Fast and Robust Joint Models for Biomedical Event Extraction

Sebastian Riedel **Andrew McCallum**
Department of Computer Science
University of Massachusetts, Amherst
{riedel,mccallum}@cs.umass.edu

Abstract

Extracting biomedical events from literature has attracted much recent attention. The best-performing systems so far have been pipelines of simple subtask-specific local classifiers. A natural drawback of such approaches are cascading errors introduced in early stages of the pipeline. We present three joint models of increasing complexity designed to overcome this problem. The first model performs joint trigger and argument extraction, and lends itself to a simple, efficient and exact inference algorithm. The second model captures correlations between events, while the third model ensures consistency between arguments of the same event. Inference in these models is kept tractable through dual decomposition. The first two models outperform the previous best joint approaches and are very competitive with respect to the current state-of-the-art. The third model yields the best results reported so far on the BioNLP 2009 shared task, the BioNLP 2011 Genia task and the BioNLP 2011 Infectious Diseases task.

1 Introduction

Whenever we advance our scientific understanding of the world, we seek to publish our findings. The result is a vast and ever-expanding body of natural language text that is becoming increasingly difficult to leverage. This is particularly true in the context of life sciences, where large quantities of biomedical articles are published on a daily basis. To support tasks such data mining, search and visualization, there is a clear need for structured representations of the knowledge these articles convey. This is indicated by a large number of public databases with content ranging from simple protein-protein interactions to complex pathways. To increase coverage of such databases, and to keep up with the rate of publishing, we need to *automatically* extract structured representations from biomedical text—a process often referred to as biomedical text mining.

One major focus of biomedical text mining has been the extraction of named entities, such genes or gene products, and of flat binary relations between such entities, such as protein-protein interactions. However, in recent years there has also been an increasing interest in the extraction of biomedical *events* and their causal relations. This gave rise to the BioNLP 2009 and 2011 shared tasks which challenged participants to gather such events from biomedical text (Kim et al., 2009; Kim et al., 2011). Notably, these events can be complex and recursive: they may have several arguments, and some of the arguments may be events themselves.

Current state-of-the-art event extractors follow the same architectural blueprint and divide the extraction process into a pipeline of three stages (Björne et al., 2009; Miwa et al., 2010c). First they predict a set of candidate event trigger words (say, tokens 2, 5 and 6 in figure 1), then argument mentions are attached to these triggers (say, token 4 for trigger 2). The final stage decides how arguments are shared between events—compare how one event subsumes all arguments of trigger 6 in figure 1, while two events share the three arguments of trigger 4 in figure 2. This architecture is prone to cascading errors: If we miss a trigger in the first stage, we will never be able to extract the full event

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1–12,
Edinburgh, Scotland, UK, July 27–31, 2011.

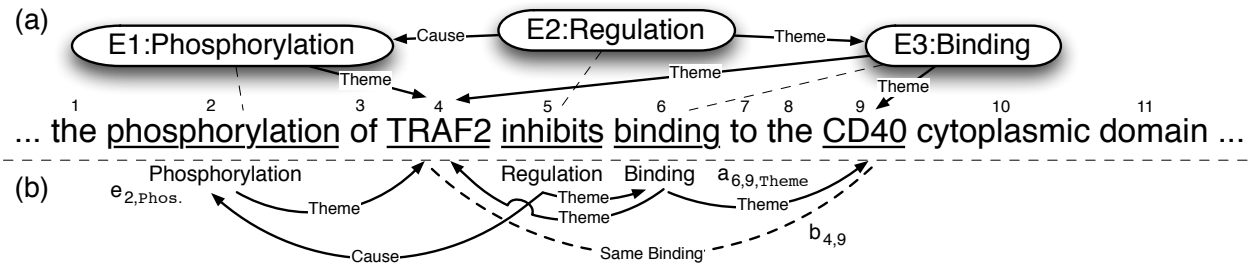

Figure 1: (a) sentence with target event structure to extract; (b) projection to a set of labelled graph over tokens.

it concerns. Current systems attempt to tackle this problem by passing several candidates to the next stage. However, this tends to increase the false positive rate. In fact, Miwa et al. (2010c) observe that 30% of their errors stem from this type of ad-hoc module communication.

Joint models have been proposed to overcome this problem (Poon and Vanderwende, 2010; Riedel et al., 2009). However, besides not being as accurate as their pipelined competitors, mostly because they do not yet exploit the rich set of features used by Miwa et al. (2010b) and Björne et al. (2009), they also suffer from the complexity of inference. For example, to remain tractable, the best joint system so far (Poon and Vanderwende, 2010) works with a simplified representation of the problem in which certain features are harder to capture, employs local search without certificates of optimality, and furthermore requires a 32-core cluster for quick train-test cycles. Existing joint models also rely on heuristics when it comes to deciding which arguments share the same event. Contrast this with the best current pipeline (Miwa et al., 2010c; Miwa et al., 2010b) which uses a classifier for this task.

We present a family of event extraction models that address the aforementioned problems. The first model jointly predicts triggers and arguments. Notably, the highest scoring event structure under this model can be found efficiently in $O(mn)$ time where m is the number of trigger candidates, and n the number of argument candidates. This is only slightly slower than the $O(m'n)$ runtime of a pipeline, where m' is the number of trigger candidates as filtered by the first stage. We achieve these guarantees through a novel algorithm that jointly picks best trigger label and arguments on a per-token basis. Remarkably, it takes roughly as much time to train this model on one core as the model of Poon and Vanderwende (2010) on 32 cores, and leads to better results.

The second model enforces additional constraints that ensure consistency between events in hierarchical regulation structures. While inference in this model is more complicated, we show how dual decomposition (Komodakis et al., 2007; Rush et al., 2010) can be used to efficiently find exact solutions for a large fraction of problems.

Our third model includes the first two, and explicitly captures which arguments are part in the same event—the third stage of existing pipelines. Due to a complex coupling between this model and the first two, inference here requires a *projected* version of the sub-gradient technique demonstrated by Rush et al. (2010).

When evaluated on the BioNLP 2009 shared task, the first two models outperform the previous best joint approaches and are competitive when compared to current state-of-the-art. With 57.4 F1 on the test set, the third model yields the best results reported so far with a 1.1 F1 margin to the results of Miwa et al. (2010b). For the BioNLP 2011 Genia task 1 and the BioNLP 2011 Infectious Diseases task, Model 3 yields the second-best and best results reported so far. The second-best results are achieved with Model 3 as is (Riedel and McCallum, 2011), the best results when using Stanford event predictions as input features (Riedel et al., 2011). The margins between Model 3 and the best runner-ups range from 1.9 F1 to 2.8 F1.

