Joint Inference for Knowledge Extraction from Biomedical Literature

Hoifung Poon[*]
Dept. of Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
hoifung@cs.washington.edu

Lucy Vanderwende
Microsoft Research
Redmond, WA 98052
Lucy.Vanderwende@microsoft.com

Abstract

Knowledge extraction from online repositories such as PubMed holds the promise of dramatically speeding up biomedical research and drug design. After initially focusing on recognizing proteins and binary interactions, the community has recently shifted their attention to the more ambitious task of recognizing complex, nested event structures. State-of-the-art systems use a pipeline architecture in which the candidate events are identified first, and subsequently the arguments. This fails to leverage joint inference among events and arguments for mutual disambiguation. Some joint approaches have been proposed, but they still lag much behind in accuracy. In this paper, we present the first joint approach for bio-event extraction that obtains state-of-the-art results. Our system is based on Markov logic and adopts a novel formulation by jointly predicting events and arguments, as well as individual dependency edges that compose the argument paths. On the BioNLP'09 Shared Task dataset, it reduced F1 errors by more than 10% compared to the previous best joint approach.

1 Introduction

Extracting knowledge from unstructured text has been a long-standing goal of NLP and AI. The advent of the World Wide Web further increases its importance and urgency by making available an astronomical number of online documents containing virtually unlimited amount of knowledge (Craven et al., 1999). A salient example domain is biomedical literature: the PubMed[1] online repository contains over 18 million abstracts on biomedical research, with more than two thousand new abstracts added each day; the abstracts are written in grammatical English, which enables the use of advanced NLP tools such as syntactic and semantic parsers.

Traditionally, research on knowledge extraction from text is primarily pursued in the field of information extraction with a rather confined goal of extracting instances for flat relational schemas with no nested structures (e.g, recognizing protein names and protein-protein interaction (PPI)). This restriction mainly stems from limitations in available resources and algorithms. The BioNLP'09 Shared Task (Kim et al., 2009) is one of the first that faced squarely information needs that are complex and highly structured. It aims to extract nested bio-molecular events from research abstracts, where an event may have variable number of arguments and may contain other events as arguments. Such nested events are ubiquitous in biomedical literature and can effectively represent complex biomedical knowledge and subsequently support reasoning and automated discovery. The task has generated much interest, with twenty-four teams having submitted their results. The top system by UTurku (Bjorne et al., 2009) attained the state-of-the-art F1 of 52.0%.

The nested event structures make this task particularly attractive for applying joint inference. By allowing information to propagate among events and arguments, joint inference can facilitate mutual disambiguation and potentially lead to substantial gain

[*] This research was conducted during the author's internship at Microsoft Research.

[1] http://www.ncbi.nlm.nih.gov/pubmed

in predictive accuracy. However, joint inference is underexplored for this task. Most participants either reduced the task to classification (e.g., by using SVM), or used heuristics to combine manual rules and statistics. The previous best joint approach was Riedel et al. (2009). While competitive, it still lags UTurku by more than 7 points in F1.

In this paper, we present the first joint approach that achieves state-of-the-art results for bio-event extraction. Like Riedel et al. (2009), our system is based on Markov logic, but we adopted a novel formulation that models dependency edges in argument paths and jointly predicts them along with events and arguments. By expanding the scope of joint inference to include individual argument edges, our system can leverage fine-grained correlations to make learning more effective. On the development set, by merely adding a few joint inference formulas to a simple logistic regression model, our system raised F1 from 28% to 54%, already tying UTurku.

We also presented a heuristic method to fix errors in syntactic parsing by leveraging available semantic information from task input, and showed that this in turn led to substantial performance gain in the task. Overall, our final system reduced F1 error by more than 10% compared to Riedel et al. (2009).

We begin by describing the shared task and related work. We then introduce Markov logic and our Markov Logic Network (MLN) for joint bio-event extraction. Finally, we present our experimental results and conclude.

