Computational Intelligence, Volume Special Issue: Extracting bio-molecular events from literature
Chief Guest Editor: Jin-Dong Kim (jdkim@is.s.u-tokyo.ac.jp) , Number 000, 2009

Bio-Molecular Event Extraction with Markov Logic

Sebastian Riedel
Department of Computer Science
University of Massachusetts Amherst, Amherst, USA

Rune Sætre
Department of Computer Science
University of Tokyo, Japan

Hong-Woo Chun
Knowledge Information Center
Korea Institute of Science and Technology Information, Republic of Korea

Toshihisa Takagi
Database Center for Life Science
Research Organization of Information and System, Japan

Jun'ichi Tsujii
Department of Computer Science
University of Tokyo, Japan

This article presents a novel approach to event extraction from biological text using Markov Logic. It can be described by three design decisions: (1) instead of building a pipeline using local classifiers, 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 relations over the token indices of a sentence, as opposed to structures that relate event entities to gene or protein mentions.

In this article we extend our original work (Riedel *et al.*, 2009) by providing an error analysis for binding events. Moreover, we investigate the impact of different loss functions to precision, recall and F-measure. Finally, we show how to extract events of different types that share the same event clue. This extension allowed us to improve our performance our performance even further, leading to the third best scores for task 1 (in close range to the second place) and the best results for task 2 with a 14 percent point margin.

Key words: Event Extraction, Joint Inference, Markov Logic, BioNLP

[1] Address correspondence to Sebastian Riedel, Department of Computer Science, University of Massachusetts Amherst, 140 Governors Drive Amherst, MA 01003-9264; email: riedel@cs.umass.edu.

© 2009 The Authors. Journal Compilation © 2009 Wiley Periodicals, Inc.

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) was arranged to extract such events from text. In the course of the shared task, the organisers provided a training/development set of abstracts for biomedical papers, annotated with events mentioned in the text. Participants were required to use this data in order to engineer an event detector 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. Event E14 describes a "Positive_Regulation" event and is bound to the clue word "induction". The theme of this event is another event: the gene expression event E15. The cause of E14 is the protein "gp41", bound to the term "gp41" in the sentence. Note that E14 itself is an argument to the "Negative_regulation" event E13.

The second task was an extension of the first one, requiring participants to not only detect the core arguments of each event, but also the cellular site(s) the event is associated with in the text. The events in this task were hence similar to those in Figure 1, but would also have arguments that are cellular location/site terms. In contrast to the protein terms, cellular location terms were not given as input and had to be detected, 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). It also shares the spirit of recent work in *collective* Information Extraction (Bunescu and Mooney, 2004; Finkel *et al.*, 2005; Poon and Domingos, 2007). Here the limitations of local classifier, linear chain and pipeline approaches are overcome by using complex Markov Networks to capture global correlation between variables.

We can characterise our approach by three decisions that we will illustrate in the following. First, as opposed to the other participants of the shared task (Kim *et al.*, 2009) and other existing BioNLP extractors (Sætre *et al.*, 2009), we do not build a pipelined system that first detects event clues and cellular locations, and then relations between these. Instead, we propose 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. 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 prediction[1] model (for example, as done by Toutanova *et al.* (2005) and McDonald and Pereira (2006)) 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 (an event must be a regulation if its theme is an event) is captured by the simple formula:

$$\begin{aligned} &\text{eventType}\,(e,t) \wedge \text{role}\,(e,a,r) \wedge \text{event}\,(a) \Rightarrow \\ &\qquad \text{regType}\,(t) \end{aligned} \tag{1}$$

[1]Note that we use the term *structured prediction* in the Machine Learning sense, where the task is to detect/predict a complex structure as opposed to simple class labels.

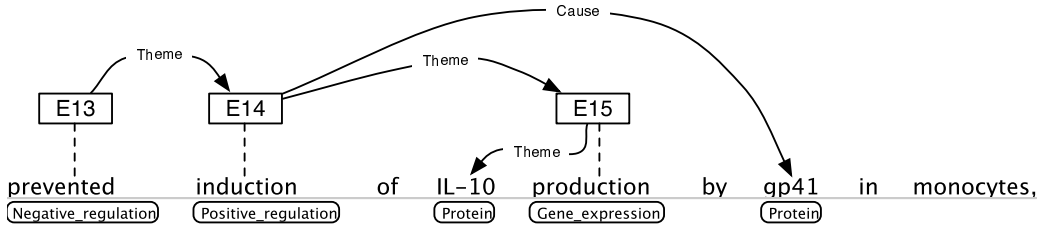

Figure 1. Example gold annotation for task 1 of the shared task.

Third, we represent events as **relations over token indices** of a sentence (see Figure 2), as opposed to structures that relate event entities to gene/protein mentions (see Figure 1).[2] That is, instead of directly detecting events we detect relations between token indices which are converted to events in a post-processing step. The reason is as follows. Markov Logic, for the time being, is tailored to *link detection*[3] problems where we make inferences about the existence of relations between given entities. However, when the identity and number of objects of our domain is unknown, both modelling and inference become more complicated. By mapping to a relational structure over token indices, we also show a direct connection to recent formulations of Semantic Role Labelling which may be helpful in the future. Note that we do not argue that this approach is unique (virtually all other participating systems followed the same path); it is, however, important design choice that influenced both our model and pre/post-processing steps.

This article is based on the work of Riedel *et al.* (2009) but makes the following additional contributions. First, we show how our model can be extended to cases where events of different types have the same event trigger. This is the case in sentences such as

Our results suggest that **overexpression** of v-erbA is required for its function as an oncoprotein.

Here the word "overexpression" is an event clue for both an expression and a positive regulation event. Our original model would only be able to detect the expression event—now we can extract the regulation event, too. This is conveniently achieved through the addition of a small set of formulae and lead to a 1% improvement in total F1 score. More importantly, it shows the flexibility Markov Logic gives us when modelling problems.

Second, we investigate the impact of tuning the loss function we are minimising during training to gain either higher recall or precision. While rather technical in nature, this simple tuning made a dramatic difference to our final performance.

Finally, we include an analysis of our biggest weakness: binding events. This analysis shows where some of the errors come from, and how they could be avoided in the future.

The remainder of this article is organised as follows: we will first present the preprocessing steps we perform (section 2), then the conversion to a link detection problem (section 3). Subsequently, we will describe Markov Logic (section 4) and our Markov Logic Network for event extraction (section 5). We will then explain how we detected events of different types with identical event triggers (section 6). Finally, we present our results (in section 7), a detailed error analysis for binding events (section 8) and conclude (section 10).

