Biomedical Event Extraction from Abstracts and Full Papers using Search-based Structured Prediction

Andreas Vlachos and **Mark Craven**
Department of Biostatistics and Medical Informatics
University of Wisconsin-Madison
{vlachos,craven}@biostat.wisc.edu

Abstract

In this paper we describe our approach to the BioNLP 2011 shared task on biomedical event extraction from abstracts and full papers. We employ a joint inference system developed using the search-based structured prediction framework and show that it improves on a pipeline using the same features and it is better able to handle the domain shift from abstracts to full papers. In addition, we report on experiments using a simple domain adaptation method.

1 Introduction

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. The BioNLP 2011 shared task GENIA Task1 (BioNLP11ST-GE1) (Kim et al., 2011) focuses on extracting events from abstracts and full papers. The inclusion of full papers in the datasets is the only difference from Task1 of the BioNLP 2009 shared task (BioNLP09ST1) (Kim et al., 2009), which used the same task definition and abstracts dataset. Each event consists of a *trigger* and one or more *arguments*, the latter being proteins or other events. The protein names are annotated in advance and any token in a sentence can be a trigger for one of the nine event types. 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 nested events shown in Fig. 1d.

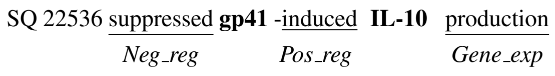
In our submission, we use the event extraction system of Vlachos and Craven (2011) which employs the search-based structured prediction framework (SEARN) (Daumé III et al., 2009). SEARN converts the problem of learning a model for structured prediction into learning a set of models for cost-sensitive classification (CSC). In CSC, each training instance has a vector of misclassification costs associated with it, thus rendering some mistakes in some instances to be more expensive than others. Compared to other structured prediction frameworks such as Markov Logic Networks (Poon and Vanderwende, 2010), SEARN provides high modeling flexibility but it does not requiring task-dependent approximate inference.

In this work, we show that SEARN is more accurate than a pipeline using the same features and it is better able to handle the domain shift from abstracts to full papers. Furthermore, we report on experiments with the simple domain adaptation method proposed by Daumé III (2007), which creates a version of each feature for each domain. While the results were mixed, this method improves our performance on full papers of the test set, for which little training data is available.

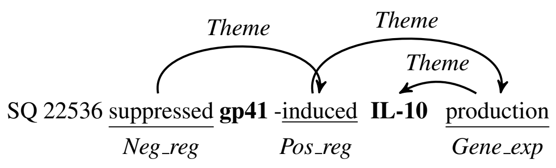
2 Event extraction decomposition

Figure 1 describes the event extraction decomposition that is used throughout the paper. Each stage has its own module to perform the classification needed.

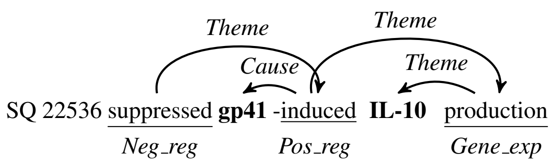
In trigger recognition the system decides whether a token acts as a trigger for one of the nine event types or not. We only consider tokens that are tagged as nouns, verbs or adjectives by the parser, as they



(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.

cover the majority of the triggers in the data. The main features used in the classifier represent the lemma of the token which is sufficient to predict the event type correctly in most cases. In addition, we include features that conjoin each lemma with its part-of-speech tag and its immediate lexical and syntactic context, which allows us to handle words that can represent different event types, e.g. "activity" often denotes a *Regulation* event but in "binding activity" it denotes a *Binding* event instead.

In *Theme* assignment, we form an agenda of candidate trigger-argument pairs for all trigger-protein combinations in the sentence and classify them as *Themes* or not. Whenever a trigger is predicted to be associated with a *Theme*, we form candidate pairs between all the *Regulation* triggers in the sentence and that trigger as the argument, thus allowing the prediction of nested events. Also, we remove candidate pairs that could result in directed cycles, as they are not allowed by the task. In *Cause* assignment, we form an agenda of candidate trigger-argument pairs and classify them as *Causes* or not. We form pairs between *Regulation* class triggers that were assigned at least one *Theme*, and protein names and other triggers that were assigned at least one *Theme*.

The features used in these two stages are extracted from the syntactic dependency path and the textual string between the trigger and the argument. We extract the shortest unlexicalized dependency path connecting each trigger-argument pair using Dijkstra's algorithm, allowing the paths to follow either dependency direction. One set of features represents the shortest unlexicalized path between the pair and in addition we have sets of features representing each path conjoined with the lemma, the PoS tag and the event type of the trigger, the type of the argument and the first and last lemmas in the dependency path.