2 Bio-Event Extraction

We follow the BioNLP'09 Shared Task (Kim et al., 2009) on problem setup for bio-event extraction. A bio-molecular event (bio-event) refers to the change of state for bio-molecules such as DNAs and proteins. The goal is to extract these events from unstructured text such as biomedical abstracts. For each event, one needs to identify the trigger words that signifies the event and the theme arguments that undergo the change. In addition, for regulation events, the cause argument also needs to be identified if it is present. The task considers nine event types: `Expression`, `Transcription`, `Localization`, `Phosphorylation`, `Catabolism`, `Binding`, `Regulation`, `Positive_regulation`, and `Negative_regulation`. Only `Binding` can take multiple themes. Regulation events may take events as arguments. To facilitate evaluation, the task fixes the type of non-event arguments to protein and provides ground truth of protein mentions as input. [2]

Like any NLP task, ambiguity is a central problem. The same event can be expressed in many variations. For example, a `Negative_regulation` event may be signified by "inhibition", "down-regulation", "is abrogated by", to name a few. On the other hand, depending on the context, the same expression may represent different events. For example, "level" may signify any one of five event types in the training set, or signify none.

In addition, the nested event structures present new challenges to knowledge extraction systems. To recognize a complex event, besides from identifying the event type and trigger words, one also needs to identify its arguments and recursively identify their event structures. A mistake in any part will render a failure in this extraction.

The interdependencies among events and arguments naturally argue for joint predictions. For example, given the snippet "the level of VCAM-1 mRNA", knowing that "level" might signify an event helps to recognize the prepositional phrase (PP) as its theme. Conversely, the presence of the PP suggests that "level" is likely an event. Moreover, the word "mRNA" in the PP indicates that the event type is probably `Transcription`.

Most existing systems adopt a pipeline architecture and reduce the task to independent classifications of events and arguments. For example, the best system UTurku (Bjorne et al., 2009) first extracts a list of candidate triggers with types, and then determines for each pair of candidate triggers or proteins whether one is a theme or cause of the other. The triggers missed in the first stage can never be recovered in the second one. Moreover, since the second stage is trained with gold triggers as input, any trigger identified in the first stage tends to get at least

[2]The Shared Task also defines two other tasks (Tasks 2 and 3), which aim either to extract additional arguments (e.g., sites), or to determine if an event is a negation or speculation. In this paper, we focus on the core task (Task 1) as it is what most systems participate in, but our approach can be extended straightforwardly to handle the other tasks.

one argument, even though it may not be an event at all. As a result, the authors had to use an ad hoc procedure to trade off precision and recall for the final prediction task while training the first-stage extractor. In addition, each trigger or argument is classified independently using a multi-class SVM.

While joint inference can potentially improve accuracy, in practice, it is often very challenging to make it work (Poon and Domingos, 2007). The previous best joint approach for this task was proposed by Riedel et al. (2009) (labeled UT+DBLS in Kim et al. (2009)). Their system is also based on Markov logic (Domingos and Lowd, 2009). While competitive (ranked fourth in the evaluation), their system still lags UTurku by more than 7 points in F1.

Most systems, Riedel et al.'s included, classify each candidate argument path as a whole. A notable exception is the UTokyo system (Saetre et al., 2009), which incorporated sequential modeling by adapting a state-of-the-art PPI system based on MEMM. But they considered adjacent words in the sentence, which offered little help in this task, and their system trailed UTurku by 15 points in F1.

All top systems for event extraction relied heavily on syntactic features. We went one step further by formulating joint predictions directly on dependency edges. While this leverages sequential correlation along argument paths, it also makes our system more prone to the adverse effect of syntactic errors. Joint syntactic and semantic processing has received much attention lately (Hajic et al., 2009). In this paper, we explore using a heuristic method to correct syntactic errors based on semantic information, and show that it leads to significant performance gain for event extraction.