2. PREPROCESSING

As mentioned above, we map the event structures provided in the shared task dataset to edges, or *links*, between token indices of the sentence, and hence event recognition to link detection (Taskar

[2]Note that in the following we will often identify the term "token" with "token index", as it is common in Natural Language Processing.

[3]In Machine Learning we also use the term *link prediction*; however, since in biology the term "prediction" is used for hypothesising interactions before they are observed, we use the term "detection" instead.

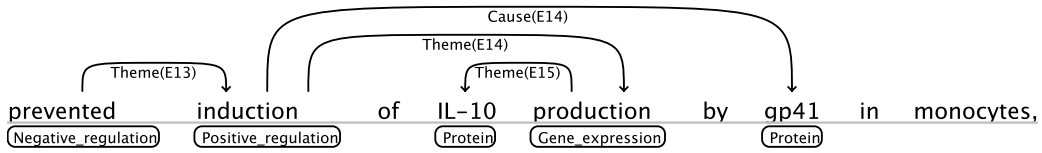

FIGURE 2. Link detection version of the events in Figure 1.

et al., 2003). Here each (labelled) link between two tokens A and B represents the fact that the gene/protein or location represented by token B (or that ends at token B) is a semantic argument of the event represented by the event clue A with a role that corresponds to the link label.

However, before we map the event extraction task to a link detection task, we perform several preprocessing steps that we will describe below. First we need to give the reader a better idea of the provided data. The original data format provided by the shared task organisers consists of (a) a collection of biomedical abstracts, and (b) standoff annotation that describes the proteins, events and sites mentioned in these abstracts. The organisers also provided a set of dependency and constituent parses for the abstracts. Note that these parses are based on a different tokenization 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 tokenization used in the parses. This allows us to formulate our probabilistic model in terms of one consistent tokenization (and be able to speak of token instead of character offsets).

The new tokenization is still not optimal: sometimes several protein mentions can appear in one token ("p50/p55" is such a case). In a token-based approach this complicates matters. Hence we retokenize the input text (and the corresponding parse trees) according the protein boundaries that were given in the shared task data. Finally, this tokenization is used to once again adapt the stand-off annotation (using the previously adapted version as input).

The data also contains events that have arguments in preceding sentences. In its current state, this type of event cannot be tackled with our model, and hence we remove these cases from the data-set.

3. LINK PREDICTION REPRESENTATION

We convert the task to link detection (Taskar *et al.*, 2003; Bilgic *et al.*, 2007) 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/site or event, and r is the role a plays with respect to e. Notice that in this representation we identify each protein or site with the index of its rightmost token. Finally, we will identify the set of all proteins, location/sites and events for a sentence with P, S and E, respectively.

For example, in Figure 1 we have $P = \{4, 7\}, S = \emptyset$ 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 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

Algorithm 1 Event to link conversion

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

labelled token-to-token links (i, j, r), and C is a set of labelled event clues (i, t). Here i and j refer to token indices, and t to an event type.

Note that this mapping has another benefit: it creates a "predicate-argument" structure very similar to recent formulations of Semantic Role Labelling (Surdeanu *et al.*, 2008). That is, pairs of token indices identify the syntactic heads of predicates and their semantic arguments. Hence it may be possible to re-use or adapt successful approaches in Semantic Role Labelling in order to improve bio-molecular event extraction. Since our approach is inspired by the Markov Logic role-labeller of Riedel and Meza-Ruiz (2008), this work can be seen as an attempt in this direction.

For a sentence with given P, S and E, Algorithm 1 presents our mapping from E to (L, C). After initialising both L and C, we iterate over all events (i, t, A) in E that have no arguments with the same event clue i—we show in section 6 how we can tackle events that fall outside of this class. For a given event e we first create a clue element in C using the clue index and the type of e (Step 4). Then we iterate over all arguments of the event, and for each one we create a labelled link from the clue index to the argument index based on the label of the argument. Notice that in case of protein or site arguments in step 9 we directly use them as indices (as we represented sites and proteins by their token index); in the case of event arguments (step 7) we use the clue index of the event as target of the generated link.

For our running example in Figure 1, eventsToLinks would return

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

and

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

corresponding to Figure 2.

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 and use it). This mapping is presented in algorithm 2 and discussed in the following. Note that we expect the relational structure E we detect to be cycle free.

The function linksToEvent takes a set of event clues C and the event-to-argument links L. It transforms this input to an event structure by calling the resolve function for all event clues in C and joining its results.

The resolve method returns, for a given event clue token i and the pair (C, L), all event structures described by (C, L) that have i has clue index. How this is done depends on the event type corresponding to the clue index (as provided by the set C). If the clue index corresponds to a binding event type, we simply create a single event with the given type (i.e. Binding) and clue index (step 4). The arguments of the newly created event correspond to all tokens that the clue index is

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 */
1 function linksToEvents(C, L)
2   return ⋃_{(i,t)∈C} resolve(i, C, L)

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

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

mapped to in the link structure L; the labels of these arguments correspond to the labels of the corresponding links in L. Note that this assumes each link target of a binding event in L to point to proteins or sites (otherwise we would need to resolve the target index as well).

Notice that this heuristic for binding events is based on the hypothesis that each binding event trigger corresponds to exactly one binding event. In section 8 we show that this assumption does *not* hold. However, some preliminary experiments with a heuristic similar to the one of Björne *et al.* (2009) did not lead to improvements. Possibly because we do not detect enough binding links to begin with.

In case of events that do not describe bindings we proceed as follows. First, we collect all the roles of all arguments of i in the link structure (step 6). Then, for each role r we collect all arguments of i with this role (step 8), and convert them to the events, proteins or sites they refer to (step 9). This yields a set of events/sites/proteins B_r for each role r. Finally we return one event for each possible set of arguments A, as generated by the *expand* function (step 10); Notice that A always contains exactly one argument for each role we have found to be associated with i in L.

If we re-converted C and L from Equation 2 and 3, respectively, we would return to our original event structure in Figure 1. However, converting back and forth is not loss-free in general. If we had a non-binding event in the original E set with two arguments A and B that both have the Theme role, 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 (Getoor and Taskar, 2007) language 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 link detection

between tokens). In Markov Logic we can model this task by first introducing a set of predicates such as:

- eventType(i,t) that holds between a token index i and an event type t if and only if the event for the clue at the token i has the type t;
- role(e, a, r) that holds for the token indices e and a, and the event argument role r, if and only if the gene, protein or cellular location mentioned at a has the semantic role r with respect to the event mentioned at token index e;
- pos(i, p) that holds between token index i and POS-tag p if and only if the token at index i has the tag p.

