Presented in the 10th European Workshop on Reinforcement Learning 1–9

An investigation of imitation learning algorithms for structured prediction

Andreas Vlachos ANDREAS.VLACHOS@CL.CAM.AC.UK
Computer Laboratory, University of Cambridge, UK.

Abstract

In the imitation learning paradigm algorithms learn from expert demonstrations, thus relieving practicioners from having to specify a reward function. Daumé III et al. (2009) framed structured prediction in this paradigm and developed the search-based structured prediction algorithm (SEARN) which has been applied successfully to various natural language processing tasks with state-of-the-art performance. Recently, Ross et al. (2011) proposed the dataset aggregation algorithm (DAGGER) and evaluated it in sequential prediction problems. In this paper, we compare these two algorithms in the context of a complex structured prediction task, namely biomedical event extraction. We demonstrate that DAGGER has more stable performance and faster learning than SEARN, and that these advantages are more pronounced in the parameter-free versions of the algorithms.

Keywords: Real-world Applications, Imitation Learning, Natural Language Processing, Structured Prediction.

1. Introduction

Imitation learning algorithms aim at learning controllers from demonstrations by human experts (Schaal, 1999; Abbeel, 2008; Syed, 2010) without requiring the specification of a reward function by the practicioner. Intuitively, the algorithm observes a human expert perform a series of actions to accomplish the task in question and learns a policy that "imitates" the expert by inferring the rewards. These actions have dependencies between them, since earlier ones affect the input to the following ones and the algorithm needs to handle the discrepancy between the actions of the expert in the demonstration during training and the actions predicted by the learned controller during testing. Imitation learning algorithms have been applied to a variety of domains and tasks including autonomous helicopter flight (Coates et al., 2008) and statistical dialog management (Syed and Schapire, 2007).

Daumé III et al. (2009) observed that a similar discrepancy occurs in structured prediction, namely that the gold standard used in training differs from the predictions made during testing. They proposed an imitation learning algorithm, search-based structured prediction (SEARN), that reduces the problem of learning a model for structured prediction into learning a set of classifiers. This conversion enables SEARN to tackle structured prediction tasks with complex output spaces that are not amenable to sequential or grammar-based approaches. It has been applied successfully to a variety of tasks including summarization (Daumé III et al., 2009) and biomedical event extraction (Vlachos and Craven, 2011).

In this work, we investigate a novel imitation learning algorithm proposed by Ross et al. (2011), dataset aggregation (DAGGER) that also reduces the structured prediction problem

to classification. It was compared to Searn on learning video game-playing agents and handwriting recognition and was shown to be more stable and have faster learning while achieving state-of-the-art performance.

In this paper we make the following contributions. We present Searn and DAgger in a unified description, highlighting the connections between imitation learning and structured prediction. We compare them in the context of biomedical event extraction (Kim et al., 2011), a structured prediction problem more complex than handwriting recognition, and confirm the aforementioned advantages of DAgger over Searn which are more pronounced in the parameter-free versions of the algorithms. Furthermore, we explore the effect of the learning rate on the balance between precision and recall achieved by the algorithms. We believe that these contributions are relevant to applications of imitation learning algorithms to other structured prediction tasks, as well as to the development and evaluation of imitation learning algorithms.

2. Imitation learning algorithms for structured prediction

Searn and DAgger form the structured output prediction of an instance s as a sequence of T actions $\hat{y}_{1:T}$ made by a learned hypothesis H. The latter is a set of classifiers h_i that are learned jointly. Each action $\hat{y}_t$ can use features from s and all previous actions $\hat{y}_{1:t-1}$, thus exploiting possible dependencies.

