Relieving the Computational Bottleneck: Joint Inference for Event Extraction with High-Dimensional Features

Deepak Venugopal and **Chen Chen** and **Vibhav Gogate** and **Vincent Ng**
Department of Computer Science and Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
dxv021000@utdallas.edu, {yzcchen,vgogate,vince}@hlt.utdallas.edu

Abstract

Several state-of-the-art event extraction systems employ models based on Support Vector Machines (SVMs) in a pipeline architecture, which fails to exploit the joint dependencies that typically exist among events and arguments. While there have been attempts to overcome this limitation using Markov Logic Networks (MLNs), it remains challenging to perform joint inference in MLNs when the model encodes many high-dimensional sophisticated features such as those essential for event extraction. In this paper, we propose a new model for event extraction that combines the power of MLNs and SVMs, dwarfing their limitations. The key idea is to reliably learn and process high-dimensional features using SVMs; encode the output of SVMs as low-dimensional, soft formulas in MLNs; and use the superior joint inferencing power of MLNs to enforce joint consistency constraints over the soft formulas. We evaluate our approach for the task of extracting biomedical events on the BioNLP 2013, 2011 and 2009 Genia shared task datasets. Our approach yields the best F1 score to date on the BioNLP'13 (53.61) and BioNLP'11 (58.07) datasets and the second-best F1 score to date on the BioNLP'09 dataset (58.16).

1 Introduction

Event extraction is the task of extracting and labeling all instances in a text document that correspond to a pre-defined event type. This task is quite challenging for a multitude of reasons: events are often nested, recursive and have several arguments; there is no clear distinction between arguments and events; etc. For instance, consider the BioNLP Genia event extraction shared task (Nédellec et al., 2013). In this task, participants are asked to extract instances of a pre-defined set of biomedical events from text. An event is identified by a keyword called the *trigger* and can have an arbitrary number of arguments that correspond to pre-defined argument types. The task is complicated by the fact that an event may serve as an argument of another event (nested events). An example of the task is shown in Figure 1. As we can see, event $E13$ takes as arguments two events, $E14$ and $E12$, which in turn has $E11$ as one of its arguments.

A standard method that has been frequently employed to perform this shared task uses a *pipeline* architecture with three steps: (1) detect if a token is a trigger and assign a trigger type label to it; (2) for every detected trigger, determine all its arguments and assign types to each detected argument; and (3) combine the extracted triggers and arguments to obtain events. Though adopted by the top-performing systems such as the highest scoring system on the BioNLP'13 Genia shared task (Kim et al., 2013), this approach is problematic for at least two reasons. First, as is typical in pipeline architectures, errors may propagate from one stage to the next. Second, since each event/argument is identified and assigned a type independently of the others, it fails to capture the relationship between a trigger and its neighboring triggers, an argument and its neighboring arguments, etc.

More recently, researchers have investigated joint inference techniques for event extraction using Markov Logic Networks (MLNs) (e.g., Poon and Domingos (2007), Poon and Vanderwende (2010), Riedel and McCallum (2011a)), a statistical relational model that enables us to model the dependencies between different instances of a data sample. However, it is extremely challenging to make joint inference using MLNs work well in practice (Poon and Domingos, 2007). One reason is that it is generally difficult to model sophisticated linguistic features using MLNs. The diffi-

. . . demonstrated that HOIL-1L interacting protein (HOIP), a ubiquitin ligase that can catalyze the assembly of linear polyubiquitin chains, is recruited to DC40 in a TRAF2-dependent manner following engagement of CD40 . . .

(a) Sentence fragment

ID	Event Type	Trigger	Arguments
$E11$	*Binding*	recruited	*Theme*={HOIL-1L interacting protein,CD40}
$E12$	*Regulation*	dependent	*Theme*=$E11$, *Cause*=TRAF2
$E13$	*+ve Regulation*	following	*Theme*=$E12$, *Cause*=$E14$
$E14$	*Binding*	engagement	*Theme*=CD40

(b) Events

Figure 1: Example of event extraction in the BioNLP Genia task. The table in (b) shows all the events extracted from sentence (a). Note that successful extraction of $E13$ depends on $E12$ and $E14$.