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*). Ideally, this distribution 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 (Figure 2) a suitable set of weighted formulae would assign a higher probability to the world

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

than to the world

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

In Markov Logic a set $M = \{(\phi, w_\phi)\}_\phi$ of weighted first order formulae is called a *Markov Logic Network* (MLN). Here each ϕ is a first order formula, and each w is a real-valued weight (that can be positive or negative). Note that weights does not have sum up to one.

An MLN assigns the probability

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

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. Note that this distribution corresponds to a Markov Network (the so-called *Ground Markov Network*) where nodes represent ground atoms and factors represent ground formulae.

In the following we will refer to predicates such as pos/2 as *observed* because they are known in advance. In contrast, role/3 is *hidden* because we need to infer its ground atoms at test time.

Note that Markov Logic is surely not the only Statistical Relational Learning language. However, it is one of the few languages that support undirected Graphical Models (i.e., Markov Networks) and discriminative training. This type of model has been our focus because in most areas of supervised Machine Learning they have been shown to perform better than their generative counterparts. Another option for undirected discriminative models are Relational Markov Networks (Taskar *et al.*, 2002). Yet, to this date there exists no publicly available software that supports them. By contrast, several open-source interpreters for Markov Logic exists and are under active development (Riedel, 2008b; Kok *et al.*, 2005).

Finally, instead of using Markov Logic to define our Markov Networks for each sentence, we could have manually created them. However, this (a) requires more engineering, and (b) renders the application of inference algorithms such as Cutting Plane Inference (Riedel, 2008a) or Lifted Belief Propagation (Singla and Domingos, 2008) difficult. These algorithms perform inference on subsets, or abstract versions, of the full propositional Markov Network and exploit the first order information captured in the MLN. For many applications this leads to dramatic speed-ups when compared to inference in the full network.

Note that we use the open source Markov Logic interpreter *thebeast* (Riedel, 2008b) as inference and learning engine for our MLNs. thebeast has been optimised for Natural Language Processing

applications, and supports Cutting Plane Inference and max-margin learning with MIRA (see Section 4.1).

4.1. Inference and Learning

Assuming that we have an MLN, a set of weights and a given sentence, we need to find the 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, 2008a, CPI) with Integer Linear Programming (ILP) as *base solver*.

Instead of fully instantiating the Markov Network that a Markov Logic Network describes, CPI begins with a subset of factors/edges—in our case we use the factors that correspond to the local formulae of our model—and solves the MAP problem for this subset using the base solver. It then inspects the solution for ground formulae/features that are not yet included but could, if added, lead to a different solution—this is usually referred to as separation. The ground formulae that we have found are added and the network is solved again. This process is repeated until the network does not change anymore.

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 we use for MIRA is a weighted sum $FP+\alpha FN$ where FP is the number of false positives and FN the number of false negatives. With $\alpha = 1$ we observed a high discrepancy between precision and recall, leading to low F1 scores. By setting $\alpha = 0.01$ (and hence penalising false negatives 100 times more than false positives), this problem was overcome.

5. MARKOV LOGIC NETWORK FOR EVENT EXTRACTION

Akin to the example in section 4, we define four hidden predicates our task: event(i) indicates that there is an event that has its clue word at token index i; eventType(i,t) denotes that at token index i there is an event with type t; site(i) denotes a cellular location mentioned at token index i; role(i,j,r) indicates that token index i has an argument at index j that plays the role r. In other words, the four hidden predicates represent the set of sites S (via site/1), the set of event clues C (via event/1 and eventType/2) and the set of links L (via role/3) presented in section 3.

There are several observed predicates we use. Firstly, we are given ground truth information about the proteins in a sentence. This information is captured by the predicate protein(i), indicating that we find a protein mention at token index i.[4] We also provide predicates that allow us to 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 a set of predicates that describe properties of tokens (such as the word or stem of a token) and token pairs (such as the syntactic dependency between two tokens); this set is presented in Table 1. While most of the predicates (and descriptions) in Table 1 should be intuitively clear, the path/3 and pathNL/3 predicates may need some further explanation. When path(i,j,p,*parser*) is true, there must be a labelled dependency path p between token index i and j according to the parser *parser*. This path is the sequence of dependency labels and directions (head to modifier or modifier to head) one has to pass in order to get from i to j in the tree extracted by *parser*. pathNL/3 is defined in a similar way, it just omits the dependency labels.

For each sentence we use two dependency parses: one based on the parser of Charniak and Johnson (2005) using a self-trained biomedical parsing model (McClosky and Charniak, 2008), the other based on a CCG parser (Clark and Curran, 2007). Both types of parses were provided as part of the shared task dataset. Hence, for Figure 1 we would observe, among others, path(1,5,dobj↓prep_of↓,mcclosky-charniak) and pathNL(1,5,↓↓,ccg).

[4] In case were protein mentions span several tokens *protein(i)* indicates that a protein mention ends at i.

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

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

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.

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

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

of the local formula[5]

$$\begin{aligned} \text{dep}(h, i, d, parser) \wedge \text{word}\,(h, +w) \Rightarrow \\ \text{eventType}(i, +t) \end{aligned} \tag{6}$$

connects a single hidden eventType/2 ground atom with an observed word/2 and dep/3 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). The corresponding factor graph snippet for this ground can be seen in Figure 3. Here we see observed ground atoms depicted as shaded vertices, and hidden ones as white vertices.

5.1.1. *Local Event Clue and Site Formulae.* The local formulae for the hidden event predicate can be summarised as follows. First, we add a formula that postulates the existence of an event for

[5]Here the grounding is created by substituting $h \leftarrow 1$, $i \leftarrow 2$, $d \leftarrow$ dobj, $parser \leftarrow$ ccg, $w \leftarrow$ prevented and $t \leftarrow$ pos_reg.

each token:

$$\text{event}\,(i) \tag{7}$$

The weight of this formulae serves as a general bias for or against the existence of events.