Algorithm 1 presents the training procedure for Searn and DAgger. Both algorithms require a loss function ℓ that compares structured output predictions of the training data instances $\mathcal{S}$ against the gold standard. In addition, an *optimal policy* $\pi^\star$ must be specified which returns the action $\hat{y}_t$ that minimizes the loss over the instance given the previous actions $\hat{y}_{1:t-1}$ assuming that all future actions $\hat{y}_{t+1:T}$ are also optimal. $\pi^\star$ is typically derived from the gold standard and it must be able to deal with mistaken $\hat{y}_{1:t-1}$. Finally, the learning rate β and a cost sensitive classification (CSC) learner (*CSCL*) must be provided. In CSC each training instance has a vector of misclassification costs associated with it, thus rendering some mistakes on some instances to be more expensive than others (Domingos, 1999).

Each training iteration of both algorithms begins by setting the probability p (line 3) of using $\pi^\star$ in the current policy π. In the first iteration only $\pi^\star$ is used but in later iterations π becomes stochastic as for each action we use $\pi^\star$ with probability p and the learned hypothesis from the previous iteration h_{i-1} with probability $1-p$ (line 4). Then π is used to predict each instance s in the training data (line 8). For each s and each action $\hat{y}_t$, a CSC example is generated (lines 10-17). The features Φ_t are extracted from s and the previous actions $\hat{y}_{1:t-1}$ (line 10). The cost for each possible action y_t^j is estimated by predicting the remaining actions $y\prime_{t+1:T}$ in s using either π or $\pi^\star$(line 13 or 15) and calculating the loss incurred given that action w.r.t. the gold standard using ℓ (line 16). Using a CSC learning algorithm, a new classifier h_i is learned (line 18) which is combined with the previously learned ones to form the new hypothesis H_i.

The optimal policy in robotics is the expert demonstrating the task, which is equivalent to an annotator reproducing the gold standard in structured prediction. In each iteration, the algorithm predicts the instances in the training data sometimes querying the optimal policy, estimates the cost of each action and learns hypotheses that are progressively better

Algorithm 1: Imitation learning training

Input: training data $\mathcal{S}$, *optimal policy* $\pi^\star$, loss function ℓ, learning rate β, CSC learner *CSCL*
Output: hypothesis H_N

1 Examples $E = \emptyset$
2 **for** $i = 1$ **to** N **do**
3 $p = (1-\beta)^{i-1}$
4 current policy $\pi = p\pi^\star + (1-p)H_{i-1}$
5 **if** SEARN **then**
6 Examples $E = \emptyset$
7 **for** s ***in*** S **do**
8 Predict $\pi(s) = \hat{y}_{1:T}$
9 **for** $\hat{y}_t$ ***in*** $\pi(s)$ **do**
10 Extract features $\Phi_t = f(s, \hat{y}_{1:t-1})$
11 **foreach** *possible action* y_t^j **do**
12 **if** SEARN **then**
13 Predict $y\prime_{t+1:T} = \pi(s|\hat{y}_{1:t-1}, y_t^j)$
14 **else**
15 Predict $y\prime_{t+1:T} = \pi^\star(s|\hat{y}_{1:t-1}, y_t^j)$
16 Estimate $c_t^j = \ell(\hat{y}_{1:t-1}, y_t^j, y\prime_{t+1:T})$
17 Add (Φ_t, c_t) to E
18 Learn a hypothesis $h_i = CSCL(E)$
19 **if** SEARN **then**
20 $H_i = \beta \sum_{j=1}^{i} \frac{(1-\beta)^{i-j}}{1-(1-\beta)^i} h_j$
21 **else**
22 $H_i = h_i$

at imitating the expert. This procedure is commonly referred to as inverse reinforcement learning (Abbeel and Ng, 2004), since unlike standard reinforcement learning, an optimal policy is given but we try to learn the reward for each action (cost). The reason we want to learn a hypothesis even though we have an optimal policy is that the latter is only available for the training data, while the former can generalize to unseen data. By gradually decreasing the use of the optimal policy in the current policy, both algorithms adapt the learned hypothesis to its own predictions.