In the following we will first introduce biomedical event extraction and our notation. Then we go on to present our models and their inference routines. We present related work, show our empirical evaluation, and conclude.

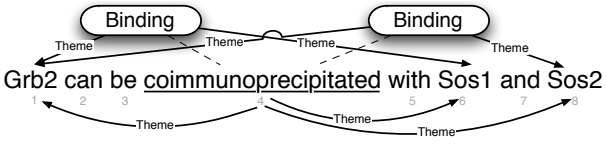

Figure 2: Two binding events with identical trigger. The projection graph does not change even if both events are merged.

2 Biomedical Event Extraction

By bio-molecular event we mean a change of state of one or more bio-molecules. Our task is to extract structured information about such events from natural language text. More concretely, let us consider part (a) of figure 1. We see a snippet of text from a biomedical abstract, and the three events that can be extracted from it. We will use these to characterize the types of events we ought to extract, as defined by the 2009 BioNLP shared task. Note that for the shared task, protein mentions are given by the task organizers and hence do not need to be extracted.

The event E1 in the figure refers to a *Phosphorylation* of the TRAF2 protein. It is an instance of a set of *simple events* that describe changes to a single gene or gene product. Other members of this set are: *Expression*, *Transcription*, *Localization*, and *Catabolism*. Each of these events has to have exactly one *theme*, the protein of which a state change is described. A labelled edge in figure 1a) shows that TRAF2 is the theme of E1.

Event E3 is a *Binding* of TRAF2 and CD40. Binding events are particular in that they may have more than one theme, as there can be several biomolecules associated in a binding structure. This is in fact the case for E3.

In the top-center of figure 1a) we see the *Regulation* event E2. Such events describe regulatory or causal relations between events. Other instances of this type of events are: *Positive Regulation* and *Negative Regulation*. Regulations have to have exactly one theme; this theme can a be protein or, as in our case, another event. Regulations may also have zero or one *cause* arguments that denote events or proteins which trigger the regulation.

In the BioNLP shared task, we are also asked to find a *trigger* (or *clue*) token for each event. This token grounds the event in text and allows users to quickly validate extracted events. For example, the trigger for event E2 is "inhibit", as indicated by a dashed line.

2.1 Event Projection

To formulate the search for event structures of the form shown in figure 1a) as an optimization problem, it will be convenient to represent them through a set of binary variables. We introduce such a representation, inspired by previous work (Riedel et al., 2009; Björne et al., 2009) and based on a projection of events to a graph structure over tokens, as seen figure 1b).

Consider sentence $\mathbf{x}$ and a set of *candidate trigger* tokens, denoted by $\text{Trig}(\mathbf{x})$. We label each candidate i with the event type it is a trigger for, or `None` if it is not a trigger. This decision is represented through a set of binary variables $e_{i,t}$, one for each possible event type t. In our example we have $e_{6,\texttt{Binding}} = 1$. The set of possible event types will be denoted as $\mathcal{T}$, the regulation event types as $\mathcal{T}_{\text{Reg}} \stackrel{\text{def}}{=} \{\texttt{PosReg}, \texttt{NegReg}, \texttt{Reg}\}$ and its complement as $\mathcal{T}_{\neg\text{reg}} \stackrel{\text{def}}{=} \mathcal{T} \setminus \mathcal{T}_{\text{Reg}}$.

For each candidate trigger i we consider the arguments of all events that have i as trigger. Each argument a will either be an event itself, or a protein. For events we add a labelled edge between i and the trigger j of a. For proteins we add an edge between i and the syntactic head j of the protein mention. In both cases we label the edge $i \rightarrow j$ with the role of the argument a. The edge is represented through a binary variable $a_{i,j,r}$, where $r \in \mathcal{R}$ is the argument role and $\mathcal{R} \stackrel{\text{def}}{=} \{\texttt{Theme}, \texttt{Cause}, \texttt{None}\}$. The role `None` is active whenever no `Theme` or `Cause` role is present. In our example we get, among others, $a_{2,4,\texttt{Theme}} = 1$.

So far our representation is equivalent to mappings in previous work (Riedel et al., 2009; Björne et al., 2009) and hence shares their main shortcoming: we cannot differentiate between two (or more) binding events with the same trigger but different arguments, or one binding event with several arguments. Consider, for example, the arguments of trigger 6 in figure 1b) that are all subsumed in a single event. By contrast, the arguments of trigger 4 shown in figure 2 are split between two events.

Previous work has resolved this ambiguity

through ad-hoc rules (Björne et al., 2009) or with a post-processing classifier (Miwa et al., 2010c). We propose to augment the graph representation through edges between pairs of proteins that are themes in the same binding event. For two protein tokens p and q we represent this edge through the binary variable $b_{p,q}$. Hence, in figure 1b) we have $b_{4,9} = 1$, whereas for figure 2 we get $b_{1,6} = b_{1,8} = 1$ but $b_{6,8} = 0$. By explicitly modeling such "sibling" edges we not only minimize the need for post-processing. We can also improve attachment decisions akin to second order models in dependency parsing (McDonald and Pereira, 2006). Note that while merely introducing such variables is easy, enforcing consistency between them and the $e_{i,t}$ and $a_{i,j,r}$ variables is not. We address this in section 3.3.1.

Reconstruction of events from solutions $(\mathbf{e}, \mathbf{a}, \mathbf{b})$ can be done almost exactly as described by Björne et al. (2009). However, while they group binding arguments according to ad-hoc rules based on dependency paths from trigger to argument, we simply query the variables $b_{p,q}$.

To simplify our exposition we introduce additional notation. We denote the set of protein head tokens with $\text{Prot}(\mathbf{x})$; the set of a possible targets for outgoing edges from a trigger is $\text{Cand}(\mathbf{x}) \stackrel{\text{def}}{=} \text{Trig}(\mathbf{x}) \cup \text{Prot}(\mathbf{x})$. We will often omit the domains of indices and instead assign them a fixed domain in advance: $i, l \in \text{Trig}(\mathbf{x})$, $j, k \in \text{Cand}(\mathbf{x})$, $p, q \in \text{Prot}(\mathbf{x})$, $r \in \mathcal{R}$ and $t \in \mathcal{T}$. Bold face letters are used to denote composite vectors $\mathbf{e}$, $\mathbf{a}$ and $\mathbf{b}$ of variables $e_{i,t}$, $a_{i,j,r}$ and $b_{p,q}$. The vector $\mathbf{y}$ is the joint vector of $\mathbf{e}, \mathbf{a}$ and $\mathbf{b}$. The short-form $\mathbf{e}_i \leftarrow t$ will mean $\forall t' : e_{i,t'} \leftarrow \delta_{t,t'}$ where $\delta_{t,t'}$ is the Kronecker Delta. Likewise, $\mathbf{a}_{i,j} \leftarrow r$ means $\forall r' : a_{i,j,r'} \leftarrow \delta_{r,r'}$.

