A Markov Logic Approach to Bio-Molecular Event Extraction

Sebastian Riedel[*†] **Hong-Woo Chun**[*†] **Toshihisa Takagi**[*¶] **Jun'ichi Tsujii**[†‡§]
[*]Database Center for Life Science, Research Organization of Information and System, Japan
[†]Department of Computer Science, University of Tokyo, Japan
[¶]Department of Computational Biology, University of Tokyo, Japan
[‡]School of Informatics, University of Manchester, UK
[§]National Centre for Text Mining, UK
{sebastian,chun,takagi}@dbcls.rois.ac.jp
tsujii@is.s.u-tokyo.ac.jp

Abstract

In this paper we describe our entry to the BioNLP 2009 Shared Task regarding biomolecular event extraction. Our work can be described by three design decisions: (1) instead of building a pipeline using local classifier technology, we design and learn a joint probabilistic model over events in a sentence; (2) instead of developing specific inference and learning algorithms for our joint model, we apply Markov Logic, a general purpose Statistical Relation Learning language, for this task; (3) we represent events as relational structures over the tokens of a sentence, as opposed to structures that explicitly mention abstract event entities. Our results are competitive: we achieve the 4th best scores for task 1 (in close range to the 3rd place) and the best results for task 2 with a 13 percent point margin.

1 Introduction

The continuing rapid development of the Internet makes it very easy to quickly access large amounts of data online. However, it is impossible for a single human to read and comprehend a significant fraction of the available information. Genomics is not an exception, with databases such as MEDLINE storing a vast amount of biomedical knowledge.

A possible way to overcome this is information extraction (IE) based on natural language processing (NLP) techniques. One specific IE sub-task concerns the extraction of molecular events that are mentioned in biomedical literature. In order to drive forward research in this domain, the BioNLP Shared task 2009 (Kim et al., 2009) concerned the extraction of such events from text. In the course of the shared task the organizers provided a training/development set of abstracts for biomedical papers, annotated with the mentioned events. Participants were required to use this data in order to engineer a event predictor which was then evaluated on unseen test data.

The shared task covered three sub-tasks. The first task concerned the extraction of events along with their clue words and their main arguments. Figure 1 shows a typical example. The second task was an extension of the first one, requiring participants to not only predict the core arguments of each event, but also the cellular locations the event is associated with in the text. The events in this task were similar in nature to those in figure 1, but would also contain arguments that are neither events nor proteins but cellular location terms. In contrast to the protein terms, cellular location terms were not given as input and had to be predicted, too. Finally, for task 3 participants were asked to extract negations and speculations regarding events. However, in our work we only tackled Task 1 and Task 2, and hence we omit further details on Task 3 for brevity.

Our approach to biomedical event extraction is inspired by recent work on Semantic Role Labelling (Meza-Ruiz and Riedel, 2009; Riedel and Meza-Ruiz, 2008) and can be characterized by three decisions that we will illustrate in the following. First, we do not build a pipelined system that first predicts event clues and cellular locations, and then relations between these; in-

Proceedings of the Workshop on BioNLP: Shared Task, pages 41–49,
Boulder, Colorado, June 2009.

stead, we design and learn a **joint** discriminative model of the complete event structure for a given sentence. This allows us to incorporate global correlations between decisions in a principled fashion. For example, we know that any event that has arguments which itself are events (such as the positive regulation event in figure 1) has to be a regulation event. This means that when we make the decision about the type of an event (e.g., in the first step of a classification pipeline) *independently* from the decisions about its arguments and their type, we run the risk of violating this constraint. However, in a joint model this can be easily avoided.

Our second design choice is the following: instead of designing and implementing specific inference and training methods for our structured model, we use **Markov Logic**, a Statistical Relational Learning language, and define our global model declaratively. This simplified the implementation of our system significantly, and allowed us to construct a very competitive event extractor in three person-months. For example, the above observation is captured by the simple formula:

$$eventType\,(e,t) \wedge role\,(e,a,r) \wedge event\,(a) \Rightarrow regType\,(t) \quad (1)$$

Finally, we represent event structures as **relational structures over tokens** of a sentence, as opposed to structures that explicitly mention abstract event entities (compare figure 1 and 2). The reason is as follows. Markov Logic, for now, is tailored to *link prediction* problems where we may make inferences about the existence of relations between given entities. However, when the identity and number of objects of our domain is unknown, things become more complicated. By mapping to relational structure over grounded text, we also show a direct connection to recent formulations of Semantic Role Labelling which may be helpful in the future.