3 Markov Logic

In many NLP applications, there exist rich relation structures among objects, and recent work in statistical relational learning (Getoor and Taskar, 2007) and structured prediction (Bakir et al., 2007) has shown that leveraging these can greatly improve accuracy. One of the leading frameworks for joint inference is Markov logic, a probabilistic extension of first-order logic (Domingos and Lowd, 2009). A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that generated it. The probability of a state x in such a network is given by $P(x) = (1/Z) \exp\left(\sum_i w_i f_i(x)\right)$, where Z is a normalization constant, w_i is the weight of the ith clause, $f_i = 1$ if the ith clause is true, and $f_i = 0$ otherwise.

Markov logic makes it possible to compactly specify probability distributions over complex relational domains. Efficient inference can be performed using MC-SAT (Poon and Domingos, 2006). MC-SAT is a "slice sampling" Markov chain Monte Carlo algorithm that uses an efficient satisfiability solver to propose the next sample. It is orders of magnitude faster than previous MCMC algorithms like Gibbs sampling, making efficient sampling possible on a scale that was previously out of reach.

Supervised learning for Markov logic maximizes the conditional log-likelihood of query predicates given the evidence in the train data. This learning objective is convex and can be optimized using gradient descent, where the gradient is estimated using MC-SAT.

In practice, it is often difficult to tune the learning rate, especially when the number of groundings varies widely among clauses (known as *ill-conditioning* in numerical optimization). This problem is particularly severe in relational domains. One remedy is to apply preconditioning to the gradient. For example, Poon & Domingos (2007) divided the global learning rate by the number of true groundings of the corresponding clause in the training data, whereas Lowd & Domingos (2007) divided it by the variance of the clause (also estimated using MC-SAT). The latter can be viewed as approximating the Hessian with its diagonal, and is guaranteed optimal when the weights are not correlated (e.g., in logistic regression). Lowd & Domingos (2007) also used a scaled conjugate gradient algorithm to incorporate second-order information and further adapt the search direction.

The open-source Alchemy package (Kok et al., 2009) provides implementations of existing algorithms for Markov logic.

4 An MLN for Joint Bio-Event Extraction

In this section, we present our MLN for joint bio-event extraction. As standard for this task, we assume that Stanford dependency parses are available in the input. Our MLN jointly makes the following predictions: for each token, whether it is a trigger word (and if so, what is the event type), and for each dependency edge, whether it is in an argument path leading to a theme or cause.

To the best of our knowledge, the latter part makes this formulation a novel one. By breaking the prediction of an argument path into that on individual dependency edges, it can leverage the correlation among adjacent edges and make learning more effective. Indeed, compared to other top systems, our MLN uses a much simpler set of features, but is still capable of obtaining state-of-the-art results.[3] Computationally, this formulation is also attractive. The number of predictions is bounded by the number of tokens and edges, and is linear in sentence length, rather than quadratic.

Our MLN also handles the regulation events differently. We notice that events of the three regulation types often occur in similar contexts, and sometimes share trigger words (e.g., "involve"). Therefore, our MLN merges them into a single event type `Regulation`, and additionally predicts the regulation direction (`Positive` or `Negative`). This allows it to pool information shared by the three types.

Base MLN: The following are the main query predicates we used, along with descriptions:

$\mathtt{Event(i)}$**:** token $\mathtt{i}$ signifies an event;

$\mathtt{EvtType(i,e)}$**:** $\mathtt{i}$ is of event type $\mathtt{e}$;

$\mathtt{RegType(i,r)}$**:** $\mathtt{i}$ is of regulation type $\mathtt{r}$;

$\mathtt{InArgPath(i,j,a)}$**:** the dependency edge from $\mathtt{i}$ to $\mathtt{j}$ is in an argument path of type $\mathtt{a}$, with $\mathtt{a}$ being either `Theme` or `Cause`.

If event $\mathtt{i}$ has type `Positive_regulation`, both $\mathtt{EvtType(i,Regulation)}$ and $\mathtt{RegType(i,Positive)}$ are true. Similarly for `Negative_regulation`. If the type is `Regulation`, only $\mathtt{EvtType(i,Regulation)}$ is true.

[3] In future work, we plan to incorporate a much richer set of features; Markov logic makes such extensions straightforward.