Second, we add one formula

$$T\,(i, +t) \Rightarrow \text{event}\,(i) \tag{8}$$

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/2 for T we get a formula that encourages or discourages the existence of a mentioned event based on the word form of the current token:

$$\text{word}\,(i, +t) \Rightarrow \text{event}\,(i) \tag{9}$$

Third, we add the formula

$$\text{genia}\,(i, p) \Rightarrow \text{event}\,(i) \tag{10}$$

Here *genia* associates each token index i with the percentage (or precision) p of times we have seen the word at this index labelled as event clue in the Genia corpus. Crucially, we multiply the feature-weight product for each of the formula's groundings with the value assigned to p in the grounding. This corresponds to so-called real-valued feature functions (because we multiply the formulae weight with a value between 0 and 1, as opposed to value that is either 0 or 1). They allow us to easily incorporate various types of numerical confidence values (such probabilities taken from a generative model) into our MLN.

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

For the cellular location site/1 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 operation.

Likewise, for the eventType/2 predicate we simply replace event(i) with eventType(i,$+t$) for each of the formulae described above.

5.1.2. *Local Event Role 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:

$$\text{role}\,(i, j, +r) \tag{11}$$

Akin to formula 7, this formula captures a (role-dependent) bias for the existence of a role between any two tokens.

The second formula we add is

$$\text{dict}\,(i, +d_i) \wedge \text{dict}\,(j, +d_j) \Rightarrow \text{role}\,(i, j, +r) \tag{12}$$

and assesses each combination of dictionaries that the event and argument token are part of.

Furthermore, we add the formula

$$\text{path}\,(i, j, +p, +parser) \Rightarrow \text{role}\,(i, j, +r) \tag{13}$$

that relates the dependency path between two token indices i and j with the role that j plays with respect to i. We also add an unlabelled version of this formula (using pathNL/3 instead of path/3).

Finally, we add a formula

$$\begin{gathered}\text{path}\,(i, j, +p, +parser) \wedge T\,(i, +t) \Rightarrow \\ \text{role}\,(i, j, +r)\end{gathered} \tag{14}$$

for each T in {word,stem,pos,dict} and a formula

$$\begin{gathered}\text{path}\,(i, j, +p, +parser) \wedge \text{protein}\,(j) \Rightarrow \\ \text{role}\,(i, j, +r)\end{gathered} \tag{15}$$

Again versions of the above sets of formulae for unlabelled paths are used in addition.

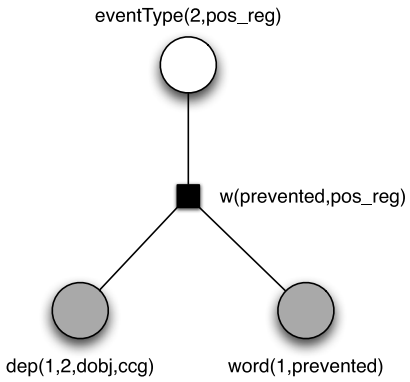

Figure 3. Grounding of a local formula with one hidden and three observed ground atom variable. The label of the factor indicates that the weight associated with it is parametrised by word and event type.

5.2. Global Formulae

Global formulae relate two or more hidden ground atoms. For example, the formula in Equation 1 is global, and a grounding of it can be seen in Figure 4. Here we have three hidden ground atoms (white vertices) in one factor.

While local formulae can be used in any conventional classifier (in the form of feature functions conditioned only on the input data and the candidate class) 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 detect roles between events and arguments with another). However, this does not solve the typical chicken-and-egg problem we encounter here: evidence for possible arguments can help us to detect the existence of event clues, and evidence for events can help us to detect 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/1 and eventType/2 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. Note that VALID also makes sure that events transitively involve proteins.[6] This is achieved through formula 9, which ensures that arguments are either proteins or themes, and formula 12.

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 (with infinite weight) a token to be an argument of more than one event. In fact, this formula does not hold all the time, but by adding it we could improve performance.[7] Formula 15 is our answer to a type of event "chain" that earlier models would tend to produce.

[6]This was an aspect of the data that made learning somewhat difficult. For example, there could be a grammatical construction that clearly indicates phosphorylation, and another one that indicates the regulation of this event. However, if no concrete protein is mentioned as the theme of the phosphorylation, neither the phosphorylation nor the regulation event will appear in the training or test set. Hence lexical and grammatical features alone will not be sufficient when it comes to detecting a regulation: Our model also needs to know whether a regulation is transitively involving a protein.

[7]The reason is that the task of only detecting one element out of n is easier for a learner than to detect both number and identity of elements. Moreover, there are not enough cases of arguments with multiple parents in order to offset this advantage.

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

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

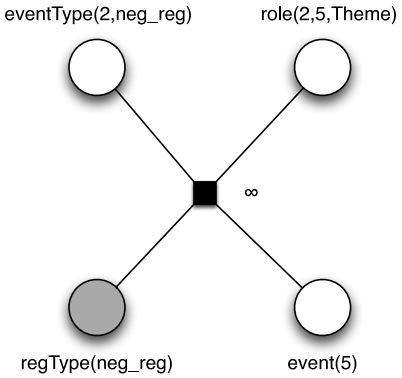

FIGURE 4. Grounding of a global formula with three hidden and one observed ground atom variable.

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. EVENT CLUES WITH MULTIPLE TYPES

In our original model we did not allow event clues with more than one type. This is enforced through formula 3 in Table 2. However, in practice we find this assumption to be sometimes violated. Consider, for example, the sentence

Our results suggest that **overexpression** of v-erbA is required for its function as an oncoprotein.

Here the word "overexpression" is an event clue for both an expression and a positive regulation event. This is an example of a more general scenario: some event triggers belong to both a regulation and the event the regulation modifies. We will refer to this as "same-trigger-regulation."

Notice this that this type of event expression is not very frequent: while there are 1385 of 6482 event clues in the training set that represent multiple events, only 125 of these have multiple events with different types. This means that by being able to detect such events we cannot dramatically improve our overall accuracy. However, since any mentioned event may be important, we cannot simply neglect them either.