The remainder of this paper is organized as follows: we will first present the preprocessing steps we perform (section 2), then the conversion to a link prediction problem (section 3). Subsequently, we will describe Markov Logic (section 4) and our Markov Logic Network for event extraction (section 5). Finally, we present our results (in section 6) and conclude (section 7).

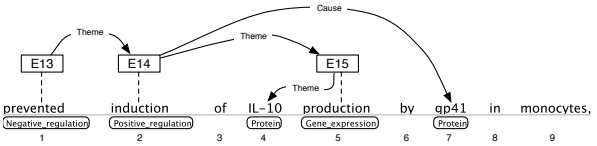

Figure 1: Example gold annotation for task 1 of the shared task.

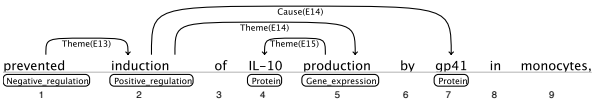

Figure 2: Link Prediction version of the events in figure 1.

2 Preprocessing

The original data format provided by the shared task organizers consists of (a) a collection biomedical abstracts, and (b) standoff annotation that describes the proteins, events and sites mentioned in these abstracts. The organizers also provided a set of dependency and constituent parses for the abstracts. Note that these parses are based on a different tokenisation of the text in the abstracts.

In our first preprocessing step we convert the standoff annotation in the original data to standoff annotation for the tokenisation used in the parses. This allows us to formulate our probabilistic model in terms of one consistent tokenisation (and be able to speak of token instead of character offsets). Then we we retokenise the input text (for the parses) according the protein boundaries that were given in the shared task data (in order to split strings such as "p50/p55"). Finally, we use this tokenisation to once again adapt the stand-off annotation (using the previously adapted version as input).

3 Link Prediction Representation

As we have mentioned earlier, before we learn and apply our Statistical Relational Model, we convert the task to link prediction over a sequence of tokens. In the following we will present this transformation in detail.

To simplify our later presentation we will first introduce a formal representation of the events, proteins and locations mentioned in a sentence. Let us simply identify both proteins and cellular location entities with their token position in the sentence. Furthermore, let us describe an event e as a tuple (i, t, A) where i is the token position of the clue word of e and t is the event type of e; A is a set of labelled arguments (a, r) where each a is either a protein, location or event, and r is the role a plays with respect to e. We will identify the set of all proteins, locations and events for a sentence with P, L and E, respectively.

For example, in figure 1 we have $P = \{4, 7\}, L = \emptyset$ and $E = \{e_{13}, e_{14}, e_{15}\}$ with

$$\begin{aligned} e_{15} &= (5, \text{gene\_expr}, \{(4, \text{Theme})\}) \\ e_{14} &= (2, \text{pos\_reg}, \{(e_{15}, \text{Theme}), (7, \text{Cause})\}) \\ e_{13} &= (1, \text{neg\_reg}, \{(e_{14}, \text{Theme})\}) \end{aligned}$$

3.1 Events to Links

As we mentioned in section 1, Markov Logic (or its interpreters) are not yet able to deal with cases where the number and identity of entities is unknown, while relations/links between known objects can be readily modelled. In the following we will therefore present a mapping of an event structure E to a labelled relation over tokens. Essentially, we project E to a pair (L, C) where L is a set of labelled token-to-token links (i, j, r), and C is a set of labelled event clues (i, t). Note that this mapping has another benefit: it creates a "predicate-argument" structure very similar to most recent formulations of Semantic Role Labelling (Surdeanu et al., 2008). Hence it may be possible to re-use or adapt the successful approaches in SRL in order to improve bio-molecular event extraction. Since our approach is inspired by the Markov Logic role labeller in (Riedel and Meza-Ruiz, 2008), this work can be seen as an attempt in this direction.