3 Models

In this section we will present three structured prediction models of increasing complexity and expressiveness, as well as their corresponding MAP inference algorithms. Each model m can be represented by a mapping from sentence $\mathbf{x}$ to a set of *legal* structures $\mathcal{Y}_m(\mathbf{x})$, and a linear *scoring function*

$$s_m(\mathbf{y}; \mathbf{x}, \mathbf{w}) = \langle \mathbf{w}, \mathbf{f}(\mathbf{y}, \mathbf{x}) \rangle . \quad (1)$$

Here $\mathbf{f}$ is a feature function on structures $\mathbf{y}$ and input $\mathbf{x}$, and $\mathbf{w}$ is a weight vector for these features.

We can use the scoring function s_m and the set of legal structures $\mathcal{Y}_m(\mathbf{x})$ to predict the event $\mathbf{h}_m(\mathbf{x})$ for a given sentence $\mathbf{x}$ according to

$$\mathbf{h}_m(\mathbf{x}) \stackrel{\text{def}}{=} \arg \max_{\mathbf{y} \in \mathcal{Y}_m(\mathbf{x})} s_m(\mathbf{y}; \mathbf{x}, \mathbf{w}) . \quad (2)$$

For brevity we will from now on omit observations $\mathbf{x}$ and weights $\mathbf{w}$ when they are clear from the context.

3.1 Model 1

Model 1 performs a simple version of joint trigger and argument extraction. It independently scores trigger labels and argument roles:

$$s_1(\mathbf{e}, \mathbf{a}) \stackrel{\text{def}}{=} \sum_{e_{i,t}=1} s_{\text{T}}(i, t) + \sum_{a_{i,j,r}=1} s_{\text{R}}(i, j, r) . \quad (3)$$

Here $s_{\text{T}}(i, t) = \langle \mathbf{w}_{\text{T}}, \mathbf{f}_{\text{T}}(i, t) \rangle$ is a per-trigger scoring function that measures how well the event label t fits to token i. Likewise, $s_{\text{R}}(i, j, r) = \langle \mathbf{w}_{\text{R}}, \mathbf{f}_{\text{R}}(i, j, r) \rangle$ measures the compatibility of role r as label for the edge $i \rightarrow j$.

The jointness of Model 1 stems from enforcing consistency between the trigger label of i and its *outgoing* edges. By consistency we mean that: (a) there is at least one `Theme` whenever there is an event at i; (b) only regulation events are allowed to have `Cause` arguments; (c) all arguments of a `None` trigger must have the `None` role. We will denote the set assignments that fulfill these constraints by O and hence have $\mathcal{Y}_1 \stackrel{\text{def}}{=} \text{O}$.

Enforcing $(\mathbf{e}, \mathbf{a}) \in \text{O}$ guarantees that we never predict triggers i for which no sensible, high-scoring, argument j can be found. It also ensures that when we see an "obvious" argument edge $i \stackrel{r}{\rightarrow} j$ with high score $s_{\text{R}}(i, j, r)$ there is pressure to extract a trigger at i, even if the fact that i is a trigger may not be as obvious.

3.1.1 Inference

As it turns out, the maximizer of equation 2 can be found very efficiently in $O(mn)$ time where $m = |\text{Trig}(\mathbf{x})|$ and $n = |\text{Cand}(\mathbf{x})|$. The corresponding procedure, bestOut$(\cdot)$, is shown in algorithm 1. It takes as input a vector of trigger and edge *penalties* $\mathbf{c}$ that are added to the local scores of the s_{T} and s_{R} functions. For Model 2 and 3 we will use these

penalties to enforce agreement with predictions of other inference subroutines. When using Model 1 by itself we set them to $\mathbf{0}$. We point out that the scoring function s_1 is multiplied with $\frac{1}{2}$ throughout the algorithm. For doing inference in Model 1 and 2 this has no effect, but when we use bestOut$(\cdot)$ for Model 3 inference, it is required.

The bestOut $(\mathbf{c})$ routine exploits the fact that the constraints of Model 1 only act on the label for trigger i and its outgoing edges. In particular, enforcing consistency between $e_{i,t}$ and outgoing edges $a_{i,j,r}$ has no effect on consistency between $e_{l,t}$ and $a_{i',j',r'}$ for any other trigger $i' \neq i$. Moreover, for a given trigger the constraints only differentiate between three cases: (a) regulation event, (b) non-regulation event and (c) no event. This means that we can extract events on a per-trigger basis, and find the best per-trigger structure by comparing cases (a), (b) and (c). Note that bestOut $(\mathbf{c})$ uses the shorthand emptyOut (i) to denote the partial assignment $\mathbf{e}_i \leftarrow \texttt{None}$ and $\forall j : \mathbf{a}_{i,j} \leftarrow \texttt{None}$. The function $s_1^{\mathbf{c}}(i, \mathbf{y}) \stackrel{\text{def}}{=} \sum_t e_{i,t} \left(c_{i,t} + \frac{1}{2} s_{\text{T}}(i,t)\right) + \sum_{j,r} a_{i,j,r} \left(c_{i,j,r} + \frac{1}{2} s_{\text{R}}(i,j,r)\right)$ is a per-trigger frame score with penalties $\mathbf{c}$.

3.2 Model 2

Model 1 may still predict structures that cannot be mapped to events. For example, in figure 1b) we may label token 5 as `Regulation`, add the edge $5 \stackrel{\texttt{Cause}}{\rightarrow} 2$ but fail to label token 2 as an event. While consistent with $(\mathbf{e}, \mathbf{a}) \in \text{O}$, this violates the constraint that every active edge must either end at a protein, or at an active event trigger. This is a requirement on the label of a trigger and the assignment of roles for its *incoming* edges.