culty stems from the fact that some of these features are extremely high dimensional (e.g., Chen and Ng (2012), Huang and Riloff (2012b), Li et al. (2012), Li et al. (2013b), Li et al. (2013c)), and to reliably learn weights of formulas that encode such features, one would require an enormous number of data samples. Moreover, even the complexity of approximate inference on such models is quite high, often prohibitively so. For example, a trigram can be encoded as an MLN formula, $\texttt{Word}(w_1, p-1) \wedge \texttt{Word}(w_2, p) \wedge \texttt{Word}(w_3, p+1) \Rightarrow \texttt{Type}(p, T)$. For any given position (p), this formula has W^3 groundings, where W is the number of possible words, making it too large for learning/inference. Therefore, current MLN-based systems tend to include a highly simplified model ignoring powerful linguistic features. This is problematic because such features are essential for event extraction.

Our contributions in this paper are two-fold. First, we propose a novel model for biomedical event extraction based on MLNs that addresses the aforementioned limitations by leveraging the power of Support Vector Machines (SVMs) (Vapnik, 1995; Joachims, 1999) to handle high-dimensional features. Specifically, we (1) learn SVM models using rich linguistic features for trigger and argument detection and type labeling; (2) design an MLN composed of *soft formulas* (each of which encodes a soft constraint whose associated weight indicates how important it is to satisfy the constraint) and *hard formulas* (constraints that always need to be satisfied, thus having a weight of ∞) to capture the relational dependencies between triggers and arguments; and (3) encode the SVM output as *prior knowledge* in the MLN in the form of soft formulas, whose weights are computed using the confidence values generated by the SVMs. This formulation naturally allows SVMs and MLNs to complement each other's strengths and weaknesses: learning in a large and sparse feature space is much easier with SVMs than with MLNs, whereas modeling relational dependencies is much easier with MLNs than with SVMs.

Our second contribution concerns making inference with this MLN feasible. Recall that inference involves detecting and assigning the type label to all the triggers and arguments. We show that existing Maximum-a-posteriori (MAP) inference methods, even the most advanced approximate ones (e.g., Selman et al. (1996), Marinescu and Dechter (2009), Sontag and Globerson (2011)), are infeasible on our proposed MLN because of their high memory cost. Consequently, we identify decompositions of the MLN into disconnected components and solve each independently, thereby drastically reducing the memory requirements.

We evaluate our approach on the BioNLP 2009, 2011 and 2013 Genia shared task datasets. On the BioNLP'13 dataset, our model significantly outperforms state-of-the-art pipeline approaches and achieves the best F1 score to date. On the BioNLP'11 and BioNLP'09 datasets, our scores are slightly better and slightly worse respectively than the best reported results. However, they are significantly better than state-of-the-art MLN-based systems.

2 Background

2.1 Related Work

As a core task in information extraction, event extraction has received significant attention in the natural language processing (NLP) community. The development and evaluation of large-scale learning-based event extraction systems was propelled in part by the availability of annotated corpora produced as part of the Message Understanding Conferences (MUCs), the Automatic Content Extraction (ACE) evaluations, and the BioNLP shared

tasks on event extraction. Previous work on event extraction can be broadly divided into two categories, one focusing on the development of features (henceforth *feature-based* approaches) and the other focusing on the development of models (henceforth *model-based* approaches).

Feature-based approaches. Early work on feature-based approaches has primarily focused on designing local sentence-level features such as token and syntactic features (Grishman et al., 2005; Ahn, 2006). Later, it was realized that local features were insufficient to reliably and accurately perform event extraction in complex domains and therefore several researchers proposed using *high-level features*. For instance, Ji and Grishman (2008) used global information from related documents; Gupta and Ji (2009) extracted implicit time information; Patwardhan and Riloff (2009) used broader sentential context; Liao and Grishman (2010; 2011) leveraged document-level cross-event information and topic-based features; and Huang and Riloff (2012b) explored discourse properties.

Model-based approaches. The model-based approaches developed to date have focused on modeling global properties and seldom use rich, high-dimensional features. To capture global event structure properties, McClosky et al. (2011a) proposed a dependency parsing model. To extract event arguments, Li et al. (2013b) proposed an Integer Linear Programming (ILP) model to encode the relationship between event mentions. To overcome the error propagation problem associated with the pipeline architecture, several joint models have been proposed, including those that are based on MLNs (e.g., Poon and Domingos (2007), Riedel et al. (2009), Poon and Vanderwende (2010)), structured perceptrons (e.g., Li et al. (2013c)), and dual decomposition with minimal domain adaptation (e.g., Riedel and McCallum (2011a; 2011b)).