Table 1: Formulas in the base MLN.

$\mathtt{Token(i,+t)} \Rightarrow \mathtt{EvtType(i,+e)}$
$\mathtt{Token(i,+t)} \Rightarrow \mathtt{RegType(i,+r)}$
$\mathtt{Token(j,+t)} \wedge \mathtt{Dep(i,j,d)} \Rightarrow \mathtt{EvtType(i,+e)}$
$\mathtt{Dep(i,j,+d)} \Rightarrow \mathtt{InArgPath(i,j,+a)}$
$\mathtt{Dep(i,j,+d)} \wedge \mathtt{Prot(i)} \Rightarrow \mathtt{InArgPath(i,j,+a)}$
$\mathtt{Dep(i,j,+d)} \wedge \mathtt{Prot(j)} \Rightarrow \mathtt{InArgPath(i,j,+a)}$
$\mathtt{Token(i,+t)} \wedge \mathtt{Dep(i,j,+d)} \Rightarrow \mathtt{InArgPath(i,j,+a)}$
$\mathtt{Token(j,+t)} \wedge \mathtt{Dep(i,j,+d)} \Rightarrow \mathtt{InArgPath(i,j,+a)}$

The main evidence predicates are:

$\mathtt{Token(i,w)}$**:** token $\mathtt{i}$ has word $\mathtt{w}$;

$\mathtt{Dep(i,j,d)}$**:** there is a dependency edge from $\mathtt{i}$ to $\mathtt{j}$ with label $\mathtt{d}$; [4]

$\mathtt{Prot(i)}$**:** $\mathtt{i}$ is a protein.

Our base MLN is a logistic regression model, and can be succintly captured by eight formulas in Table 1. All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the formula, with a separate weight, for each value combination of the variables with a plus sign. The first three formulas predict the event type and regulation direction based on the token word or its neighbor in the dependency tree. The next five formulas predict whether a dependency edge is in an argument path, based on some combinations of token word, dependency label, and whether the nodes are proteins.

By default, we also added the unit formulas: $\mathtt{Theme(x,y)}$, $\mathtt{Cause(x,y)}$, $\mathtt{EventType(x,+e)}$, $\mathtt{RegType(x,+r)}$, which capture default regularities.

Joint Inference: Like any classification system, the formulas in the base MLN make independent predictions at inference time. This is suboptimal, because query atoms are interdependent due to either hard constraints (e.g., an event must have a type) or soft correlation (e.g., "increase" signifies an event and the `dobj` edge from it leads to a theme). We thus

[4] For convenience, we include the reverse dependency edges in the evidence. For example, if $\mathtt{Dep(i,j,nn)}$ is true, then so is $\mathtt{Dep(j,i,-nn)}$.

augment the base MLN with two groups of joint-inference formulas. First we incorporate the following hard constraints.

$$\texttt{Event(i)} \Rightarrow \exists \texttt{t}.\ \texttt{EvtType(i,t)}$$

$$\texttt{EvtType(i,t)} \Rightarrow \texttt{Event(i)}$$

$$\texttt{RegType(i,r)} \Rightarrow \texttt{EvtType(i,Regulation)}$$

$$\begin{gathered}\texttt{InArgPath(i,j,Theme)} \Rightarrow \texttt{Event(i)} \\ \vee\ \exists\, \texttt{k} \neq \texttt{j}.\ \texttt{InArgPath(k,i,Theme)}\end{gathered}$$

$$\begin{gathered}\texttt{InArgPath(i,j,Cause)} \\ \Rightarrow \texttt{EvtType(i,Regulation)} \\ \vee\ \exists\, \texttt{k} \neq \texttt{j}.\ \texttt{InArgPath(k,i,Cause)}\end{gathered}$$

$$\begin{gathered}\texttt{InArgPath(i,j,Theme)} \Rightarrow \texttt{Prot(j)} \\ \vee\ \exists\, \texttt{k} \neq \texttt{i}.\ \texttt{InArgPath(j,k,Theme)}\end{gathered}$$

$$\begin{gathered}\texttt{InArgPath(i,j,Cause)} \Rightarrow \texttt{Event(j)} \vee \texttt{Prot(j)} \\ \vee\ \exists\, \texttt{k} \neq \texttt{i}.\ \texttt{InArgPath(j,k,Cause)}\end{gathered}$$