Model 2 enforces the above constraint in addition to $(\mathbf{e}, \mathbf{a}) \in \text{O}$, while inheriting the scoring function from Model 1. Hence, using I to denote the set of assignments with consistent trigger labels and incoming edges, we get $\mathcal{Y}_2 \stackrel{\text{def}}{=} \mathcal{Y}_1 \cap \text{I}$ and $s_2(\mathbf{y}) \stackrel{\text{def}}{=} s_1(\mathbf{y})$.

3.2.1 Inference

Inference in Model 2 amounts to optimizing $s_2(\mathbf{e}, \mathbf{a})$ over $\text{O} \cap \text{I}$. This is more involved, as we now have to ensure that when predicting an outgoing edge from trigger i to trigger l there is a high-scoring event at l. We follow Rush et al. (2010) and solve this problem in the framework of dual decomposi-

Algorithm 1 Sub-procedures for inference in Model 1, 2 and 3.

best label and outgoing edges for all triggers under penalties $\mathbf{c}$

bestOut $(\mathbf{c})$:
  $\forall i$ $\mathbf{y}^0 \leftarrow$ emptyOut (i)
    $\mathbf{y}^1 \leftarrow$ out $\left(i, \mathbf{c}, \mathcal{T}_{\text{reg}}, \mathcal{R}\right)$
    $\mathbf{y}^2 \leftarrow$ out $\left(i, \mathbf{c}, \mathcal{T}_{\neg\text{reg}}, \mathcal{R} \setminus \{\texttt{Cause}\}\right)$
    $\mathbf{y}_i \leftarrow \arg\max_{\mathbf{y} \in \{\mathbf{y}^0, \mathbf{y}^1, \mathbf{y}^2\}} s_1^{\mathbf{c}}(i, \mathbf{y})$
  return $(\mathbf{y}_i)_i$

best label and incoming edges for all triggers under penalties $\mathbf{c}$

bestIn $(\mathbf{c})$:
  $\forall l$ $\mathbf{y}^0 \leftarrow$ emptyIn (l)
    $\mathbf{y}^1 \leftarrow$ in $(l, \mathbf{c}, \mathcal{T}, \mathcal{R} \setminus \{\texttt{None}\})$
    $\mathbf{y}_l \leftarrow \arg\max_{\mathbf{y} \in \{\mathbf{y}^0, \mathbf{y}^1\}} s_2^{\mathbf{c}}(l, \mathbf{y})$
  return $(\mathbf{y}_l)_l$

pick best binding pairs p, q *and trigger* i *for each using penalties* $\mathbf{c}$

bestBind $(\mathbf{c})$:
  $\forall p, q$ $b_{p,q} \leftarrow [s_{\text{B}}(p,q) + \max_i c_{i,p,q} > 0]$
    $I_{p,q} \leftarrow \left\{i | c_{i,p,q} = \max_{i'} c_{i',p,q}\right\}$
    if $b_{p,q} = 1$ **or** $\max_{i'} c_{i',p,q} > 0$
      $\forall i : t_{i,p,q} \leftarrow [i \in I_{p,q}] \, |I_{p,q}|^{-1}$
    else
      $\forall i : t_{i,p,q} \leftarrow 0$
  return $(\mathbf{b}, \mathbf{t})$

best label in T *and outgoing edge roles in* R *for* i*, using penalties* $\mathbf{c}$

out $(i, \mathbf{c}, T, R)$:
  $\mathbf{e}_i \leftarrow \arg\max_{t \in T} \frac{1}{2} s_{\text{T}}(i,t) + c_{i,t}$
  $\mathbf{a}_{i,\text{bestTheme}(i,\mathbf{c})} \leftarrow \texttt{Theme}$
  $\forall j$ $\mathbf{a}_{i,j} \leftarrow \arg\max_{r \in R} \frac{1}{2} s_{\text{R}}(i,j,r) + c_{i,j,r}$
  return $(\mathbf{e}_i, \mathbf{a}_i)$

best label in T*, incoming edge roles in* R
and outgoing protein roles, using costs $\mathbf{c}$

in $(l, \mathbf{c}, T, R)$:
  $\mathbf{e}_l \leftarrow \arg\max_{t \in T} \frac{1}{2} s_{\text{T}}(l,t) + c_{l,t}$
  $\forall i$ $\mathbf{a}_{i,l} \leftarrow \arg\max_{r \in R} \frac{1}{2} s_{\text{R}}(i,l,r) + c_{i,l,r}$
  $\forall p$ $\mathbf{a}_{l,p} \leftarrow \arg\max_{r \in R} \frac{1}{2} s_{\text{R}}(l,p,r) + c_{l,p,r}$
  return $(\mathbf{e}_i, \mathbf{a}_i)$

best `Theme` *argument for* i

bestTheme $(i, \mathbf{c})$:
  $\text{s}(j) \stackrel{\text{def}}{=} \max_{j,r} \frac{1}{2} s_{\text{R}}(i,j,r) + c_{i,j,r}$
  $\Delta(j) \stackrel{\text{def}}{=} \frac{1}{2} s_{\text{R}}(i,j,\texttt{Theme}) + c_{i,j,\texttt{Theme}} - \text{s}(j)$
  return $\arg\max_j \Delta(j)$

tion. To this end we write our optimization problem as

$$\begin{aligned} \underset{\mathbf{e},\mathbf{a},\bar{\mathbf{e}},\bar{\mathbf{a}}}{\text{maximize}} \quad & \frac{1}{2} s_2(\mathbf{e},\mathbf{a}) + \frac{1}{2} s_2(\bar{\mathbf{e}},\bar{\mathbf{a}}) \\ \text{subject to} \quad & (\mathbf{e},\mathbf{a}) \in \mathrm{O} \wedge (\bar{\mathbf{e}},\bar{\mathbf{a}}) \in \mathrm{I} \wedge \\ & \mathbf{e} = \bar{\mathbf{e}} \wedge \mathbf{a} = \bar{\mathbf{a}} \end{aligned} \tag{M2}$$

and note that this problem could be solved separately for $\mathbf{e}, \mathbf{a}$ and $\bar{\mathbf{e}}, \bar{\mathbf{a}}$ if the coupling constraints $\mathbf{e} = \bar{\mathbf{e}}$ and $\mathbf{a} = \bar{\mathbf{a}}$ were removed.

M2 is an Integer Linear Program, as variables are binary and both objective and constraints can be represented through linear constraints.[1] Dual decomposition solves a Linear Programming (LP) relaxation of M2 (that allows fractional values for all binary variables) through subgradient descent on a particular dual of M2. This dual can be derived by introducing Lagrange multipliers for the coupling constraints. Its attractiveness stems from the fact that calculating the subgradient amounts to solving the decoupled problems in isolation. If, by design, these decoupled problems can be solved efficiently, we can often quickly find the optimal solution to an LP relaxation of our original problem.