One could try to write a simple heuristic to capture these cases. Alternatively, one could develop another classifier for this task. However, in the Markov Logic framework we can incorporate this kind of events in a principled manner by simply introducing one more predicate, sametrigger/2, defined over token indices and regulation types. Here sametrigger(i, t) holds if and only if there is a regulation event mentioned at token index i with type t, and the event that it is modifying has its event clue at index i, too. For this predicate we use the same local formulae as for eventType/2 and just replace eventType/2 with sametrigger/2. An example is:

$$\text{word}\,(i, +w) \Rightarrow \text{sametrigger}\,(i, +t) \tag{16}$$

In addition, we make sure that there is not more than one type for self-regulating event clues:

$$\text{sametrigger}\,(i, t) \wedge t \neq o \Rightarrow \neg\text{sametrigger}\,(i, o) \tag{17}$$

In fact, this alone would be equivalent to building another local classifier (yet, globally trained together with the other predicates). Again, with Markov Logic we are able to do more: we can enforce the constraint that if a token is a "same-trigger-regulation" clue it must also be the clue of a regular event. This can be easily captured by

$$\text{sametrigger}\,(i, t) \Rightarrow \text{event}\,(i) \tag{18}$$

This is all we have to do in order to adapt our link detection model. However, we also have to take care of "same-trigger-regulation" during pre-and post-processing. For pre-processing we simply augment the procedure in Algorithm 1 to map all same-trigger-regulation events to a list of same-trigger-regulation event clues. From these we create sametrigger/2 atoms in our input data. After our Markov Logic interpreter detected a set of events and same-trigger-regulation events, we first resolve the normal events (see Algorithm 2) and then iterate over the same-trigger-regulation event atoms. This creates one extra regulation event for each token marked as same-trigger-regulation clue according to the sametrigger/2 atoms, and each regular event that is associated with the same token.

7. EXPERIMENTS

With our experiments we try to answer the following questions. First, what is the impact of our global formulae? Second, how does the choice of loss function affect precision and recall? Third, do we benefit from modelling "same-trigger-regulation" as described in section 6? Finally, how do we fare compared to other participants of the shared task?

7.1. Impact of Global Formulae

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.

Unfortunately we cannot evaluate our CORE model using the evaluation metrics of the shared task because some of the results we detect with CORE are not valid event structures; in this case

	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 3. Performances on the development set for different sets of global formulae with respect to different hidden predicates and the global F1 score for all ground atoms (Total).

the evaluation interface simply rejects the input. Instead, we use Table 3 to present an evaluation in terms of ground atom F1-score for the different hidden predicates of our model. Roughly speaking, this amounts to a per-role, per-site and per-event evaluation, in contrast to an evaluation that scores events as a whole. We also present the total ground atom F1-score. The numbers here will not directly correspond to actual scores, but generally we can assume that if we do better in our metric, we will likely do better in the full metric, and vice versa.

In Table 3 we notice that explicitly adding formulae which ensure 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. In joint models this effect is not uncommon; it reflects how 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).

7.2. Loss Function

In section 4.1 we have described how we parametrised the loss function $FP+\alpha FN$ that is used within the MIRA Online learning update rule. Here the parameter α controls the impact of the number of false negatives for the loss of a given event structure. The higher α, the higher the price to pay for spurious arguments and events. Hence α gives us control over the precision/recall behaviour of our model. With a high α we expect higher precision, with a low α higher recall.

In Figure 5 we see how precision, recall and F1 score on the development set behaves with varying α on a log scale. As expected, increasing α does indeed improve precision while reducing recall. However, since we are optimising for F1 score (the harmonic mean of recall and precision), our operating point is chosen to be 0.01. Note that by setting α to high values we achieve precisions in the range of the highest precision of all shared task submissions Cohen *et al.* (2009) with similar performance on recall (around 25%).

Interestingly, recall seems to saturate at low values of α, even though this corresponds to only penalising false negatives. It may be possible to increase recall further by scaling up the contribution of false negatives to the loss function, instead of only reducing the contribution of false positives.

7.3. Same-Trigger-Regulation

We have described in section 6 how to model same-trigger-regulation events, but how much does this help? Table 4 shows the performance on the development set with respect to regulation events for our original model and for a model that includes explicit modelling of self-reference. We notice improvements for positive regulation events (which are in fact far more frequent than the other types of regulation events). In particular, we see that recall was increased by 3.5% points—we are now able to detect events that the previous model, by design, could not extract. Also note that performance for regulation and negative regulation dropped, although not enough to offset the increase for positive regulation. This may be the result of applying the same learning parameters (number of iterations, loss function) for both the original and extended model. In future work we will try to re-tune the parameters for this model.

How good is our model at detecting same-trigger-regulation events? To answer this question

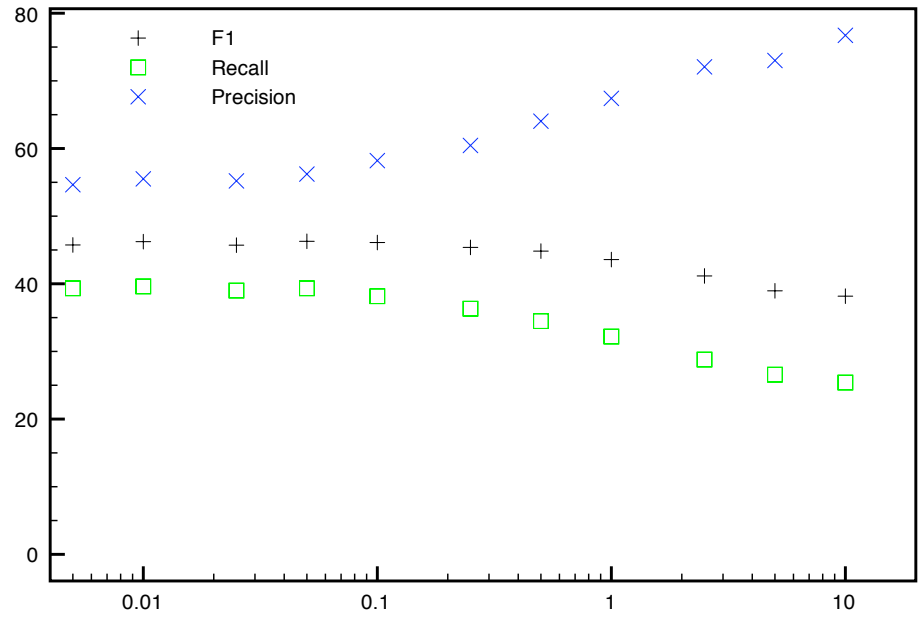

Figure 5. Dependency of precision, recall and F1 score (harmonic mean of precision and recall) on the alpha parameter that controls the loss function.

we look at precision, recall and F1 measure for sametrigger/2 ground atoms. Here we achieve a recall of 44.4%, a precision of 70% and a F1 measure of 54.2%. We expected this task to be easier than these numbers suggests, partly because in most cases same-trigger-regulation events have very clear lexical triggers (such as "overexpression"). The following might be an explanation what makes this task harder. While words such as "overexpression" indicate same-trigger-regulation in some sentences, it does not do so in others. For example, in

v-erbA **overexpression** is required to extinguish c-erbA function in erythroid cell differentiation and regulation of the erbA target gene CAII .