In light of the high annotation cost required by supervised learning-based event extraction systems, several semi-supervised, unsupervised, and rule-based systems have been proposed. For instance, Huang and Riloff (2012a) proposed a bootstrapping method to extract event arguments using only a small amount of annotated data; Lu and Roth (2012) developed a novel unsupervised sequence labeling model; Bui et al. (2013) implemented a rule-based approach to extract biomedical events; and Ritter et al. (2012) used unsupervised learning to extract events from Twitter data.

Our work extends prior work by developing a rich framework that leverages sophisticated feature-based approaches as well as joint inference using MLNs. This combination gives us the best of both worlds because on one hand, it is challenging to model sophisticated linguistic features using MLNs while on the other hand, feature-based approaches employing sophisticated high-dimensional features suffer from error propagation as the model is generally not rich enough for joint inference.

2.2 The Genia Event Extraction Task

The BioNLP Shared Task (BioNLP-ST) series (Kim et al. (2009), Kim et al. (2011a) and Nédellec et al. (2013)) is designed to tackle the problem of extracting structured information from the biomedical literature. The Genia Event Extraction task is arguably the most important of all the tasks proposed in BioNLP-ST and is also the only task organized in all three events in the series.

The 2009 edition of the Genia task (Kim et al., 2009) was conducted on the Genia event corpus (Kim et al., 2008), which only contains abstracts of the articles that represent domain knowledge around NFκB proteins. The 2011 edition (Kim et al., 2011b) augmented the dataset to include full text articles, resulting in two collections, the abstract collection and the full text collection. The 2013 edition (Kim et al., 2013) further augmented the dataset with recent full text articles but removed the abstract collection entirely.

The targeted event types have also changed slightly over the years. Both the 2009 and 2011 editions are concerned with nine fine-grained event sub-types that can be categorized into three main types, namely simple, binding and regulation events. These three main event types can be distinguished by the kinds of arguments they take. A simple event can take exactly one protein as its $Theme$ argument. A binding event can take one or more proteins as its $Theme$ arguments, and is therefore slightly more difficult to extract than a simple event. A regulation event takes exactly one protein or event as its $Theme$ argument and optionally one protein or event as its $Cause$ argument. If a regulation event takes another event as its $Theme$ or $Cause$ argument, it will lead to a nested event. Regulation events are considered the most difficult-to-extract among the three event types owing in part to the presence of an optional $Cause$ argument and their recursive structure. The 2013 edition intro-

duced a new event type, protein-mod, and its three sub-types. Theoretically, a protein-mod event takes exactly one protein as its $Theme$ argument and optionally one protein or event as its $Cause$ argument. In practice, however, it rarely occurs: there are only six protein-mod events having $Cause$ arguments in the training data for the 2013 edition. Consequently, our model makes the simplifying assumption that a protein-mod event can only take one $Theme$ argument, meaning that we are effectively processing protein-mod events in the same way as simple events.

2.3 Markov Logic Networks

Statistical relational learning (SRL) (Getoor and Taskar, 2007) is an emerging field that seeks to unify logic and probability, and since most NLP techniques are grounded either in logic or probability or both, NLP serves as an ideal application domain for SRL. In this paper, we will employ a popular SRL approach called Markov logic networks (MLNs) (Domingos and Lowd, 2009). At a high level, an MLN is a set of weighted first-order logic formulas (f_i, w_i), where w_i is the weight associated with formula f_i. Given a set of constants that model objects in the domain, it defines a Markov network or a log-linear model (Koller and Friedman, 2009) in which we have one node per ground first-order atom and a propositional feature corresponding to each grounding of each first-order formula. The weight of the feature is the weight of the corresponding first-order formula.

Formally, the probability of a world ω, which represents an assignment of values to all ground atoms in the Markov network, is given by:

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N(f_i, \omega)\right)$$

where $N(f_i, \omega)$ is the number of groundings of f_i that evaluate to True in ω and Z is a normalization constant called the partition function.