Dual decomposition applied to Model 2 is shown in algorithm 2. It maintains the dual variables $\boldsymbol{\lambda}$ that will appear as local penalties in the subproblems to be solved. The algorithm will try to tune these variables such that at convergence the coupling constraints will be fulfilled. This is done by first optimizing $s_2(\mathbf{e},\mathbf{a})$ over O and $s_2(\bar{\mathbf{e}},\bar{\mathbf{a}})$ over I. Now, whenever there is disagreement between two variables to be coupled, the corresponding dual parameter is shifted, increasing the chance that next time both models will agree. For example, if in the first iteration we predict $e_{6,\texttt{Bind}} = 1$ but $\bar{e}_{6,\texttt{Bind}} = 0$, we set $\lambda_{6,\texttt{Bind}} = -\alpha$ where α is some stepsize (chosen according to Koo et al. (2010)). This will decrease the coefficient for $e_{6,\texttt{Bind}}$, and increase the coefficient for $\bar{e}_{6,\texttt{Bind}}$. Hence, we have a higher chance of agreement for this variable in the next iteration.

The algorithm repeats the process described above until all variables agree, or some predefined number R of iterations is reached. In the former case we in fact have the exact solution to the original ILP.

[1]The ILP representation could be taken from the MLNs of Riedel et al. (2009) and the mapping to ILPs of Riedel (2008).

Algorithm 2 Subgradient descent for Model 2, and projected subgradient descent for Model 3.

require:
R: *max. iteration,* α_t: *stepsizes*
$t \leftarrow 0$ *[model 2,3]* $\boldsymbol{\lambda} \leftarrow 0$ *[model 2,3]* $\boldsymbol{\mu} \leftarrow 0$ *[model 3]*
repeat

model	
2	$(\mathbf{e},\mathbf{a}) \leftarrow \text{bestOut}(\boldsymbol{\lambda})$
2,3	$(\bar{\mathbf{e}},\bar{\mathbf{a}}) \leftarrow \text{bestIn}(-\boldsymbol{\lambda})$
3	$(\mathbf{e},\mathbf{a}) \leftarrow \text{bestOut}(\mathbf{c}^{\text{out}}(\boldsymbol{\lambda},\boldsymbol{\mu}))$
3	$(\mathbf{b},\mathbf{t}) \leftarrow \text{bestBind}\left(\mathbf{c}^{\text{bind}}(\boldsymbol{\mu})\right)$
2,3	$\lambda_{i,t} \leftarrow \lambda_{i,t} - \alpha_t (e_{i,t} - \bar{e}_{i,t})$
2,3	$\lambda_{i,j,r} \leftarrow \lambda_{i,j,r} - \alpha_t (a_{i,j,r} - \bar{a}_{i,j,r})$
3	$\mu^{\text{trig}}_{i,p,q} \leftarrow \left[\mu^{\text{trig}}_{i,p,q} - \alpha_t (e_{i,\texttt{Bind}} - t_{i,p,q})\right]_+$
3	$\mu^{\text{arg1}}_{i,j,k} \leftarrow \left[\mu^{\text{arg1}}_{i,p,q} - \alpha_t (a_{i,p,\texttt{Theme}} - t_{i,p,q})\right]_+$
3	$\mu^{\text{arg2}}_{i,p,q} \leftarrow \left[\mu^{\text{arg2}}_{i,p,q} - \alpha_t (a_{i,q,\texttt{Theme}} - t_{i,p,q})\right]_+$
2,3	$t \leftarrow t + 1$

until no $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ changed or $t > R$
return $(\mathbf{e},\mathbf{a})$*[model 2]* or $(\mathbf{e},\mathbf{a},\mathbf{b})$ *[model 3]*

In the later case we have no such guarantee, but find that in practice the solutions are still of high quality. Notice that we could still assess the quality of this approximation by measuring the duality gap between primal score and the final dual score.

Algorithm 2 for Model 2 requires us to optimize $s_2(\mathbf{e},\mathbf{a})$ over O and $s_2(\bar{\mathbf{e}},\bar{\mathbf{a}})$ over I. The former, with added penalties, can be done with bestOut($\mathbf{c}$). As the constraint set for I again decomposes on a per-token basis, solving the latter problem requires a very similar procedure, and again $O(mn)$ time. Algorithm 1 shows this procedure under bestIn($\mathbf{c}$). It chooses, for each trigger candidate, the best label and *incoming* set of arguments together with the best outgoing edges to proteins. Adding edges to proteins is not strictly required, but simplifies our exposition. Algorithm bestIn($\mathbf{c}$) requires a per-trigger incoming score: $s^{\mathbf{c}}_2(l,\mathbf{y}_l) \stackrel{\text{def}}{=} \sum_t e_{l,t}\left(c_{l,t} + \frac{1}{2} s_{\mathrm{T}}(l,t)\right) + \sum_{i,r} a_{i,l,r}\left(c_{i,l,r} + \frac{1}{2} s_{\mathrm{R}}(i,l,r)\right) + \sum_{p,r} a_{l,p,r}\left(c_{l,p,r} + \frac{1}{2} s_{\mathrm{R}}(l,p,r)\right)$. Finally, note that emptyIn(i) not only assigns None as trigger label of i and to all incoming edges, but also greedily picks outgoing protein edges (as done within in$(\cdot)$).

3.3 Model 3

Model 2 does not predict the $b_{p,q}$ variables that represent protein pairs p, q in bindings. *Model 3* fixes this by (a) adding binding variables $b_{p,q}$ into the objective, and (b) enforcing that the binding assignment $\mathbf{b}$ is consistent with the trigger and argument assignments $\mathbf{e}$ and $\mathbf{a}$. We will also enforce that the same pair of entities p, q cannot be arguments in more than one event together.

The scoring function for Model 3 is simply

$$s_3(\mathbf{e}, \mathbf{a}, \mathbf{b}) \stackrel{\text{def}}{=} s_2(\mathbf{e}, \mathbf{a}, \mathbf{b}) + \sum_{b_{p,q}=1} s_{\text{B}}(p, q). \quad (4)$$

Here $s_{\text{B}}(p, q) = \langle \mathbf{w}_{\text{B}}, \mathbf{f}_{\text{B}}(p, q) \rangle$ is a per-protein-pair score based on a feature representation of the lexical and syntactic relation between both protein heads.