The main algorithmic difference between SEARN and DAGGER is in the learning of the classifiers h_i in each iteration and in combining them into a hypothesis H_i. Under SEARN, each h_i is learned using only the CSC examples generated in iteration i (line 6). It is combined with the classifiers learned in the previous iterations $h_{1:i-1}$ according to the learning rate β (line 20), which results in a weighted ensemble of hypotheses biased towards the more recently learned ones. On the other hand, DAGGER learns h_i using CSC examples from iterations $1:i$ and uses it as the learned hypothesis H_i (line 22). Thus DAGGER can

combine the training signal obtained from all iterations more flexibly, which results in faster learning and more stable performance compared to Searn.

Another difference between the two algorithms is that DAgger uses the optimal policy $\pi^\star$ to predict the remaining actions in $y\prime_{t+1:T}$.[1] This approach to costing had been proposed by Daumé III et al. (2009) in the context of Searn, referred to as optimal approximation.

The learning rate β determines how fast the current policy π moves away from $\pi^\star$. A special case is obtained when $\beta = 1$, also referred to as pure policy iteration or parameter-free. In this case, $\pi^\star$ is used only in the first iteration to reproduce the gold standard and π only uses only the learned hypothesis from the previous iteration H_{i-1}. Furthermore, the classifier combination under Searn becomes the same as the one of DAgger, i.e. only the most recently learned classifier is used. In this setting the algorithms cannot query $\pi^\star$ after the first iteration, thus $\pi^\star$ does not need to handle mistakes in previous actions since all actions are optimal in the first iteration. Furthermore, the predictions in lines 8 and 12 become deterministic. However, relying only on the learned hypotheses for action prediction beyond the first iteration (note that the cost estimation still uses the gold standard via the loss function) renders the learning harder as the algorithms are given less supervision.

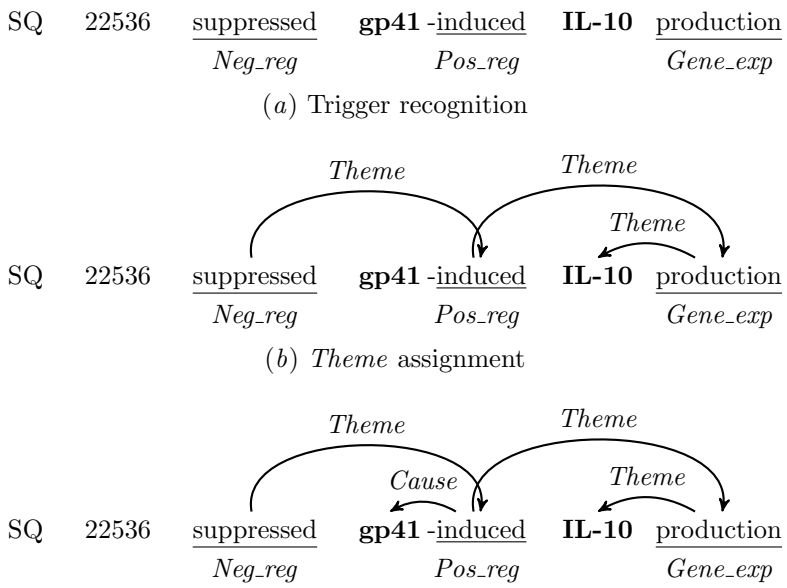
3. Tackling biomedical event extraction with imitation learning

The term biomedical event extraction is used to refer to the task of extracting descriptions of actions and relations among one or more entities from the biomedical literature. In the BioNLP 2011 shared task GENIA Task1 (BioNLP11ST-GE1) (Kim et al., 2011), each event consists of a trigger and one or more arguments, the latter being proteins or other events. Protein names are annotated in advance and any token in a sentence can be a trigger for one of the nine event types. Depending on their event types, triggers are assigned theme and cause arguments. In an example demonstrating the complexity of the task, given the passage "...SQ 22536 suppressed gp41-induced IL-10 production in monocytes", systems should extract the three appropriately nested events listed in Figure 1(*d*). The task can be viewed as a structured prediction problem in which the output for a given instance is a directed acyclic graph in which vertices correspond to triggers or proteins, and edges represent argument assignments. Thus it is not amenable to several commonly used structured prediction methods which rely on the output structure being a sequence or a tree. Performance is measured using Recall, Precision and F-score over complete events, i.e. the trigger, the event type and the arguments all must be correct in order to obtain a true positive.