"overexpression" is not referring to a regulation event according to the shared task annotation. By contrast, in

Our results suggest that **overexpression** of v-erbA is required for its function as an oncoprotein.

it is. This will obviously confuse the learner at training time.

Table 4 shows that in terms of overall performance the improved extraction of positive regulations lead to a 1% point increase in F-measure on the development set. This is not a striking gain, mostly due to the fact that there are not many of these events to begin with. However, since the amount of engineering necessary to incorporate these events is small, we think that this extension is a significant contribution.

7.4. Test Set Performance

In Table 5 we can see our results on the test data of 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 detected 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: with our new model for same-trigger-regulation we reach an F

	W/o Self Regulation			With Self-regulation		
	R	P	F	R	P	F
Reg	34.9	44.7	39.2	33.7	43.5	38.0
Pos	32.9	47.9	39.0	36.6	48.6	41.8
Neg	27.6	39.1	32.3	25.5	36.2	29.9
Total	32.2	45.5	37.7	33.9	45.4	38.81
Total	42.2	57.3	48.6	43.6	57.3	49.5

TABLE 4. (R)ecall, (P)recision, and (F)-Score for task 1 on the development set.

	Task 1			Task 2		
	R	P	F	R	P	F
Expr.	64.4	74.5	69.1	64.4	74.5	69.1
Trans	16.1	37.3	22.5	16.1	37.3	22.5
Cata.	35.7	50.0	41.7	35.7	50.0	41.6
Phos	79.3	81.1	80.2	70.5	74.2	72.3
Loc	36.8	88.9	52.0	31.0	75.0	43.9
Bind	22.5	43.8	29.7	21.2	41.6	28.1
Total	48.5	68.9	56.9	46.8	67.0	55.0
Reg	22.7	36.5	28.0	21.6	33.8	26.6
Pos	29.9	45.1	35.96	29.7	43.5	35.6
Neg	27.9	41.9	33.5	26.9	40.3	32.3
Total	28.2	42.9	34.0	27.6	41.9	33.3
Total	37.9	55.8	45.2	36.8	54.3	43.9

TABLE 5. (R)ecall, (P)recision, and (F)-Score for task 1 and 2 in terms of event types.

score of 45.2% and this would put us on the third place among the 20 participants of the to the task. We would be in close reach to the second best system (Kilicoglu and Bergler, 2009) with 46.6% F-score, and still quite far away from the best-performing entry (Björne *et al.*, 2009) with 52.0% F-score. Note that our original submission (without a model for same-trigger-regulation) came in 4th with an F-score of 44.4%, and a 4% point margin to the next best system (Van Landeghem *et al.*, 2009).

Our new model leads to the highest scores for task 2 with a 14% margin to the runner-up. Also note that for these results we only trained on the training set, although participants were allowed to include the development set as well.

In terms of accuracy across different event types our model performs worse for binding and regulation type events. Some explanations as to why binding events are generally hard to extract, and why we are performing particularly poorly in this regard, are given in section 8.

One reason why regulation events are difficult to detect is the fact that they often have arguments which themselves are events, too (see Figure 1). In this case our recall is bound by the recall for argument events because we can never find a regulation event if we cannot detect the argument event.

Note that we are also doing poorly for transcription events. This may be due to overfitting on the types of transcription events mentioned in the training set. This view is supported by the fact that we observed results of up to 49% F-score for transcription events on the development set. It indicates that our model is not necessarily weak at detecting transcription events in general, but weak at detecting those events mentioned in the test set.

7.5. Runtime

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.

Team	Rank	R	P	F
UTurku	1	40.1	49.8	44.4
VIBGhent	2	38.0	38.6	38.3
JULIELab	3	49.6	35.3	41.2
UT+DBCLS	**4/3***	**22.5**	**43.8**	**29.7**
UNIMAN	5	20.5	40.6	27.2
ASU-+HU+BU	6	12.7	40.4	19.3
UTOKYO	7	34.6	50.6	41.1

Table 6. Binding results for the top 7 (according to total score) participating systems; * with results presented here.

8. BINDING EVENT ANALYSIS

Section 7 shows that we perform quite poorly in terms of binding events. Table 6 indicates that this is not only because the task seems to be hard in general. It shows the binding results (again using the "approximate span, approximate recursive match" criterion) for the top 7 participants on task 1. While our results (rank 4) are on-par with those of the next two best systems, they are significantly worse than those of the top 3 systems as well as the 7th best system.

In this section we seek to investigate reasons that lead to our poor performance for binding events. In particular, we focus on two issues: the post-processing heuristic we use for binding events (see section 3.2) and inconsistencies in the data. We also describe possible future work that could help to alleviate our problem with bindings.

8.1. Binding Event Post-processing Heuristic

The post-processing Algorithm 2 responsible for converting link structure to events makes a clear distinction between binding and non-binding events. For non-binding events we may create several events per event clue: one per combination of arguments attached to the clue word. However, for binding events we create exactly **one** event per clue, regardless of the number of possible argument combinations we could construct. This decision was based on the observation that binding events, in contrast to non-binding events, can have several Theme arguments.

Just how accurate is the assumption that each binding event clue corresponds to exactly one binding event? Table 7 clearly shows that it is wrong. We see, for example, that for the training/development set event clues with two proteins associated as "Theme" arguments in the link structure result in 1.44/1.34 events. In one extreme case in the training data, as many as 12 events (1% of the 887 events) were created based on a single binding clue. In total, the training and development corpora contain 1,136 events and only 853 trigger words. This shows that for optimal performance we should, on average, create 1.33 binding events per clue word.