Our strategy will be based on enforcing consistency partly through linear constraints which we dualize, and partly within our search algorithm. To this end we first introduce a set of auxiliary binary variables $t_{i,p,q}$. When a $t_{i,p,q}$ is active, we enforce that there is a binding trigger at i with proteins p and q as `Theme` arguments. A set of linear constraints can be used for this: $e_{i,\texttt{Bind}} - t_{i,p,q} \geq 0$, $a_{i,p,\texttt{Theme}} - t_{i,p,q} \geq 0$ and $a_{i,q,\texttt{Theme}} - t_{i,p,q} \geq 0$ for all suitable i, p and q. We denote the set of assignments $(\mathbf{e}, \mathbf{a}, \mathbf{t})$ that fulfill these constraints by T.

Consistency between $\mathbf{e}$, $\mathbf{a}$ and $\mathbf{b}$ can now be enforced by making sure that $\mathbf{t}$ is consistent with $\mathbf{e}$ and $\mathbf{a}$, and that $\mathbf{b}$ is consistent with this $\mathbf{t}$. The latter means that an active $b_{p,q}$ requires a trigger i to point to p and q. Or in other words, $t_{i,p,q} = 1$ for exactly one trigger i.

With the set of consistent assignments $(\mathbf{b}, \mathbf{t})$ referred to as B, and a slight abuse of notation, this gives us $\mathcal{Y}_3 \stackrel{\text{def}}{=} \mathcal{Y}_2 \cap \text{T} \cap \text{B}$. Note that it is $(\mathbf{e}, \mathbf{a}, \mathbf{t}) \in \text{T}$ that will be enforced by dualizing constraints, and $(\mathbf{b}, \mathbf{t}) \in \text{B}$ that will be enforced within search.

3.3.1 Inference

We note that inference in Model 3 can be performed by solving the following problem:

$$\begin{aligned} \underset{\mathbf{e},\mathbf{a},\bar{\mathbf{e}},\bar{\mathbf{a}},\mathbf{b},\mathbf{t}}{\text{maximize}} \quad & \frac{1}{2} s_1(\mathbf{e}, \mathbf{a}) + \frac{1}{2} s_2(\bar{\mathbf{e}}, \bar{\mathbf{a}}) + \sum_{b_{p,q}=1} s_{\text{B}}(p, q) \\ \text{subject to} \quad & (\mathbf{e}, \mathbf{a}) \in \text{O} \wedge (\bar{\mathbf{e}}, \bar{\mathbf{a}}) \in \text{I} \wedge (\mathbf{b}, \mathbf{t}) \in \text{B} \wedge \\ & \mathbf{e} = \bar{\mathbf{e}} \wedge \ \mathbf{a} = \bar{\mathbf{a}} \wedge (\mathbf{e}, \mathbf{a}, \mathbf{t}) \in \text{T}. \end{aligned} \quad \text{(M3)}$$

Again, without the final row, M3 would be separable. We exploit this by performing dual decomposition with a dual objective that has multipliers $\boldsymbol{\lambda}$ for the coupling constraints and multipliers $\boldsymbol{\mu}$ for the constraints which enforce $(\mathbf{e}, \mathbf{a}, \mathbf{t}) \in \text{T}$. The resulting subgradient descent method is also shown in algorithm 2. Notably, since the constraints for T are inequalities, we require a *projected* version of the descent algorithm which enforces $\boldsymbol{\mu} \geq 0$. This manifests itself when $\boldsymbol{\mu}$ is updated using the $[\cdot]_+$ projection.

We have already described how to find the best $\mathbf{e}, \mathbf{a}$ and $\bar{\mathbf{e}}, \bar{\mathbf{a}}$ assignments. What changes for Model 3 is the derivation of the penalties for $\mathbf{e}$ and $\mathbf{a}$ that now come from both $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. We set $\mathbf{c}^{\text{out}}_{i,t}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \lambda_{i,t} + \delta_{t,\texttt{Bind}} \sum_{p,q} \mu^{\text{trig}}_{i,p,q}$. For $j \notin \text{Prot}(\mathbf{x})$ we set $\mathbf{c}^{\text{out}}_{i,j,r}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \lambda_{i,j,r}$; otherwise we use $\mathbf{c}^{\text{out}}_{i,j,r}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \lambda_{i,j,r} + \sum_p \mu^{\text{arg1}}_{i,j,p} + \sum_q \mu^{\text{arg2}}_{i,q,j}$.

For finding a $(\mathbf{b}, \mathbf{t}) \in \text{B}$ that maximizes $\sum_{b_{p,q}=1} s_{\text{B}}(p, q)$ we use bestBind$(\mathbf{c})$, as shown in algorithm 1. It groups together two proteins p, q if their score plus the penalty of the best possible trigger i exceeds 0. In this case, or if there is at least one trigger with positive penalty $c_{i,p,q} > 0$, we activate the set of triggers $I(p, q)$ with maximal score.

Note that when several triggers i maximize the score, we assign them all the same fractional value $|I(p, q)|^{-1}$. This enforces the constraint that at most one binding event can point to both p and q and also means that we are solving an LP relaxation. We could enforce integer solutions and pick arbitrary triggers at a tie, but this would lower the chances of matching against predictions of other routines.

The penalties for bestBind$(\mathbf{c})$ are derived from the dual $\boldsymbol{\mu}$ by setting $\mathbf{c}^{\text{bind}}_{i,p,q}(\boldsymbol{\mu}) = -\mu^{\text{trig}}_{i,p,q} - \mu^{\text{arg1}}_{i,p,q} - \mu^{\text{arg2}}_{i,,p,q}$.

3.4 Training

We choose prediction-based passive-aggressive (PA) online learning (Crammer and Singer, 2003) with averaging to estimate the weights $\mathbf{w}$ for each of our models. PA is an error-driven learner that shifts weights towards features of the gold solution, and away from features of the current guess, whenever the current model makes a mistake.

PA learning takes into account a user-defined loss function for which we use a weighted sum

of false positives and false negatives: $l(\mathbf{y}, \mathbf{y}') \stackrel{\text{def}}{=} \text{FP}(\mathbf{y}, \mathbf{y}') + \alpha \text{FN}(\mathbf{y}, \mathbf{y}')$. We set $\alpha = 3.8$ by optimizing on the BioNLP 2009 development set.