Following Vlachos and Craven (2011), we treat each sentence independently and decompose event extraction in four stages: trigger recognition, theme assignment, cause assignment and event construction (Fig. 1). Apart from the last one which is rule-based, each stage has its own module to perform the classification needed. The basic features used are extracted from the lemmatization and the syntactic parse of the sentence. Furthermore, we extract structural features for each action from the previous ones.

The loss function sums the number of false positive and false negative events, following the task evaluation. The optimal policy for a sentence is derived from the events contained in the gold standard and returns the action that minimizes the loss over the sentence given

1. This fact was pointed out by a reviewer as it was not mentioned by Ross et al. (2011).



(*a*) Trigger recognition

(*b*) *Theme* assignment

(*c*) *Cause* assignment

ID	type	*Trigger*	*Theme*	*Cause*
E1	*Neg_reg*	suppressed	*E2*	
E2	*Pos_reg*	induced	*E3*	gp41
E3	*Gene_exp*	production	IL-10	

(*d*) Event construction

Figure 1: The stages of our biomedical event extraction system.

the previous actions and assuming that all future actions are optimal. In the first iteration it returns the actions required to reproduce the gold standard events for a sentence. In subsequent iterations, in order to deal with mistakes in previous actions, it avoids assigning arguments to incorrectly tagged triggers and avoids using incorrect events as arguments of other events.

The classifiers learned for trigger recognition, theme assignment and cause assignment are unlikely to be able to replicate to the optimal policy due to the difficulty of the tasks. In such cases, the optimal approximation method for costing (line 15 in Alg. 1) is unlikely to estimate the cost of each action correctly, since the actions predicted by the learned hypothesis are likely to differ from the ones returned by the optimal policy. Therefore, in our experiments we use the SEARN-style cost estimation (line 13 in Alg. 1) for both SEARN and DAGGER. In order to restrict the effects of the stochasticity of this approach, we use the *focused costing* method (Vlachos and Craven, 2011), in which the cost estimation for an action takes into account only the part of the output graph connected with that action, thus limiting the actions considered for this purpose.

The actions taken in the biomedical event extraction decomposition above are irreversible and can prohibit other actions from being taken, e.g. if a token is incorrectly predicted not to be a trigger, it is impossible to assign arguments to it. Note that this

is unlike sequential tagging tasks such as handwriting recognition, where an incorrect prediction of a token does not prohibit the correct prediction of the remaining ones. Both algorithms under comparison use the learned hypotheses from previous iterations in order to generate training examples (line 9 in Alg 1), therefore incorrectly predicted actions can inhibit the algorithm from reaching parts of the training data that could provide useful CSC examples. Thus, the learning rate which determines how frequently the optimal policy is queried is likely to be more important for biomedical event extraction than it was in the comparisons of Ross et al. (2011).

4. Experiments

In our experiments we train Searn and DAgger for 12 iterations and perform CSC learning using the online passive-aggressive (PA) algorithm (Crammer et al., 2006). BioNLP11ST-GE1 comprises three datasets – training, development and test – which consist of five full articles each and 800, 150 and 260 abstracts respectively. We extract features from the output of the parser by (McClosky, 2010) as provided by the shared task organizers (Stenetorp et al., 2011) and from the lemmatizer *morpha* (Minnen et al., 2001). While the gold standard is provided for the training and development datasets, evaluation on the test dataset is only possible once per day via a webserver in order to maintain the fairness of comparisons between systems. Vlachos and Craven (2012) evalutated the system described in Section 3 trained with Searn achieving the second-best reported performance on the test dataset. In ths work we focus on comparing the algorithms in terms of stability and learning speed, so we report results on the development set.