The first three formulas enforce that events must have a type, that a token assigned an event (regulation) type must be an (regulation) event. The next four formulas enforce the consistency of argument path assignments: an argument path must start with an event, in particular, a cause path must start with a regulation event; a theme path must eventually trace to a protein, whereas a cause path may also stop at an event (which does not have a cause itself). To avoid looping, we forbid reverse edges in a path.[5]

Notice that with these constraints, adjacent edges in the dependency tree correlate with each other in their InArgPath assignments, much like in an HMM for linear sequences. Moreover, these assignments correlate with the event and event-type ones; knowing that i probably signifies an event makes it easier to detect an argument path, and vice versa. In addition, events that share partial argument paths can inform each other through the predictions on edges. In the experiments section, we will see that merely adding these hard constraints leads to 26-point gain in F1.

We also notice that different trigger words may use different dependencies to start an argument path of a particular type. For example, for many verbs, nsubj tends to start a cause path and dobj a theme path. However, for "bind" that signifies a Binding event, both lead to themes, as in "A binds B". Such soft regularities can be captured by a single joint formula: $\texttt{Token(i,+w)} \wedge \texttt{Dep(i,j,+d)} \Rightarrow \texttt{Event(i)} \wedge \texttt{InArgPath(i,j,+a)}$, which correlates event and argument type with token and dependency.

Linguistically-Motivated Formulas: Natural languages often possess systematic syntactic alternations. For example, for the word "increase", if both subject and object are present, as in "A increases the level of B", the subject is the cause whereas the object is the theme. However, if only subject is present, as in "The level of B increases", the subject is the theme. We thus augment the MLN with a number of context-specific formulas such as: $\texttt{Token(i,increase)} \wedge \texttt{Dep(i,j,nsubj)} \wedge \texttt{Dep(i,k,dobj)} \wedge \texttt{Event(i)} \wedge \texttt{Cause(i,j)}$.[6]

5 Learning And Inference

When training data comprises of many independent subsets (e.g., individual abstracts), stochastic gradient descent (SGD) is often a favorable method for parameter learning. By adopting small and frequent updates, it can dramatically speed up learning and sometimes even improve accuracy. Moreover, it easily scales to large datasets since each time it only needs to bring a few subsets into the memory.

In this paper, we used SGD to learn weights for our MLN. During this process, we discovered some general challenges for applying SGD to relational domains. For example, the ill-conditioning problem is particularly severe, and using a single learning rate either makes learning extremely slow or leads to divergence. Like Lowd & Domingos (2007), we combat this by dividing the learning rate by the variance. However, this still leads to divergence as learning progresses. The reason is that some weights are strongly correlated due to the joint formulas, especially the hard constraints. Therefore, the diagonal approximates the Hessian poorly. Inspired by Poon & Domingos (2007), for each formula, we count the numbers of true and false groundings in the train data, and add the smaller of the two plus one to the variance, before dividing the global rate by it.

[5]This is violated in some cases, and can be relaxed. We enforced it for simplicity in this paper.

[6]Available at http://research.microsoft.com/en-us/people/lucyv/naacl10.

We found that this is effective for making learning stable in our experiments.

To compute the most probable state, we used MC-SAT to estimate the marginal probability of each query atom, and returned the ones with probability above a threshold. This allows us to easily trade off precision and recall by varying the threshold. To speed up burn-in, we followed Poon et al. (2009) and first ran MC-SAT with deterministic annealing for initialization.