For a sentence with given P, L and E, algorithm 1 presents our mapping from E to (L, C). For brevity we omit a more detailed description of the algorithm. Note that for our running example *eventsToLinks* would return

$$C = \{(1, \text{neg\_reg}), (2, \text{pos\_reg}), (5, \text{gene\_expr})\} \quad (2)$$

Algorithm 1 Event to link conversion

```
/* returns all clues C and links L given
   by the events in E */
function eventsToLinks(E):
C ← ∅, L ← ∅
for each event (i,t,A) ∈ E do
  C ← C∪{(i,t)}
  for each argument (a,r) ∈ A do
    if a is an event (i',t',A') do
      L ← L∪{(i,i',r)} with a = (i',t',A')
    else
      L ← L ∪ {(i,a,r)}
return (C,L)
```

and

$$\begin{aligned} L = \; & \{(1, 2, \text{Theme}), (2, 5, \text{Theme}), \\ & (2, 7, \text{Cause}), (5, 4, \text{Theme})\}. \end{aligned} \quad (3)$$

3.2 Links to Events

The link-based representation allows us to simplify the design of our Markov Logic Network. However, after we applied the MLN to our data, we still need to transform this representation back to an event structure (in order to use or evaluate it). This mapping is presented in algorithm 2 and discussed in the following. Note that we expect the relational structure L to be cycle free. We again omit a detailed discussion of this algorithm. However, one thing to notice is the special treatment we give to binding events. Roughly speaking, for the binding event clue c we create an event with all arguments of c in L. For a non-binding event clue c we first collect all roles for c, and then create one event per assignment of argument tokens to these roles.

If we would re-convert C and L from equation 2 and 3, respectively, we could return to our original event structure in figure 1. However, converting back and forth is not loss-free in general. For example, if we have a non-binding event in the original E set with two arguments A and B with the same role Theme, the round-trip conversion would generate two events: one with A as Theme and one with B as Theme.

4 Markov Logic

Markov Logic (Richardson and Domingos, 2006) is a Statistical Relational Learning language

Algorithm 2 link to event conversion. Assume: no cycles; tokens can only be one of protein, site or event; binding events have only protein arguments.

```
/* returns all events E specified
   by clues C and links L */
function linksToEvents(C, L)
  return ⋃_{(i,t)∈C} resolve(i, C, L)

/* returns all events for
   the given token i */
function resolve(i, C, L)
  if no t with (i, t) ∈ C return {i}
  t ← type(i, C)
  if t = binding return {(i, t, A)} with
    A = {(a, r) | (i, a, r) ∈ L}
    R_i ← {r' | ∃a : (i, a, r) ∈ L}
    for each role r ∈ R_i do
      A_r ← {a | (i, a, r) ∈ L}
      B_r ← ⋃_{a∈A_r} {(resolve(a), r)}
  return ⋃_{A∈expand(B_{r_1},...,B_{r_n})} {(i, t, A)}

/* returns all possible argument
   sets for B_{r_1}, ..., B_{r_n} */
function expand(B_{r_1}, ..., B_{r_n})
  if n = 1 return B_{r_n}
  return
  ⋃_{a∈B_{r_1}} ⋃_{A∈expand(B_{r_2},...,B_{r_n})} {(a, r_1)} ∪ A
```

based on First Order Logic and Markov Networks. It can be seen as a formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative point of view, it is an expressive template language that uses First Order Logic formulae to instantiate Markov Networks of repetitive structure.

Let us introduce Markov Logic by considering the event extraction task (as relational structure over tokens as generated by algorithm 1). In Markov Logic we can model this task by first introducing a set of logical predicates such as *eventType(Token,Type)*, *role(Token,Token,Role)* and *word(Token,Word)*. Then we specify a set of weighted first order formulae that define a distribution over sets of ground atoms of these predicates (or so-called *possible worlds*). Note that we will refer predicates such as *word* as *observed* because they are known in advance. In contrast, *role* is *hidden* because we need to infer its ground atoms at test time.