Initially we compare Searn and DAgger with learning rate equal to one. Figure 2(a) shows that the performance of DAgger peaks after 6 iterations beyond which it remains stable. On the other hand, the performance of Searn oscillates between high recall/low precision and low recall/high precision iterations (Figures 2(b) and 2(c)). In particular, in the first iteration Searn learns theme and cause assignment components given correctly identified triggers only (the optimal policy only returns those), but the trigger recognition component learned returns many incorrect ones, thus resulting in high recall and low precision. This behaviour is reversed in the second iteration, in which the theme and cause assignment components are learned so that they can accommodate for incorrectly recognized triggers, but at the same time the trigger recognition component becomes extremely conservative. This pattern holds for subsequent iterations, albeit progressively less pronounced. In contrast, DAgger can combine training signal from both iterations, thus its perfomance improves faster and avoids such oscillating behaviour. The unstable behaviour of Searn affects the training time as well, since in the high recall/low precision iterations the algorithm needs to consider many more actions during the cost-sensitive example generation steps (lines 8-17 in Alg. 1).

Figure 2(b) shows a substantial drop in recall for both algorithms in the second iteration. This is due to the learned hypothesis in the first iteration being unable to replicate the gold standard combined without using the optimal policy. This issue did not emerge in the experiments of Ross et al. (2011), but as explained in Section 3, event extraction is likely to be affected by it. Using slower learning rates ameliorates this problem (Figure 2(e)) and