These results show that we should investigate better heuristics, possibly along the lines of the method presented by Björne *et al.* (2009). Here arguments are clustered according to the syntactic relation/path to the event clue, and arguments in the same cluster are restricted from appearing in the same event. However, some observations indicate that this will only partially solve our problem. First, a preliminary implementation of a similar scheme lead to no improvements at all. In some cases arguments in the same syntactic relation to the clue did not participate in the same event and were therefore successfully resolved by our new heuristics. However, in other cases they were indeed part of the same event, and our new heuristic created more errors.[8]

Another reason why the post-processing heuristic is only a part of our problem can be seen in Table 8. The table shows a breakdown in terms of the number of themes for our binding errors on the development set. Note that we detect only 39 events with two themes, of which 19 are correct. Together with the 6 events we extract for three or more themes (and of which none are correct), this leaves only 26 events that a better post-processing heuristic could possibly improve upon. Likely

[8]For example, arguments appearing in a coordination belonged sometimes to one and sometimes to two events.

	avg. # event/trigger		
#associated arguments	train-set	dev-test	detected
1 Protein	1.02	1.00	1.00
2 Proteins	1.44	1.34	1.00
3 Proteins	2.36	2.40	1.00
4 Proteins	2.87	3.38	1.00
5 Proteins	4.50	-	1.00
6 Proteins	12.00	5.00	-

TABLE 7. Gold Binding Trigger Statistics, with dev-test detections

type	**Total**	**TP**	**Predicted**	**Recall**	**Precision**	**F1**
Task1 Gold Events	248	60	124	24.2%	48.4%	32.3%
Single Theme	154	41	85	26.7%	48.2%	34.4%
Double Theme	94	19	39	20.2%	48.7%	28.6%
3-5 Themes	0	0	6	-	0.0%	-

TABLE 8. Development set, Binding Events breakdown for Task 1.

this number is even lower, since we cannot expect all these events to have a correct link structure and only need better post-processing heuristics—our less than 50% precision even on "one theme" events clearly suggests this.

All in all we seem to detect too few binding events to begin with, and hence any heuristic would have little chance to make a difference.

8.2. Consistency of Data

Another reason why our model detects only few binding events may be inconsistencies in the data. For example, we noticed that in some cases there were several potential event clues for one event, and the annotators had to choose one. These choices may have been somewhat random, and are hence difficult to detect by a probabilistic model. An example in the development set illustrates this:

The KBF1/p50 factor **binds** as a **homodimer** ...

Both "binds" and "homodimer" are potential event clues for a binding event. Here we detected "binds" to be the event clue while the annotators chose "homodimer". While there are only two such cases in the development set where this results in a wrong detection, we expect more ambiguous event clues in the six times larger training set.[9]

Another case of inconsistency was observed with regard to the "* Equiv" relation which indicates when two protein mentions in the same sentence in fact refer to the same protein.[10] For example, in

... the binding to the human mineralocorticoid receptor (hMR) ...

the "*Equiv" relation says that "Mineralocorticoid Receptor" and "MR" are really describing the same entity. However, in some cases this relation is missing and this leads to (a) false positives at test time that are indeed correct, and (b) confusion at training time.

Even if the total number of such inconsistencies may not be that large, we still believe that they have a significant impact on our performance, in particular due to the online learning algorithm we apply. To illustrate this, let us consider a training set where some significant fraction of events appear in a context that would suggest "binds" as the event clue. While most of these events are indeed associated with "binds" as clue, some are not (just as in our example above). During online learning

[9]Note that we are detecting only a small number of binding events, and so one could expect ambiguous cases such as the one in the given example be missed completely.

[10]This information was included in the training, development and test sets.

with MIRA we will therefore observe many instances that will increase the weight for "binds" as a binding event clue. However, when we encounter cases where "binds" is not an event clue, we will reduce the corresponding weight, regardless of how often we have seen "binds" as event clue before. In other words, our learner has no increased confidence in "binds" as clue even though it has seen it frequently as such. At test time the "binds" weight will therefore have a lower weight than it should have and we may miss to extract obvious binding events—this could explain the low number of detected binding events mentioned in the previous section.

Note that confidence-weighted learning (Dredze *et al.*, 2008) may solve the above problem. Here each weight is associated with a confidence value that depends on how often it was updated before. In this scheme the weight for "binds" as clue would have a high confidence and we would update it less aggressively when we encounter "binds" as non-clue.

Note that while our binding scores are comparatively low, even the best results do not surpass 45% F1-score. This suggests that binding events are generally hard to extract. One may argue that this is a consequence of the fact that binding events may have multiple Theme arguments. However, we count that only one percent (five events) of the binding events in the training data have more than two Theme participants; in the development test-set, **no** events have more than two Theme arguments. This means that in terms of the number of protein participants, binding extraction for the BioNLP 2009 shared task is very similar the traditional Protein-Protein Interaction (PPI) task where pairs of proteins have to be linked. And here we usually observe higher F1-scores than 45%. This indicates that the number of binding arguments can only be part of the problem.

8.3. Possible Remedies

We are pursuing several directions to improve binding extraction with our model. First, we are working on a principled approach to constructing binding events from link structure. Instead of heuristically deciding which arguments of a binding event clue may or may not be part of the same event, we are planning to model this decision directly within our probabilistic model. Markov Logic would make this easy: we can simply introduce another predicate, inSameBinding(i,j), that holds for the indices i and j of two protein mentions if and only if they are part of the same binding event. We can add formulae that incorporate the syntactic relations between them (e.g., are they coordinated?), as well as other information that may help to disambiguate cases where the syntactic information is insufficient. Note that this formulation would also allow the incorporation of traditional PPI features that are defined over pairs of proteins, as opposed to protein-event-clue word pairs.

As stated above, we think that inconsistencies in the data also lead to a significant amount of errors, and to the low number of binding events we detect in general. We plan to investigate this issue further by gathering more information about the number and type of inconsistencies that arise in the data. We hope that this will help to produce a revised version of the current corpus. Moreover, we seek to explore alternative learning algorithms such as Confidence-weighted learning (Dredze *et al.*, 2008) that yield a model more robust to this kind of inconsistencies.

Finally, we also believe that binding extraction, or general event extraction, should be pursued on a per-document basis. In this BioNLP shared task, we are required to extract **every** mention of each binding event in an abstract. However, from a biologist or database curator point of view, it is often enough to find just one event from each set of identical events within an abstract; the other mentions do not add more information to the global knowledge/database. This suggests that we should not primarily seek to improve per-trigger performance, but rather a per-document metric.[11] Moreover, if we extend our model with document-wide formulae (e.g., if protein x is mentioned to bind in one sentence, it is more likely to be mentioned to bind in another sentence of the same document), we may also be able to improve our per-event-clue results. Obviously, this will create

[11] In this sense we may already do better than our current number suggest. For example, in the abstract with PMID 9164841, there are 12 binding events with "Mineralocorticoid Receptor" or "MR" as the single theme. Our system currently only recognize 3 of these events (25% recall according to the official scoring), but this is still enough in order to say that "MR binding" is described in this document (e.g. 100% recall when considering all "MR binding" are describing the same event).

more difficult inference problems, and it remains to be seen whether the potential improvements are large enough to justify the effort to be made.