Ideally, the distribution we define with these weighted formulae assigns high probability to possible worlds where events are correctly identified and a low probability to worlds where this is not the case. For example, in our running example a suitable set of weighted formulae would assign a higher probability to the world

$$\{word(1, \text{prevented}), eventType(1, \text{neg\_reg}), role(1, 2, \text{Theme}), event(2), \ldots\}$$

than to the world

$$\{word(1, \text{prevented}), eventType(1, \text{binding}), role(1, 2, \text{Theme}), event(2), \ldots\}$$

In Markov Logic a set of weighted first order formulae is called a *Markov Logic Network* (MLN). Formally speaking, an MLN M is a set of pairs (ϕ, w) where ϕ is a first order formula and w a real weigh t. M assigns the probability

$$p(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{(\phi,w)\in M} w \sum_{\mathbf{c}\in C^{\phi}} f^{\phi}_{\mathbf{c}}(\mathbf{y}) \right) \quad (4)$$

to the possible world $\mathbf{y}$. Here C^{ϕ} is the set of all possible bindings of the free variables in ϕ with the constants of our domain. $f^{\phi}_{\mathbf{c}}$ is a feature function that returns 1 if in the possible world $\mathbf{y}$ the *ground formula* we get by replacing the free variables in ϕ by the constants in the binding $\mathbf{c}$ is true and 0 otherwise. Z is a normalisation constant.

4.1 Inference and Learning

Assuming that we have an MLN, a set of weights and a given sentence, we need to predict the choice of event clues and roles with maximal *a posteriori* probability (MAP). To this end we apply a method that is both exact and efficient: Cutting Plane Inference Riedel (2008, CPI) with Integer Linear Programming (ILP) as *base solver*.

In order to learn the weights of the MLN we use the 1-best MIRA Crammer and Singer (2003) Online Learning method. As MAP inference method that is applied in the inner loop of the online learner we apply CPI, again with ILP as base solver. The loss function for MIRA is a

weighted sum $FP + \alpha FN$ where FP is the number of false positives, FN the number of false negatives and $\alpha = 0.01$.

5 Markov Logic Network for Event Extraction

We define four hidden predicates our task: *event(i)* indicates that there is an event with clue word i; *eventType(i,t)* denotes that at token i there is an event with type t; *site(i)* denotes a cellular location mentioned at token i; *role(i,j,r)* indicates that token i has the argument j with role r. In other words, the four hidden predicates represent the set of sites L (via *site*), the set of event clues C (via *event* and *eventType*) and the set of links L (via *role*) presented in section 3.

There are numerous observed predicates we use. Firstly, the provided information about protein mentions is captured by the predicate *protein(i)*, indicating there is a protein mention ending at token i. We also describe event types and roles in more detail: *regType(*t) holds for an event type t iff it is a regulation event type; *task1Role*(r) and *task2Role*(r) hold for a role r if is a role of task 1 (Theme, Cause) or task 2 (Site, CSite, etc.).

Furthermore, we use predicates that describe properties of tokens (such as the word or stem of a token) and token pairs (such as the dependency between two tokens); this set is presented in table 1. Here the *path* and *pathNL* predicates may need some further explanation. When *path(i,j,p,parser)* is true, there must be a labelled dependency path p between i and j according to the parser *parser*. For example, in figure 1 we will observe *path*(1,5,dobj↓prep_of↓,mcclosky-charniak). pathNL just omits the dependency labels, leading to *path*(1,5,↓↓,mcclosky-charniak) for the same example.

We use two parses per sentence: the outputs of a self-trained reranking parser Charniak and Johnson (2005); McClosky and Charniak (2008) and a CCG parser (Clark and Curran, 2007), provided as part of the shared task dataset. As dictionaries we use a collection of cellular location terms taken from the Genia event corpus (Kim et al., 2008), a small handpicked set of event triggers and a list of English stop words.

Predicate	Description
word(i,w)	Token i has word w.
stem(i,s)	i has (Porter) stem s.
pos(i,p)	i has POS tag p.
hyphen(i,w)	i has word w after last hyphen.
hyphenStem(i,s)	i has stem s after last hyphen.
dict(i,d)	i appears in dictionary d.
genia(i,p)	i is event clue in the Genia corpus with precision p.
dep(i,j,d,parser)	i is head of token j with dependency d according to parser *parser*.
path(i,j,p,parser)	Labelled Dependency path according to parser *parser* between tokens i and j is p.
pathNL(i,j,p,parser)	Unlabelled dependency path according to parser p between tokens i and j is *path*.