In the event construction stage, we convert the predictions of the previous stages into events. If a *Binding* trigger is assigned multiple *Themes*, we choose to form either one event per *Theme* or one event with multiple *Themes*. For this purpose, we group the arguments of each nominal *Binding* trigger according to the first label in their dependency path and generate events using the cross-product of these groups. For example, assuming the parse was correct and all the *Themes* recognized, "interactions of **A** and **B** with **C**" results in two *Binding* events with two *Themes* each, **A** with **C**, and **B** with **C** respectively. We add the exceptions that if two *Themes* are part of the same token (e.g. "**A/B** interactions"), or the trigger and one of the *Themes* are part of the same token, or the lemma of the trigger is "bind" then they form one *Binding* event with two *Themes*.

3 Structured prediction with SEARN

SEARN (Daumé III et al., 2009) forms the structured output prediction of an instance s as a sequence of T multiclass predictions $\hat{y}_{1:T}$ made by a hypothesis h. The latter is a weighted ensemble of classifiers that are learned jointly. Each prediction $\hat{y}_t$ can use features from s as well as from all the previous predictions $\hat{y}_{1:t-1}$, thus taking structure into account. These predictions are referred to as *actions* and we adopt this term in order to distinguish them from the structured output predictions.

The SEARN algorithm is presented in Alg. 1. In each iteration, SEARN uses the current hypothesis h to generate a CSC example for each action $\hat{y}_t$ chosen to form the prediction for each labeled instance s (steps 6-12). The cost associated with each action is estimated using the gold standard according to a loss function l which corresponds to the task evaluation metric (step 11). Using a CSC learning algorithm, a new hypothesis h_{new} is learned (step 13) which is combined with the current one according to the interpolation parameter β (step 14). h is initialized to the *optimal policy* (step 2) which is derived from the gold standard. In each iteration SEARN "corrupts" the optimal policy with the learned hypotheses. Thus, each h_{new} is adapted to the actions chosen by h instead of the optimal policy. The algorithm terminates when the dependence on the latter becomes insignificant.

Algorithm 1 SEARN

```
 1: **Input:** labeled instances $\mathcal{S}$, *optimal policy* $\pi$, CSC
    learning algorithm *CSCL*, loss function $\ell$
 2: current policy $h = \pi$
 3: **while** $h$ depends significantly on $\pi$ **do**
 4:     Examples $E = \emptyset$
 5:     **for** $s$ **in** $S$ **do**
 6:         Predict $h(s) = \hat{y}_1 \dots \hat{y}_T$
 7:         **for** $\hat{y}_t$ **in** $h(s)$ **do**
 8:             Extract features $\Phi_t = f(s, \hat{y}_{1:t-1})$
 9:             **for each** possible action $y_t^i$ **do**
10:                 Predict $y\prime_{t+1:T} = h(s|\hat{y}_{1:t-1}, y_t^i)$
11:                 Estimate $c_t^i = \ell(\hat{y}_{1:t-1}, y_t^i, y\prime_{t+1:T})$
12:             Add $(\Phi_t, c_t)$ to $E$
13:     Learn a classifier $h_{new} = \mathit{CSCL}(E)$
14:     $h = \beta h_{new} + (1 - \beta)h$
15: **Output:** hypothesis $h$ without $\pi$
```

4 Biomedical event extraction with SEARN

In this section we describe how we learn the event extraction decomposition described in Sec. 2 under SEARN. Each instance is a sentence and the hypothesis learned in each iteration consists of a classifier for each stage of the pipeline, excluding event construction which is rule-based.

SEARN allows us to extract structural features for each action from the previous ones. During trigger recognition, we add as features the combination of the lemma of the current token combined with the event type (if any) assigned to the previous and the next token, as well as to the tokens that have syntactic dependencies with it. During *Theme* assignment, when considering a trigger-argument pair, we add features based on whether the pair forms an undirected cycle with previously predicted *Themes*, whether the trigger has been assigned a protein as a *Theme* and the candidate *Theme* is an event trigger (and the reverse), and whether the argument is the *Theme* of a trigger with the same event type. We also add a feature indicating whether the trigger has three *Themes* predicted already. During *Cause* assignment, we add features representing whether the trigger has been assigned a protein as a *Cause* and the candidate *Cause* is an event trigger.