The key inference tasks over MLNs are computing the partition function (Z) and the most-probable explanation given evidence (the MAP task). Most queries, including those required by event extraction, can be reduced to these inference tasks. Formally, the partition function and the MAP tasks are given by:

$$Z = \sum_\omega \exp\left(\sum_i w_i N(f_i, \omega)\right) \quad (1)$$

$$\arg\max_\omega P(\omega) = \arg\max_\omega \sum_i w_i N(f_i, \omega) \quad (2)$$

3 Pipeline Model

We implement a pipeline event extraction system using SVMs. This pipeline model serves two important functions: (1) providing a baseline for evaluation and (2) producing prior knowledge for the joint model.

Our pipeline model consists of two steps: trigger labeling and argument labeling. In the trigger labeling step, we determine whether a candidate trigger is a true trigger and label each true trigger with its trigger type. Then, in the argument labeling step, we identify the arguments for each true trigger discovered in the trigger labeling step and assign a role to each argument.

We recast each of the two steps as a classification task and employ $\mathbf{SVM}^{multiclass}$ (Tsochantaridis et al., 2004) to train the two classifiers. We describe each step in detail below.

3.1 Trigger Labeling

A preliminary study of the BioNLP'13 training data suggests that 98.7% of the true triggers' head words[1] are either verbs, nouns or adjectives. Therefore, we consider only those words whose part-of-speech tags belong to the above three categories as candidate triggers. To train the trigger classifier, we create one training instance for each candidate trigger in the training data. If the candidate trigger is not a trigger, the class label of the corresponding instance is $None$; otherwise, the label is the type of the trigger. Thus, the number of class labels equals the number of trigger types plus one. Each training instance is represented by the features described in Table 1(a). These features closely mirror those used in state-of-the-art trigger labeling systems such as Miwa et al. (2010b) and Björne and Salakoski (2013).

After training, we apply the resulting trigger classifier to classify the test instances, which are created in the same way as the training instances. If a test instance is predicted as $None$ by the classifier, the corresponding candidate trigger is labeled as a non-trigger; otherwise, the corresponding candidate trigger is posited as a true trigger whose type is the class value assigned by the classifier.

[1] Head words are found using Collins' (1999) rules.

(a) Features for trigger labeling

Token features	The basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the surrounding tokens in a window of two; character bigrams and trigrams of the candidate trigger word; word n-grams (n=1,2,3) of the candidate trigger word and its context words in a window of three; whether the candidate trigger word contains a digit; whether the candidate trigger word contains an upper case letter; whether the candidate trigger word contains a symbol.
Dependency features	The basic dependency path features (see Table 1(c)) computed using the shortest paths from the candidate trigger to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right.
Other features	The distances from the candidate trigger word to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right; the number of protein words in the sentence.

(b) Features for argument labeling

Token features	Word n-grams (n=1,2,3) of (1) the candidate trigger word and its context in a window of three and (2) the candidate argument word and its context in a window of three; the basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the candidate argument word; the trigger type of the candidate trigger word.
Dependency features	The basic dependency features (see Table 1(c)) computed using the shortest path from the candidate trigger word to the candidate argument word.
Other features	The distance between the candidate trigger word and the candidate argument word; the number of proteins between the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger type and the candidate argument word.

(c) Basic token and dependency features

Basic token features	Six features are computed given a token t, including: (a) the lexical string of t, (b) the lemma of t, (c) the stem of t obtained using the Porter stemmer (Porter, 1980), (d) the part-of-speech tag of t, (e) whether t appears as a true trigger in the training data, and (f) whether t is a protein name.
Basic dependency features	Six features are computed given a dependency path p, including: (a) the vertex walk in p, (b) the edge walk in p, (c) the n-grams (n=2,3,4) of the (stemmed) words associated with the vertices in p, (d) the n-grams (n=2,3,4) of the part-of-speech tags of the words associated with the vertices in p, (e) the n-grams (n=2,3,4) of the dependency types associated with the edges in p, and (f) the length of p.

Table 1: Features for trigger labeling and argument labeling.