6 Correcting Syntactic Errors With Semantic Information

Two typical types of syntactic errors are PP-attachment and coordination. For semantic tasks such as bio-event extraction, these errors also have the most adverse impact to performance. For example, for the snippet "involvement of p70 activation in IL-10 up-regulation by gp41", the Stanford parser makes two errors by attaching "up-regulation" to "activation" instead of "involvement", and attaching "gp41" to "involvement" instead of "up-regulation". This makes it very difficult to predict that "gp41" is the cause of "up-regulation", and that "up-regulation" is the theme of "involvement". For conjucts such as "IL-2 and IL-4 expressions", the parser will align "IL-2" with "expressions", which makes it difficult to recognize the expression event on "IL-2". For nested events like "gp41 regulates IL-2 and IL-4 expressions", this results in three extraction errors: IL-2 expression and the regulation event on it are missing, whereas an erroneous regulation event on IL-2 is predicted.

Syntactic errors are often incurred due to lack of semantic information during parsing (e.g., the knowledge that IL-2 and IL-4 are both proteins). In this paper, we used a heuristic method to fix such errors by incorporating two sources of semantic information: argument paths in training data and input protein labels. For conjuncts (signified by prefix `conj` in Stanford dependencies) between a protein and a non-protein, we check whether the non-protein has a protein child, if so, we remove the conjunct and reattach the first protein to the non-protein. For PP-attachments, we notice that often the errors can be fixed by reattaching the child to the closest node that fits a known attachment pattern (e.g., "up-regulation by PROTEIN"). We used the following heuristics to gather attachment patterns. For each argument path in the training data, if it consists of a single PP edge, then we add the combination of governor, dependency label, and dependent to the pattern. (Protein names are replaced with a special string.) If a path contains multiple edges, but a PP edge attaches to a word to the left of the event trigger (e.g., "gp41" attached to "involvement"), our system concludes that the dependent should instead be attached to the trigger and adds the corresponding pattern. In addition, we added a few default patterns like "involvement in" and "effect on". For each PP edge, the candidates for reattachment include the current governor, and the governor's parent and all rightmost descendants (i.e., its rightmost child, the rightmost child of that child, etc.) that are to the left of the dependent. We reattach the dependent to the closest candidate that fits an attachment pattern. If there is none, the attachment remains unchanged. In total, the fraction of reattachments is about 4%.

7 Experiments

We evaluated our system on the dataset for Task 1 in the BioNLP'09 Shared Task (Kim et al., 2009). It consists of 800 abstracts for training, 150 for development and 260 for test. We conducted feature development and tuned hyperparameters using the development set, and evaluated our final system on test using the online tool provided by the organizers. (The test annotations are not released to the public.) All results reported were obtained using the main evaluation criteria for the shared task.[7]

7.1 System

Our system first carries out lemmatization and breaks up hyphenated words.[8] It then uses the Stanford parser (de Marneffe et al., 2006) to generate dependencies. For simplicity, if an event contains multiple trigger words, only the head word is labeled.[9]

[7]Namely, "Approximate Span/Approximate Recursive Matching". See Kim et al. (2009) for details.

[8]E.g., "gp41-induced" becomes "gp41" and "induced", with a new dependency edge labeled `hyphen` from "induced" to "gp41". To avoid breaking up protein names with hyphens, we only dehyphenate words with suffix in a small hand-picked list.

[9]Most events have only one trigger, and the chosen words only need to lie within an approximate span in evaluation.

Table 2: Comparison of our full system with its variants and with UTurku on the development set.

	Rec.	Prc.	F1
BASE	17.4	67.2	27.7
BASE+HARD	49.4	58.5	53.6
FULL	51.5	60.0	**55.5**
−LING	50.5	59.6	54.7
−SYN-FIX	48.2	54.6	51.2
UTurku	51.5	55.6	53.5

We implemented our system as an extension to the Alchemy system (Kok et al., 2009). In particular, we developed an efficient parallelized implementation of our stochastic gradient descent algorithm using the message-passing interface (MPI). For learning, we used a mini-batch of 20 abstracts and iterated through the training files twice. For each mini-batch, we estimated the gradient by running MC-SAT for 300 samples; the initialization was done by running annealed MC-SAT for 200 samples, with temperature dropping from 10 to 0.1 at 0.05 decrements. For inference, we initialized MC-SAT with 1000 annealed samples, with temperature dropping from 10 to 0.1 at 0.01 decrements, we then ran MC-SAT for 5000 samples to compute the marginal probabilities. This implementation is very efficient: learning took about 20 minutes in a 32-core cluster with 800 training files; inference took a few minutes in average.