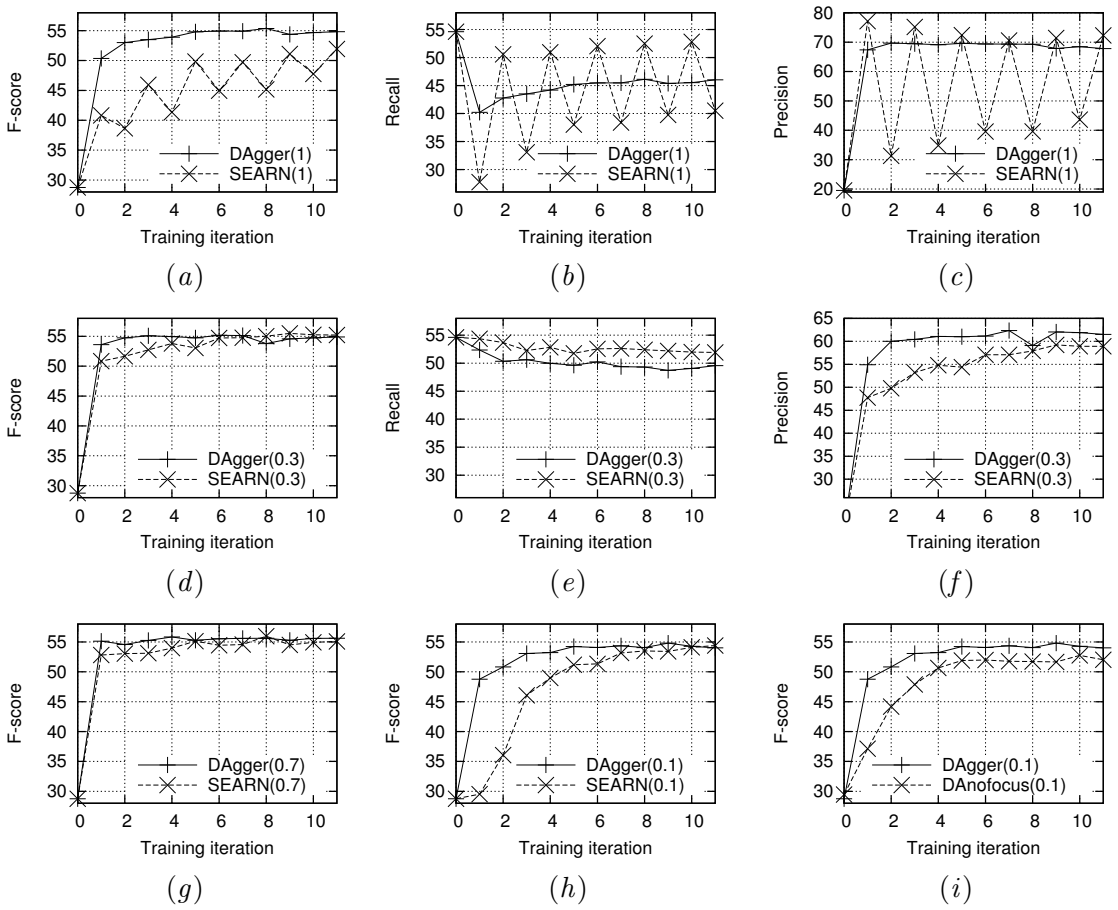


Figure 2: Development dataset results for DAgger(β) and Searn(β) with various learning rates.

renders Searn more stable, but DAgger still learns faster for a range of learning rates (0.7, 0.3 and 0.1 in Figures 2(d), 2(g) and 2(h) respectively).

Even though slower learning rates improve the performance for both Searn and DAgger, the improvement for the latter is not as dramatic (about 1.5 points in F-score). As discussed in Sec. 2, when $\beta < 1$ action costing becomes stochastic, which can result in unreliable estimates. In our experiments we used the *focused costing* approach proposed by Vlachos and Craven (2011), who reported that it improved the performance of Searn by 4 F-score points. In Figure 2(i) we compared the performance of DAgger with and without focused costing and we show that even though focused costing results in faster learning, the difference in terms of F-score is smaller, approximately 2 points. Even though the idea of focused costing is task-independent, it relies on the output structure consisting of disconnected components. While this is the case for tasks such as event extraction and named entity recognition, other structured prediction tasks such as dependency parsing have fully connected output spaces, thus focused costing might not be straightforward to

define. Other approaches to action costing proposed by Daumé III et al. (2009) include using multiple costing samples, and using the optimal policy to approximate the current policy. The former option increases the computational cost, while the latter is not helpful in complex problems where the learned hypothesis does not have near-optimal performance. Using DAgger with $\beta = 1$ sidesteps this issue while maintaining stability, thus it is more likely to be applicable to other structured prediction tasks.

5. Conclusions - Future work

In this paper we compared two imitation learning algorithms for structured prediction, Searn and DAgger. We presented them in a unified description and evaluated them on biomedical event extraction. We found that DAgger is more stable and learns faster, while being more robust with respect to the choice of learning rates and action costing. These advantages are more pronounced in the parameter-free versions of the algorithms which avoid stochastic cost estimates and need simpler optimal policy definitions. Finally, we assessed the effect of the learning rate in complex structured prediction tasks in which mistaken predictions can inhibit imitation learning algorithms from exploring useful parts of the training data. Our contributions should be relevant to applications of imitation learning to other structured prediction tasks.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 270019 (SpaceBook project www.spacebook-project.eu).

References

Pieter Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Department of Computer Science, Stanford University, 2008.

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, pages 46–53, 2004.

Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151, 2008.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009.

Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. Association for Computing Machinery, 1999.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. Overview of the Genia Event task in BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, pages 7–15, 2011.

David McClosky. *Any domain parsing: Automatic domain adaptation for natural language parsing*. PhD thesis, Department of Computer Science, Brown University, 2010.

Guido Minnen, John Carroll, and Darren Pearce. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223, 2001.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *14th International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, June 1999.

Pontus Stenetorp, Goran Topić, Sampo Pyysalo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. BioNLP Shared Task 2011: Supporting Resources. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, 2011.

Umar Syed. *Reinforcement learning without rewards*. PhD thesis, Department of Computer Science, Princeton University, 2010.

Umar Syed and Robert E. Schapire. Imitation learning with a value-based prior. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 384–391, 2007.

Andreas Vlachos and Mark Craven. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 49–57. Association for Computational Linguistics, 2011.

Andreas Vlachos and Mark Craven. Biomedical event extraction from abstracts and full papers using search-based structured prediction. *BMC Bioinformatics*, 13(suppl. 8):S5, 2012.