9. RELATED WORK

Graphical models have been applied to biomedical Information Extraction before. For example, Ray and Craven (2001) apply linear chain Hidden Markov Networks to the task of relation extraction in biomedical domains. However, while such chains[12] are good at modelling the correlations between labels of neighbouring tokens or phrases in a sentence, they cannot capture global constraints that hold across decisions at different positions within the sentence.[13] Bunescu and Mooney (2004) extract protein names from text and apply Relational Markov Networks to exploit correlation that hold between mentions across sentence boundaries. Here we only look at sentence level global inference, but tackle the more complex task of event extraction.

Much recent work in Information Extraction, and NLP in general, has focused on probabilistic models that relax the independence assumption of local classifiers, linear chain models or pipeline approaches. For example, Poon and Domingos (2007) and Singh *et al.* (2009) perform joint citation segmentation and entity resolution. Finkel *et al.* (2005) use Markov Networks and Gibbs Sampling to incorporate correlations between NER extractions in different sentences of the same document. For Semantic Role Labelling Toutanova *et al.* (2005) use global features across multiple semantic arguments of a predicate. However, to our knowledge none of these have tackled biomedical event extraction.

10. CONCLUSION

In this article we have presented our approach to the BioNLP Shared Task. It can be characterised 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 detection problem between tokens of a sentence.

Our results are competitive: we would reach the 3rd place for task 1 (in close range to the second place) and the 1st place for task 2 (with a 14% margin). Furthermore, we were able to achieve these results with a moderate amount of engineering (less than 3 person months), because inference and learning for our joint model were already provided in the Markov Logic platform we use. In particular, we were able to tackle task 2 by essentially copying the local formulae for event detection, and adding three global formulae (formula 4, 10 and 11 in Table 2). Finally, our system was fast to train (3 hours) on a recent model of a laptop. This simplified the design and testing of good sets of formulae.

We have also shown that by including global formulae we were able to significantly improve performance, in terms of event clue detection, site detection and argument extraction. While this may also be possible with reranking architectures (Toutanova *et al.*, 2005), we believe that in terms of implementation efforts our approach is at least as simple. In fact, the most time-consuming effort of this work was the conversion to and from link detection, 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.

This article also presented an extension to our model that takes into account events with different types that share the same clues. Again, this extension was easy to implement, requiring only a few additional formulae, and lead to a 1% point increase in F1-score.

Moreover, we contribute the investigation of a loss function that is based on the weighted sum of false positives and negatives ground atoms. It was shown that by heavily penalising false negatives at training we achieve much higher recall at test time. However, recall was also shown to saturate at about 40% even though false negatives were maximally penalised.

[12] and their discriminative CRF counterparts.

[13] For example, the fact that every event has to have at least one "Theme" argument cannot be enforced by considering only pairs of subsequent labels.

Finally, we analysed binding events and identified reasons that can lead to poor binding extraction performance. First, we showed that our post-processing heuristic for binding events is based on the wrong assumption of one binding event per event clue. Then we argued that inconsistencies in the data, in combination with our online learning regime, may have led to a lack of confidence when detecting binding events. Finally, we presented directions for future work that may alleviate our problem with binding events.

Acknowledgements

The authors 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). It was also supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

Bilgic, M., Namata, G. M., and Getoor, L. (2007). Combining collective classification and link prediction. In *ICDMW '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 381–386, Washington, DC, USA. IEEE Computer Society.

Björne, J., Heimonen, J., Ginter, F., Airola, A., Pahikkala, T., and Salakoski, T. (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.

Bunescu, R. and Mooney, R. J. (2004). Collective information extraction with relational Markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL' 04)*, pages 439–446.

Charniak, E. and Johnson, M. (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, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computional Linguistics*, **33**(4), 493–552.

Cohen, K. B., Verspoor, K., Johnson, H., Roeder, C., Ogren, P., Baumgartner, W., White, E., and Hunter, L. (2009). High-precision biological event extraction with a concept recognizer. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 50–58, Boulder, Colorado. Association for Computational Linguistics.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, **3**, 951–991.

Dredze, M., Crammer, K., and Pereira, F. (2008). Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM New York, NY, USA.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL' 05)*, pages 363–370.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Kilicoglu, H. and Bergler, S. (2009). Syntactic dependency based heuristics for biological event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 119–127, Boulder, Colorado. Association for Computational Linguistics.

Kim, J. D., Ohta, T., and Tsujii, J. (2008). Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, **9**(1).

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Natural Language Processing in Biomedicine*

NAACL 2009 Workshop (BioNLP '09).

Kok, S., Singla, P., Richardson, M., and Domingos, P. (2005). The Alchemy system for statistical relational AI. Technical report, University of Washington.

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

McDonald, R. and Pereira, F. (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.

Meza-Ruiz, I. and Riedel, S. (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)*.

Poon, H. and Domingos, P. (2007). Joint inference in information extraction. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI '07)*, pages 913–918.

Ray, S. and Craven, M. (2001). Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 1273–1279.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, **62**, 107–136.

Riedel, S. (2008a). 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.

Riedel, S. (2008b). markov thebeast - Markov Logic software platform **url**: `http://thebeast.googlecode.com`.

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

Riedel, S., Chun, H.-W., Takagi, T., and Tsujii, J. (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.

Sætre, R., Yoshida, K., Miwa, M., Matsuzaki, T., Kano, Y., and Tsujii, J. (2009). Akanere relation extraction: Protein interaction and normalization in the biocreative ii.5 challenge. In *BioCreative II.5 Workshop 2009 special session — Digital Annotations*, page 33, Madrid, Spain. CNIO.

Singh, S., Schultz, K., and McCallum, A. (2009). Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 414–429.

Singla, P. and Domingos, P. (2008). Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI '08)*, pages 1094–1099.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (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)*.

Taskar, B., Abbeel, P., and Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the 18th Annual Conference on Uncertainty in AI (UAI '02)*, pages 485–492.

Taskar, B., fai Wong, M., Abbeel, P., and Koller, D. (2003). Link prediction in relational data. In *Advances in Neural Information Processing Systems (NIPS '03)*.

Toutanova, K., Haghighi, A., and Manning, C. D. (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.

Van Landeghem, S., Saeys, Y., De Baets, B., and Van de Peer, Y. (2009). Analyzing text in search of bio-molecular events: a high-precision machine learning framework. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*, pages 128–136, Boulder, Colorado. Association for Computational Linguistics.