To obtain the final assignment, we set the query atoms with probability no less than 0.4 to true and the rest to false. The threshold is chosen to maximize F1 in the development set. To generate the events, we first found arguments for each trigger `i` by gathering all proteins and event triggers that were accessible from `i` along an argument path without first encountering another trigger. For triggers of base event types, we dropped other triggers from its argument list. For nested triggers, we generated events recursively by first processing argument triggers and generating their events, and then generating events for the parent trigger by including all combinations of argument events. For `Binding` triggers, we group its arguments by the first dependency labels in the argument paths, and generate events by a cross-product of the group members.

Table 3: Per-type recall/precision/F1 for our full system on the development set.

	Rec.	Prc.	F1
`Expression`	75.6	79.1	77.3
`Transcription`	69.5	73.1	71.3
`Phosphorylation`	87.2	87.2	87.2
`Catabolism`	85.7	100	92.3
`Localization`	66.0	85.4	74.5
`Binding`	39.1	61.8	47.9
`Positive_regulation`	41.8	51.0	46.0
`Negative_regulation`	39.3	56.2	46.3
`Regulation`	41.4	33.2	36.8

7.2 Results

We first conducted experiments on the development set to evaluate the contributions of individual components. Table 2 compares their performances along with that of UTurku. The base MLN (**BASE**) alone performed rather poorly. Surprisingly, by just adding the hard constraints to leverage joint inference (**BASE+HARD**), our system almost doubled the F1, and tied UTurku. In addition, adding the soft joint-inference formula results in further gain, and our full system (**FULL**) attained an F1 of 55.5. This is two points higher than UTurku and the best reported result on this dataset. The linguistically-motivated formulas are beneficial, as can seen by comparing with the system without them (−**LING**), although the difference is small. Fixing the syntactic errors with semantic information, on the other hand, leads to substantial performance gain. Without doing it (−**SYN-FIX**), our system suffers an F1 loss of more than four points. This verifies that the quality of syntactic analysis is important for event extraction. The differences between FULL and other variants (except -LING) are all statistically significant at 1% level using McNemar's test.

To understand the performance bottlenecks, we show the per-type results in Table 3 and the results at the predicate level in Table 4.[10] Both trigger and argument-edge detections leave much room for improvement. In particular, the system proposed many incorrect regulation triggers, partly because regulation triggers have the most variations

[10]Numbers in Table 3 refer to events, whereas in Table 4 to triggers. A trigger may signify multiple events, so numbers in Table 4 can be smaller than that in Table 3.

Table 4: Predicate recall/precision/F1 for our full system on the development set.

	Rec.	Prc.	F1
`Expression`	80.1	82.0	81.0
`Transcription`	68.8	71.0	69.8
`Phosphorylation`	87.5	92.1	89.7
`Catabolism`	84.2	100	91.4
`Localization`	62.5	86.2	72.5
`Binding`	62.4	82.4	71.1
`Positive_regulation`	65.8	70.7	68.2
`Negative_regulation`	58.3	71.7	64.3
`Regulation`	61.7	43.4	50.9
All triggers	68.1	71.7	69.9
Argument edge	69.0	71.8	70.4

Table 5: Comparison of our full system with top systems on the test set.

	Rec.	Prc.	F1
UTurku	46.7	58.5	52.0
JULIELab	45.8	47.5	46.7
ConcordU	35.0	61.6	44.6
Riedel et al.	36.9	55.6	44.4
FULL MLN	43.7	58.6	50.0

among all types. Our system did well in recognizing `Binding` triggers, but performed much poorer at the event level. This indicates that the bottleneck lies in correctly identifying all arguments for multi-theme events. Indeed, if we evaluate on individual event-theme pairs for `Binding`, the F1 jumps 15 points to 62.8%, with precision 82.7% and recall 50.6%.