4 Related Work

Riedel et al. (2009) use Integer Linear Programming and cutting planes (Riedel, 2008) for inference in a model similar to Model 2. By using dual decomposition instead, we can exploit tractable substructure and achieve quadratic (Model 2) and cubic (Model 3) runtime guarantees. An advantage of ILP inference are guaranteed certificates of optimality. However, in practice we also gain certificates of optimality for a large fraction of the instances we process. Poon and Vanderwende (2010) use local search and hence provide no such certificates. Their problem formulation also makes n-gram dependency path features harder to incorporate. McClosky et al. (2011b) cast event extraction as dependency parsing task. Their model assumes that event structures are trees, an assumption that is frequently violated in practice. Finally, all previous joint approaches use heuristics to decide whether binding arguments are part of the same event, while we capture these decisions in the joint model.

We follow a long line of research in NLP that addresses search problems using (Integer) Linear Programs (Germann et al., 2001; Roth and Yih, 2004; Riedel and Clarke, 2006). However, instead of using off-the-shelf solvers, we work in the framework of dual decomposition. Here we extend the approach of Rush et al. (2010) in that in addition to equality constraints we dualize more complex coupling constraints between models. This requires us to work with a projected version of subgradient descent.

While tailored towards (biomedical) event extraction, we believe that our models can also be effective in a more general Semantic Role Labeling (SRL) context. Using variants of Model 1, we can enforce many of the SRL constraints—such as "unique agent" constraints (Punyakanok et al., 2004)—without having to call out to ILP optimizers. Meza-Ruiz and Riedel (2009) showed that inducing pressure on arguments to be attached to at least one predicate is helpful; this is a soft incoming edge constraint. Finally, Model 3 can be used to efficiently capture compatibilities between semantic arguments; such compatibilities have also been shown to be helpful in SRL (Toutanova et al., 2005).

5 Experiments

We evaluate our models on several tracks of the 2009 and 2011 BioNLP shared tasks, using the official "Approximate Span Matching/Approximate Recursive Matching" F1 metric for each. We also investigate the runtime behavior of our algorithms.

5.1 Preprocessing

Each document is first processed by the Stanford CoreNLP[2] tokenizer and sentence splitter. Parse trees come from the Charniak-Johnson parser (Charniak and Johnson, 2005) with a self-trained biomedical parsing model (McClosky and Charniak, 2008), and are converted to dependency structures again using Stanford CoreNLP. Based on trigger words collected from the training set, a set of candidate trigger tokens $\text{Trig}(\mathbf{x})$ is generated for each sentence $\mathbf{x}$.

5.2 Features

The feature function $\mathbf{f}_{\text{T}}(i, t)$ extracts a per-trigger feature vector for trigger i and type $t \in \mathcal{T}$. It creates one active feature for each element in $\{t, t \in \mathcal{T}_{\text{Reg}}\} \times \text{feats}(i)$. Here $\text{feats}(i)$ denotes a collection of representations for the token i: wordform, lemma, POS tag, syntactic heads, syntactic children, and membership in two dictionaries taken from Riedel et al. (2009).

For $\mathbf{f}_{\text{R}}(i, j, r)$ we create active features for each element of $\{r\} \times \text{feats}(i, j)$. Here $\text{feats}(i, j)$ is a collection of representations of the token pair (i, j) taken from Miwa et al. (2010c) and contains: labelled and unlabeled n-gram dependency paths; edge and vertex walk features, argument and trigger modifiers and heads, words in between.

For $\mathbf{f}_{\text{B}}(p, q)$ we re-use the token pair representations from $\mathbf{f}_{\text{R}}$. In particular, we create one active feature for each element in $\text{feats}(p, q)$.

5.3 Shared Task 2009

We first evaluate our models on the Bionlp 2009 task 1. The training, development and test sets for this

[2] http://nlp.stanford.edu/software/corenlp.shtml

	SVT	BIND	REG	TOT
McClosky	75.4	**48.4**	40.4	53.5
Poon	77.5	47.9	44.1	55.5
Bjoerne	77.9	42.2	45.5	55.7
Miwa	78.6	46.9	47.7	57.8
M1	77.2	43.0	45.8	56.2
M2	77.9	42.4	47.6	57.2
M3	**78.4**	48.0	**49.1**	**58.7**

Table 1: F1 scores for the development set of Task 1 of the BioNLP 2009 shared task.

	SVT	BIND	REG	TOT
McClosky	68.3	46.9	33.3	48.6
Poon	69.5	42.5	37.5	50.0
Bjoerne	70.2	44.4	40.1	52.0
Miwa	72.1	50.6	45.3	56.3
M1	71.0	42.1	41.9	53.4
M2	70.5	41.3	43.6	53.7
M3	71.1	**52.9**	45.2	55.8
M3+enju	**72.6**	52.6	**46.9**	**57.4**

Table 2: F1 scores for the test set of Task 1 of the BioNLP 2009 shared task.

task consist of 797, 150 and 250 documents, respectively.

Table 1 shows our results for the development set. We compare our three models (M1, M2 and M3) and previous state-of-the-art systems: *McClosky* (McClosky et al., 2011a), *Poon* (Poon and Vanderwende, 2010), *Bjoerne* (Björne et al., 2009) and *Miwa* (Miwa et al., 2010b; Miwa et al., 2010a). Presented is F1 score for all events (TOT), regulation events (REG), binding events (BIND) and simple events (SVT).

Model 1 is outperforming the previous best joint models of Poon and Vanderwende (2010), as well as the best entry of the 2009 task (Björne et al., 2009). This is achieved without careful tuning of thresholds that control flow of information between trigger and argument extraction. Notably, training Model 1 takes approximately 20 minutes using a single core implementation. Contrast this with 20 minutes on 32 cores reported by Poon and Vanderwende (2010).

Model 2 focuses on regulation structures and results demonstrate this: F1 for regulations goes up by nearly 2 points. While the impact of joint modeling relative to weaker local baselines has been shown shown by Poon and Vanderwende (2010) and Riedel et al. (2009), our findings here provide evidence that it remains effective even when the baseline system is very competitive.

With Model 3 our focus is extended to binding events, improving F1 for such events by at least 5 F1. This also has a positive effect on regulation events, as regulations of binding events can now be more accurately extracted. In total we see a 1.1 F1 increase over the best results reported so far (Miwa et al., 2010b). Crucially, this is achieved using only a single parse tree per sentence, as opposed to three used by Miwa et al. (2010a).

Table 2 shows results for the test set. Here with Model 1 we again already outperform all but the results of Miwa et al. (2010a). Model 2 improves F1 for regulations, while Model 3 again increases F1 for both regulations and binding events. This yields the best binding event results reported so far. Notably, not only are we able to resolve binding ambiguity better. Binding attachments themselves also improve, as we increase attachment F1 from 61.4 to 62.7 when going from Model 2 to Model 3.