Table 1: Observable predicates for token and token pair properties.

5.1 Local Formulae

A formula is *local* if its groundings relate any number of observed ground atoms to exactly one hidden ground atom. For example, the grounding

$$dep\,(1, 2, \text{dobj}, \text{ccg}) \wedge word\,(1, \text{prevented}) \Rightarrow eventType\,(2, \text{pos\_reg}) \quad (5)$$

of the local formula

$$dep(h, i, d, parser) \wedge word\,(h, +w) \Rightarrow eventType(i, +t) \quad (6)$$

connects a single hidden *eventType* ground atom with an observed *word* and *dep* atom. Note that the "+" prefix for variables indicates that there is a different weight for each possible pair of word and event type (w, t).

5.1.1 Local Entity Formulae

The local formulae for the hidden *event/1* predicate can be summarized as follows. First, we add a $event\,(i)$ formula that postulates the existence of an event for each token. The weight of this formulae serves as a general bias for or against the existence of events.

Next, we add one formula

$$T(i, +t) \Rightarrow event(i) \quad (7)$$

for each "simple token property" predicate T in table 1 (those in the first section of the table). For example, when we plug in *word* for T we get a formula that encourages or discourages the existence of an event token based on the word form of the current token: $word(i, +t) \Rightarrow event(i)$.

We also add the formula

$$genia(i, p) \Rightarrow event(i) \quad (8)$$

and multiply the feature-weight product for each of its groundings with the precision p. This is corresponds to so-called real-valued feature functions, and allows us to incorporate probabilities and other numeric quantities in a principled fashion.

Finally, we add a version of formula 6 where we replace *eventType(i,t)* with *event(i)*.

For the cellular location *site* predicate we use exactly the same set of formulae but replace every occurrence of *event(i)* with *site(i)*. This demonstrates the ease with which we could tackle task 2: apart from a small set of global formulae we introduce later, we did not have to do more than copy one file (the event model file) and perform a search-and-replace. Likewise, in the case of the *eventType* predicate we simply replace *event(i)* with *eventType(i,+t)*.

5.1.2 Local Link Formulae

The local formulae for the *role/3* predicate are different in nature because they assess two tokens and their relation. However, the first formula does look familiar: $role(i, j, +r)$. This formula captures a (role-dependent) bias for the existence of a role between any two tokens.

The next formula we add is

$$dict(i, +d_i) \wedge dict(j, +d_j) \Rightarrow role(i, j, +r) \quad (9)$$

and assesses each combination of dictionaries that the event and argument token are part of. Furthermore, we add the formula

$$path(i, j, +p, +parser) \Rightarrow role(i, j, +r) \quad (10)$$

that relates the dependency path between two tokens i and j with the role that j plays with respect to i. We also add an unlabelled version of this formula (using *pathNL* instead of *path*). Finally, we add a formula

$$P(i, j, +p, +parser) \wedge T(i, +t) \Rightarrow role(i, j, +r) \quad (11)$$

for each P in {*path,pathNL*} and T in {*word,stem,pos,dict,protein*}. Note that for T=*protein* we replace $T(i, +t)$ with $T(i)$.

5.2 Global Formulae

Global formulae relate two or more hidden ground atoms. For example, the formula in equation 1 is global. While local formulae can be used in any conventional classifier (in the form of feature functions conditioned only on the input data) this does not hold for global ones. We could enforce global constraints such as the formula in equation 1 by building up structure incrementally (e.g. start with one classifier for events and sites, and then predict roles between events and arguments with another). However, this does not solve the typical chicken-and-egg problem: evidence for possible arguments could help us to predict the existence of event clues, and evidence for events help us to predict arguments. By contrast, global formulae can capture this type of correlation very naturally.