3.2 Argument Labeling

The argument classifier is trained as follows. Each training instance corresponds to a candidate trigger and one of its candidate arguments.[2] A candidate argument for a candidate trigger ct is either a protein or a candidate trigger that appears in the same sentence as ct. If ct is not a true trigger, the label of the associated instance is set to $None$. On the other hand, if ct is a true trigger, we check whether the candidate argument in the associated instance is indeed one of ct's arguments. If so, the class label of the instance is the argument's role; otherwise, the class label is $None$. The features used for representing each training instance, which are modeled after those used in Miwa et al. (2010b) and Björne and Salakoski (2013), are shown in Table 1(b).

After training, we can apply the resulting classifier to classify the test instances, which are created in the same way as the training instances. If a test instance is assigned the class $None$ by the classifier, the corresponding candidate argument is classified as not an argument of the trigger. Otherwise, the candidate argument is a true argument of the trigger whose role is the class value assigned by the classifier.

[2]Following the definition of the GENIA event extraction task, the protein names are provided as part of the input.

4 Joint Model

In this section, we describe our Markov logic model that encodes the relational dependencies in the shared task and uses the output of the pipeline model as prior knowledge (soft evidence). We begin by describing the structure of our Markov logic model, and then describe the parameter learning and inference algorithms for it.

4.1 MLN Structure

Figure 2 shows our proposed MLN for BioNLP event extraction, which we refer to as `BioMLN`. The MLN contains six predicates.

The *query predicates* in Figure 2(a) are those whose assignments are not given during inference and thus need to be predicted. Predicate `TriggerType`(sid,tid,$ttype$!) is true when the token located in sentence sid at position tid has type $ttype$. Δ_{ttype}, which denotes the set of constants (or objects) that the logical variable $ttype$

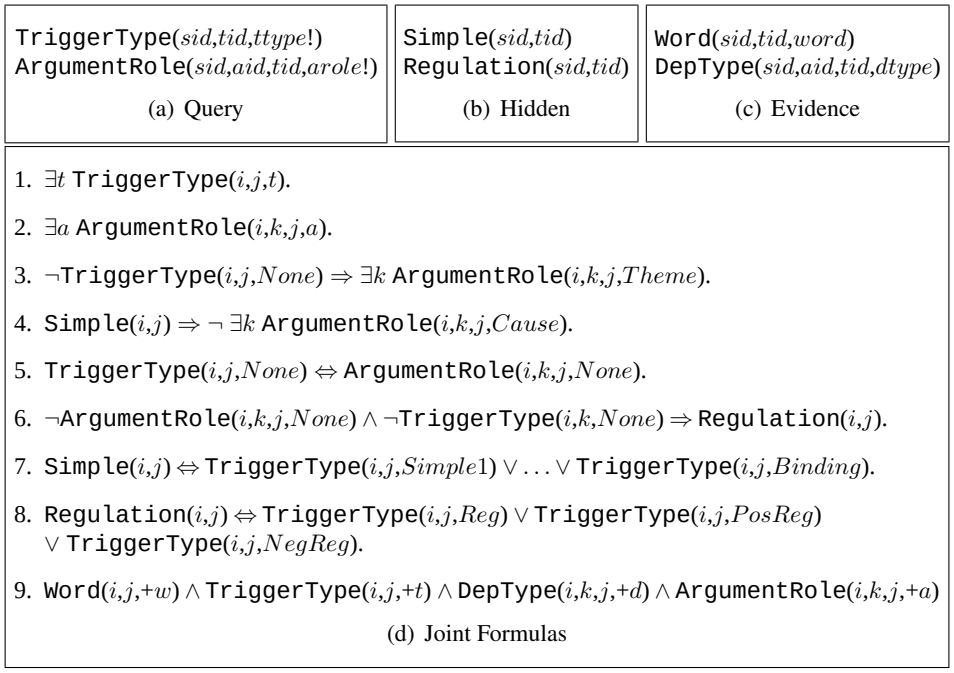

Figure 2: The BioMLN structure.

can be instantiated to, includes all possible trigger types in the dataset plus $None$ (which indicates that the token is not a trigger). The "!" symbol models commonsense knowledge that only one of the types in the domain Δ_{ttype} of $ttype$ is true for every unique combination of sid and tid. Similarly, predicate ArgumentRole(sid,aid,tid,$arole$!) asserts that a token in sentence sid at position aid plays *exactly one* argument role, denoted by $arole$, with respect to the token at position tid. Δ_{arole} includes the two argument types, namely, $Theme$ and $Cause$ plus the additional $None$ that indicates that the token is not an argument.