Since the features extracted for an action depend on previous ones, we need to define a prediction order for the actions. In trigger recognition, we process the tokens from left to right since modifiers appearing before nouns tend to affect the meaning of the latter, e.g. "binding activity". In *Theme* and *Cause* assignment, we predict trigger-argument pairs in order of increasing dependency path length, assuming that, since they are the main source of features in these stages and shorter paths are less sparse, pairs containing shorter ones should be predicted more reliably. The loss function sums the number of false positive and false negative events, which is the evaluation measure of the shared task. The optimal policy is derived from the gold standard and returns the action that minimizes the loss over the sentence given the previous actions and assuming that all future actions are optimal.

In step 11 of Alg. 1, the cost of each action is estimated over the whole sentence. While this allows us to take structure into account, it can result in costs being affected by a part of the output that is not related to that action. This is likely to occur in event extraction, as sentences can often be long and contain disconnected event components in their output graphs. For this reason we use *focused costing* (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.

	pipeline (R/P/F)			SEARN (R/P/F)		
trigger	49.1	64.0	55.6	83.2	28.6	42.6
Theme	43.7	78.6	56.2	63.8	72.0	67.6
Cause	13.9	61.0	22.6	33.9	53.8	41.6
Event	31.7	70.1	43.6	45.8	60.51	52.1

Table 1: Results on the development dataset.

5 Experiments

In our experiments, we perform multiclass CSC learning using our implementation of the online passive-aggressive (PA) algorithm proposed by Crammer et al. (2006). The aggressiveness parameter and the number of rounds in parameter learning are set by tuning on 10% of the training data and we use the variant named PA-II with prediction-based updates. For SEARN, we set the interpolation parameter β to 0.3. For syntactic parsing, we use the output of the parser of Charniak and Johnson (2005) adapted to the biomedical domain by McClosky (2010), as provided by the shared task organizers in the Stanford collapsed dependencies with conjunct dependency propagation (Stenetorp et al., 2011). Lemmatization is performed using *morpha* (Minnen et al., 2001). No other knowledge sources or tools are used.

In order to assess the benefits of joint learning under SEARN, we compare it against a pipeline of independently learned classifiers using the same features and task decomposition. Table 1 reports the Recall/Precision/F-score achieved in each stage, as well as the overall performance. SEARN obtains better performance on the development set by 8.5 F-score points. This increase is larger than the 7.3 points reported in Vlachos and Craven (2011) on the BioNLP09ST1 datasets which contain only abstracts. This result suggests that the gains of joint inference under SEARN are greater when learning from the additional data from full papers. Note that while the classifier learned with SEARN overpredicts triggers, the *Theme* and *Cause* classifiers maintain relatively high precision with substantially higher recall as they are learned jointly with it. As triggers that do not form events are ignored by the evaluation, trigger overprediction without event overprediction does not result in performance loss.

The results of our submission on the test dataset using SEARN were 42.6/61.2/50.2 (Recall/Precision/F-score) which ranked sixth in the shared task. In the *Regulation* events which are considered harder due to nesting, our submission was ranked fourth. This demonstrates the potential of SEARN for structured prediction, as the performance on regulation events depends partly on the performance on the simple ones on which our submission was ranked eighth.

After the end of the shared task, we experimented with the domain adaptation method proposed by Daumé III (2007), which creates multiple versions for each feature by conjoining it with the domain label of the instance it is extracted from (abstracts or full papers). While this improved the performance of the pipeline baseline by 0.3 F-score points, the performance under SEARN dropped by 0.4 points on the development data. Using the online service provided by the organizers, we evaluated the performance of the domain adapted SEARN-based system on the test set and the overall performance improved to 50.72 in F-score (would have ranked 5th). In particular, domain adaptation improved the performance on full papers by 1.22 points, thus reaching 51.22 in F-score. This version of the system would have ranked 3rd overall and 1st in the *Regulation* events in this part of the corpus. We hypothesize that these mixed results are due to the sparse features used in the stages of the event extraction decomposition, which become even sparser using this domain adaptation method, thus rendering the learning of appropriate weights for them harder.

6 Conclusions

We presented a joint inference approach to the BioNLP11ST-GE1 task using SEARN which converts a structured prediction task into a set of CSC tasks whose models are learned jointly. Our results demonstrate that SEARN achieves substantial performance gains over a standard pipeline using the same features.

Acknowledgments

The authors were funded by NIH/NLM grant R01 LM07050.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180.

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

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

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 256–263.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9.

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

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

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

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821.

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

Andreas Vlachos and Mark Craven. 2011. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*.