Table 2 shows the global formulae we use. We divide them into three parts. The first set of formulae (CORE) ensures that *event* and *eventType* atoms are consistent. In all our experiments we will always include all CORE formulae; without them we might return meaningless solutions that have events with no event types, or types without events.

The second set of formulae (VALID) consist of CORE and formulae that ensure that the link structure represents a valid set of events. For example, this includes formula 12 that enforces each event to have at least one theme.

Finally, FULL includes VALID and two constraints that are not strictly necessary to enforce valid event structures. However, they do help us to improve performance. Formula 14 forbids a token to be argument of more than one event. In fact, this formula does not hold all the time, but

#	Formula	Description
1	$event(i) \Rightarrow \exists t.eventType(i,t)$	If there is an event there should be an event type.
2	$eventType(i,t) \Rightarrow event(i)$	If there is an event type there should be an event.
3	$eventType(i,t) \wedge t \neq o \Rightarrow \neg eventType(i,o)$	There cannot be more than one event type per token.
4	$\neg site(i) \vee \neg event(i)$	A token cannot be both be event and site.
5	$role(i,j,r) \Rightarrow event(i)$	If j plays the role r for i then i has to be an event.
6	$role(i,j,r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(i,j,r_2)$	There cannot be more than one role per argument.
7	$eventType(e,t) \wedge role(e,a,r) \wedge event(a) \Rightarrow regType(t)$	Only reg. type events can have event arguments.
9	$role(i,j,r) \wedge taskOne(r) \Rightarrow event(j) \vee protein(j)$	For task 1 roles arguments must be proteins or events
10	$role(i,j,r) \wedge taskTwo(r) \Rightarrow site(j)$	Task 2 arguments must be cellular locations ($site$).
11	$site(j) \Rightarrow \exists i,r.role(i,j,r) \wedge taskTwo(r)$	Sites are always associated with an event.
12	$event(i) \Rightarrow \exists j.role(i,j,\text{Theme})$	Every events need a theme.
13	$eventType(i,t) \wedge \neg allowed(t,r) \Rightarrow \neg role(i,j,r)$	Certain events may not have certain roles.
14	$role(i,j,r_1) \wedge k \neq i \Rightarrow \neg role(k,j,r_2)$	A token cannot be argument of more than one event.
15	$j < k \wedge i < j \wedge role(i,j,r_1) \Rightarrow \neg role(i,k,r_2)$	No inside outside chains.

Table 2: All three sets of global formulae used: CORE (1-3), VALID (1-13), FULL (1-15).

by adding it we could improve performance. Formula 15 is our answer to a type of event "chain" that earlier models would tend to produce.

Note that all formulae but formula 15 are deterministic. This amounts to giving them a very high/infinite weight in advance (and not learning it during training).

6 Results

In table 3 we can see our results for task 1 and 2 of the shared task. The measures we present here correspond to the "approximate span, approximate recursive match" criterion that counts an event as correctly predicted if all arguments are extracted and the event clue tokens approximately match the gold clue tokens. For more details on this metric we refer the reader to the shared task overview paper.

To put our results into context: for task 1 we reached the 4th place among 20 participants, are in close range to place 2 and 3, and significantly outperform the 5th best entry. Moreover, we had highest scoring scores for task 2 with a 13% margin to the runner-up. Using both training and development set for training (as allowed by the task organisers), our task 1 score rises to 45.1, slightly higher than the score of the current third.

In terms of accuracy across different event types our model performs worse for binding, regulation type and transcription events. Binding events are inherently harder to correctly extract because they often have multiple core arguments while other non-regulation events have only one; just missing one of the binding arguments will lead to an event that is considered as error with no partial credit given. If we would give credit for binding with partially correct arguments our F-score for binding events would rise to 49.8.

One reason why regulation events are difficult to extract is the fact that they often have arguments which themselves are events, too. In this case our recall is bound by the recall for argument events because we can never find a regulation event if we cannot predict the argument event. Note that we are still unsure about transcription events, in particular because we observe 49% F-score for such events in the development set.