Finally, Table 5 compares our system with the top systems on the test set. Our system trailed UTurku due to a somewhat lower recall, but substantially outperformed all other systems. In particular, it reduced F1 error by more than 10% compared to the previous best joint approach by Riedel et al. (2009).

7.3 Error Analysis

Through manual inspection, we found that many remaining errors were related to syntactic parses. The problem is particularly severe when there are nested or co-occuring PP-attachments and conjuncts (e.g., "increased levels of IL-2 and IL-4 mRNA and protein in the cell"). Our rule-based procedure in Section 6 has high precision in fixing some of these errors, but the coverage is limited. It also makes hard decisions in a preprocessing step, which cannot be reverted. A principled solution is to resolve syntactic and semantic ambiguities in a joint model that integrates reattachment decisions and extractions. This can potentially resolve more syntactic errors, as extraction makes more semantic information available, and is more robust to reattachment uncertainty.

In some challenging cases, we found further opportunities for joint inference. For example, in the sentence "These cells are deficient in FasL expression, although their cytokine IL-2 production is normal", "normal" signifies a `Positive_regulation` event over "IL-2 production" because of its contrast with "deficient". Such events can be detected by introducing additional joint inference rules that leverage syntactic structures such as subclauses.

We also found many cases where the annotations differ for the same expressions. For example, "cotransfection with PROTEIN" is sometimes labeled as both an `Expression` event and a `Positive_regulation` event, and sometimes not labeled at all. This occurs more often for regulation events, which partly explains the low precision for them.

8 Conclusion

This paper presents the first joint approach for bio-event extraction that achieves state-of-the-art results. This is made possible by adopting a novel formulation that jointly predicts events, arguments, as well as individual dependency edges in argument paths. Our system is based on Markov logic and can be easily extended to incorporate additional knowledge and linguistic features to further improve accuracy.

Directions for future work include: leveraging additional joint-inference opportunities, better integration of syntactic parsing and event extraction, and applying this approach to other extraction tasks and domains.

9 Acknowledgements

We give warm thanks to Sebastian Riedel and three anonymous reviewers for helpful comments and suggestions.

References

G. Bakir, T. Hofmann, B. B. Schölkopf, A. Smola, B. Taskar, S. Vishwanathan, and (eds.). 2007. *Predicting Structured Data*. MIT Press, Cambridge, MA.

Jari Bjorne, 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 BioNLP Workshop 2009*.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. 1999. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. ELRA.

Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Jan Hajic, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antonia Martii, Lluis Marquez, Adam Meyers, Joakim Nivre, Sebastian Pado, Jan Stepanek, Pavel Stranak, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Junichi Tsujii. 2009. Overview of BioNLP-09 Shared Task on event extraction. In *Proceedings of the BioNLP Workshop 2009*.

Stanley Kok, Parag Singla, Matt Richardson, Pedro Domingos, Marc Sumner, Hoifung Poon, and Daniel Lowd. 2009. The alchemy system for statistical relational ai. Technical report, Dept. of CSE, Univ. of Washington, http://alchemy.cs.washington.edu/.

Daniel Lowd and Pedro Domingos. 2007. Efficient weight learning for markov logic networks. In *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211, Warsaw. Springer.

Hoifung Poon and Pedro Domingos. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, pages 458–463, Boston, MA. AAAI Press.

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the Twenty Second National Conference on Artificial Intelligence*, pages 913–918, Vancouver, Canada. AAAI Press.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of NAACL-HLT*, Boulder, Colorado. ACL.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Junichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP Workshop 2009*.

Rune Saetre, Makoto Miwa, Kazuhiro Yoshida, and Junichi Tsujii. 2009. From protein-protein interaction to molecular event extraction. In *Proceedings of the BioNLP Workshop 2009*.