Miwa et al. (2010b) use two parsers to generate their input features. For fairer comparison we augment Model 3 with syntactic features based on the *enju* parser (Miyao et al., 2009). With these features (M3+enju) we achieve the best results on this dataset reported so far, and outperform Miwa et al. (2010b) by 1.1 F1 in total, 1.6 F1 on regulation events and 2.0 F1 on binding events.

We also apply Model 3, with slight modifications, to the BioNLP 2009 task 2 which requires cellular locations to be extracted as well. With 53.0 F1 we fall 2 points short of the results of Miwa et al. (2010b) but still substantially outperform any other reported results on the dataset. More parse trees may again substantially improve results, as well as task-specific constraint and feature sets.

5.4 Shared Task 2011

We entered the Shared Task 2011 with Model 3, primarily focusing on Genia track (task 1), and the Infectious Diseases track. The Genia track differs from the 2009 task by including both abstracts and full text articles. In total 908 training, 259 development and 347 test documents are provided.

Genia Task 1		Infectious Diseases	
System	TOT	System	TOT
M3+Stanford	**56.0**	M3+Stanford	**55.6**
M3	55.2	M3	53.4
UTurku	53.3	Stanford	50.6
MSR-NLP	51.5	UTurku	44.2
ConcordU	50.3	PNNL	42.6

Table 3: F1 scores for the test sets of two tracks in the BioNLP 2011 Shared Task.

The top five entries are shown in table 3. Model 3 is the best-performing system that does not use model combination, only outperformed by a version of Model 3 that includes Stanford predictions (McClosky et al., 2011b) as input features (Riedel et al., 2011). Not shown in the table are results for full papers only. Here M3 ranks first with 53.1 F1, while M3+Stanford comes in second with 52.7 F1.

The Infectious Diseases (ID) track of the 2011 task has 152 train, 46 development and 118 test documents. Relative to Genia it provides less data and introduces more types of entities as well as the *biological process* event type. Incorporating these changes into our models is straightforward, and hence we omit details for brevity.

Table 3 shows the top five entries for the Infectious Diseases track. Again Model 3 is the best-performing system that does not use model combination, outperformed only by Model 3 with Stanford predictions as features. We should point out that the feature sets and learning parameters were kept constant when moving from Genia to ID data. The strong results we observe without any tuning to the domain indicate the robustness of joint modeling.

5.5 Runtime Behavior

Table 4 shows the asymptotic complexity of our three models with respect to $m = |\text{Trig}(\mathbf{x})|$, $n = |\text{Cand}(\mathbf{x})|$ and $p = |\text{Prot}(\mathbf{x})|$. We also show the number of iterations needed on average, the average time in milliseconds per sentence,[3] and the fraction of sentences we get certificates of optimality for.

As expected, Model 1 is most efficient, both asymptotically and on average. Given that its accuracy is already good, it can serve as a basis for large-scale extraction tasks. Models 2 and 3 require several iterations and more time, while providing slightly less certificates. However, given the improvement in F1 they deliver, and the fact preprocessing steps such as parsing would still dominate the average time, this seems like a reasonable price to pay.

[3]Measured without preprocessing and feature extraction.

	Complexity	Iter.	Time	Exact
M1	$O(nm)$	1.0	60ms	100%
M2	$O(Rnm)$	10.4	183ms	96%
M3	$O\left(Rnm + Rp^2m\right)$	11.7	297ms	94%

Table 4: Complexity and Runtime Behavior.

6 Conclusion

We presented three joint models for biomedical event extraction. Model 1 reaches near-state-of-the-art results, outperforms all previous joint models and has quadratic runtime guarantees. By explicitly capturing regulation events (Model 2), and binding events (Model 3) we achieve the best results reported so far on several event extraction tasks. The runtime penalty we pay is kept minimal by using dual decomposition. We also show how dual decomposition can be used for constraints that go beyond coupling equalities.

We use joint models, a decomposition technique and supervised online learning. This recipe can be successful in many settings, but requires expensive manual annotation. In the future we want to integrate weak supervision techniques to train extractors with existing biomedical databases, such as KEGG, and only minimal amounts of annotated text.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval. The University of Massachusetts gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*, pages 10–18, Morristown, NJ, USA. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 173–180.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL '01)*, pages 228–235.

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 Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, and Jun'ichi Tsujii. 2011. Overview of BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, Portland, Oregon, June. Association for Computational Linguistics.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. Mrf optimization via dual decomposition: Message-passing revisited. In *Proceedings of the 11st IEEE International Conference on Computer Vision (ICCV '07)*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with nonprojective head automata. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '10)*.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL '08)*.

David McClosky, Mihai Surdeanu, and Chris Manning. 2011a. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11)*, Portland, Oregon, June.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011b. Event extraction as dependency parsing in bionlp 2011. In *BioNLP 2011 Shared Task*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the ACL (EACL '06)*, pages 81–88.

Ivan Meza-Ruiz and Sebastian Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '09)*.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010a. A comparative study of syntactic parsers for event extraction. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, BioNLP '10, pages 37–45, Stroudsburg, PA, USA. Association for Computational Linguistics.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010b. Evaluating dependency representation for event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 779–787, Stroudsburg, PA, USA. Association for Computational Linguistics.

Makoto Miwa, Rune Saetre, Jin-Dong D. Kim, and Jun'ichi Tsujii. 2010c. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(1):131–146, February.

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics/computer Applications in The Biosciences*, 25:394–400.

Hoifung Poon and Lucy Vanderwende. 2010. Joint Inference for Knowledge Extraction from Biomedical Literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821, Los Angeles, California, June. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, Wen tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics (COLING '04)*, pages 1346–1352, Morristown, NJ, USA. Association for Computational Linguistics.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '06)*, pages 129–137.

Sebastian Riedel and Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the*

Natural Language Processing in Biomedicine NAACL 2011 Workshop (BioNLP '11), June.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*, pages 41–49.

Sebastian Riedel, David McClosky, Mihai Surdeanu, Christopher D. Manning, and Andrew McCallum. 2011. Model combination for event extraction in BioNLP 2011. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2011 Workshop (BioNLP '11)*, June.

Sebastian Riedel. 2008. Improving the accuracy and efficiency of MAP inference for markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI '08)*, pages 468–475.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL' 04)*, pages 1–8.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '10)*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 589–596, Morristown, NJ, USA. Association for Computational Linguistics.