How does our model benefit from the global formulae we describe in section 5 (and which represent one of the core benefits of a Markov Logic approach)? To evaluate this we compare our FULL model with CORE and VALID from table 2. Note that because the evaluation interface rejects invalid event structures, we cannot use the evaluation metrics of the shared task. Instead we use table 4 to present an evaluation in terms of ground atom F1-score for the hidden predicates of our model. This amounts to a per-

	Task 1			Task 2		
	R	P	F	R	P	F
Loc	37.9	88.0	53.0	32.8	76.0	45.8
Bind	23.1	48.2	31.2	22.4	47.0	30.3
Expr	63.0	75.1	68.5	63.0	75.1	68.5
Trans	16.8	29.9	21.5	16.8	29.9	21.5
Cata	64.3	81.8	72.0	64.3	81.8	72.0
Phos	78.5	77.4	77.9	69.1	70.1	69.6
Total	48.3	68.9	56.8	46.8	67.0	55.1
Reg	23.7	40.8	30.0	22.3	38.5	28.2
Pos	26.8	42.8	32.9	26.7	42.3	32.7
Neg	27.2	40.2	32.4	26.1	38.6	31.2
Total	26.3	41.8	32.3	25.8	40.8	31.6
Total	36.9	55.6	44.4	35.9	54.1	43.1

Table 3: (R)ecall, (P)recision, and (F)-Score for task 1 and 2 in terms of event types.

role, per-site and per-event-clue evaluation. The numbers here will not directly correspond to actual scores, but generally we can assume that if we do better in our metrics, we will likely have better scores.

In table 4 we notice that ensuring consistency between all predicates has a significant impact on the performance across the board (see the VALID results). Furthermore, when adding extra formulae that are not strictly necessary for consistency, but which encourage more likely event structure, we again see significant improvements (see FULL results). Interestingly, although the extra formulae only directly consider *role* atoms, they also have a significant impact on event and particularly site extraction performance. This reflects how in a joint model decisions which would appear in the end of a traditional pipeline (e.g., extracting roles for events) can help steps that would appear in the beginning (extracting events and sites).

For the about 7500 sentences in the training set we need about 3 hours on a MacBook Pro with 2.8Ghz and 4Gb RAM to learn the weights of our MLN. This allowed us to try different sets of formulae in relatively short time.

7 Conclusion

Our approach the BioNLP Shared Task 2009 can be characterized by three decisions: (a) jointly modelling the complete event structure for a given sentence; (b) using Markov Logic as general purpose-framework in order to implement our joint model; (c) framing the problem as a link prediction problem between tokens of a sentence.

	CORE	VALID	FULL
eventType	52.8	63.2	64.3
role	44.0	53.5	55.7
site	42.0	46.0	51.5
Total	50.7	60.1	61.9

Table 4: Ground atom F-scores for global formulae.

Our results are competitive: we reach the 4th place in task 1 and the 1st place for task 2 (with a 13% margin). Furthermore, the declarative nature of Markov Logic helped us to achieve these results with a moderate amount of engineering. In particular, we were able to tackle task 2 by copying the local formulae for event prediction, and adding three global formulae (4, 10 and 11 in table 2). Finally, our system was fast to train (3 hours) . This greatly simplified the search for good sets of formulae.

We have also shown that global formulae significantly improve performance in terms of event clue, site and argument prediction. While a similar effect may be possible with reranking architectures, we believe that in terms of implementation efforts our approach is at least as simple. In fact, our main effort lied in the conversion to link prediction, not in learning or inference. In future work we will therefore investigate means to extend Markov Logic (interpreter) in order to directly model event structure.

Acknowledgements

We thank Dr. Chisato Yamasaki and Dr. Tadashi Imanishi, BIRC, AIST, for their help. This work is supported by the Integrated Database Project (MEXT, Japan), the Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and the Genome Network Project (MEXT, Japan).

References

Charniak, Eugene 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.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Comput. Linguist.* 33(4):493–552.

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

Kim, Jin D., Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics* 9(1).

Kim, Jin-Dong, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

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

Meza-Ruiz, Ivan 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)*.

Richardson, Matt and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Riedel, Sebastian. 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)*.

Riedel, Sebastian and Ivan Meza-Ruiz. 2008. Collective semantic role labelling with markov logic. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL' 08)*. pages 193–197.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.