. While our efforts do not advance the state of the art in SRL, we hope that they are enlightening with respect to the application of various imitation learning methods.

9 Related Work

Berant and Liang (2015) used imitation learning for learning a semantic parser. Choi and Palmer (2011) explored transition based SRL and proposed some global features (e.g. copy `ARG0` from controlling predicates) but did not consider action (re-)ordering or imitation learning. Wiseman and Rush (2016) derive a learning to search framework which is related to LaSO (Daumé III and Marcu, 2005). Similar to our hybrid roll-in, they "reset" the beam as soon as the oracle prefix falls off.

10 Conclusion

In this work we study the use of imitation learning for greedy global models for SRL. We analyze the Violation Fixing Perceptron (VFP) and Locally Optimal Learning to Search (LOLS) frameworks, explaining how they fall short and offer some methods for improving them. We also study the effect of inference order on learning and the utility of global features, finding that it is a very important factor of overall performance in greedy models.

Acknowledgements This work was supported in part by the Human Language Technology Center of Excellence (HLTCOE), and a Bloomberg Data Science Research Grant.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. ACL.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. 2015. Learning to search better than your teacher. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2058–2066. JMLR Workshop and Conference Proceedings.

Jinho D Choi and Martha Palmer. 2011. Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 37–45. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. 2010. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 948–956. Association for Computational Linguistics.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *SemEval*, SemEval '12. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, Bonn, Germany.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction.

Charles Fillmore. 1982. Frame semantics. *Linguistics in the morning calm*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labelling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisboa, Portugal, September. Association for Computational Linguistics.

Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Yoav Goldberg and Michael Elhadad. 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, pages 742–750, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 142–151, Stroudsburg, PA, USA. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*. Citeseer.

Yoonkyung Lee, Yi Lin, and Grace Wahba. 2004. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99:67–81.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the*

Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012), Jeju, Korea.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning. Sofia, Bulgaria: Association for Computational Linguistics*, pages 143–52.

Vasin Punyakanok, Dan Roth, Wen tau Yih, Dav Zimak, and Yuancheng Tu. 2004. Semantic role labeling via generalized inference over classifiers. In *In: Proc. of the 8th Conference on Natural Language Learning (CoNLL-2004)*, pages 130–133.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics.

Stphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 627–635. Journal of Machine Learning Research - Workshop and Conference Proceedings.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic, June. Association for Computational Linguistics.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Comput. Linguist.*, 34(2):161–191, June.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *CoRR*, abs/1606.02960.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *EMNLP*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1127–1137.