The hidden predicates in Figure 2(b) are "clusters" of trigger types. Predicate Simple(sid,tid) is true when the token in sentence sid at position tid corresponds to one of the *Simple* event trigger types (BioNLP'13 has 9 simple events, BioNLP'09/'11 have 5) or a binding event trigger type. Similarly, Regulation(sid,tid) asserts that the token in sentence sid at position tid corresponds to any of the three regulation event trigger types.

The *evidence* predicates in Figure 2(c) are those that are always assumed to be known during inference. We define two evidence predicates based on dependency structures. Word(sid,tid,$word$) is true when the word in sentence sid at position tid is equal to $word$. DepType(sid,aid,tid,$dtype$) asserts that $dtype$ is the dependency type in the dependency parse tree that connects the token at position tid to the token at position aid in sentence sid. If the word at tid and the word at aid are directly connected in the dependency tree, then $dtype$ is the label of dependency edge with direction; otherwise $dtype$ is $None$.

The MLN formulas, expressing commonsense, prior knowledge in the domain (Poon and Vanderwende, 2010; Riedel and McCallum, 2011a), are shown in Fig. 2(d). All formulas, except Formula (9), are hard formulas, meaning that they have infinite weights. Note that during weight learning, we only learn the weights of soft formulas.

Formulas (1) and (2) along with the "!" constraint in the predicate definition ensure that the token types are mutually exclusive and exhaustive. Formula (3) asserts that every trigger should have an argument of type $Theme$, since a $Theme$ argument is mandatory for any event. Formula (4) models the constraint that a $Simple$ or $Binding$ trigger has no arguments of type $Cause$ since only regulation events have a $Cause$. Formula (5) asserts that non-triggers have no arguments and vice-versa. Formula (6) models the constraint that if a token is both an argument of t and a trigger by itself, then t must belong to one of the three regulation trigger types. This formula captures the recursive relationship between triggers. Formulas (7) and (8) connect the hidden predicates with the query predicates. Formula (9) is a soft formula encoding

the relationship between triggers and arguments in a dependency parse tree. It joins a word and the dependency type label that connects the word token to the argument token in the dependency parse tree with the trigger types and argument types of the two tokens. The "+" symbol indicates that each grounding of Formula (9) may have a different weight.

4.2 Weight Learning

We can learn BioMLN from data either discriminatively or generatively. Since discriminative learning is much faster than generative learning, we use the former. In discriminative training, we maximize the conditional log-likelihood (CLL) of the query and the hidden variables given an assignment to the evidence variables. In principle, we can use the standard gradient descent algorithm for maximizing the CLL. In each iteration of gradient descent, we update the weights using the following equation (cf. Singla and Domingos (2005) and Domingos and Lowd (2009)):

$$w_j^{t+1} = w_j^t - \alpha(\mathbb{E}_{\mathbf{w}}(n_j) - n_j) \qquad (3)$$

where w_j^t represents the weight of the j^{th} formula in the t^{th} iteration, n_j is the number of groundings in which the j^{th} formula is satisfied in the training data, $\mathbb{E}_{\mathbf{w}}(n_j)$ is the expected number of groundings in which the j^{th} formula is satisfied given the current weight vector $\mathbf{w}$, and α is the learning rate.

As such, the update rule given in Equation (3) is likely to yield poor accuracy because the number of training examples of some types (e.g., $None$) far outnumber other types. To rectify this ill-conditioning problem (Singla and Domingos, 2005; Lowd and Domingos, 2007), we divide the gradient with the number of true groundings in the data, namely, we compute the gradient using $\frac{(\mathbb{E}_{\mathbf{w}}(n_j)-n_j)}{n_j}$.

Another key issue with using Equation (3) is that computing $\mathbb{E}_{\mathbf{w}}(n_j)$ requires performing inference over the MLN. This step is intractable, #P-complete in the worst case. To circumvent this problem and for fast, scalable training, we instead propose to use the voted perceptron algorithm (Collins, 2002; Singla and Domingos, 2005). This algorithm approximates $\mathbb{E}_{\mathbf{w}}(n_j)$ by counting the number of satisfied groundings of each formula in the MAP assignment. Computing the MAP assignment is much easier (although still NP-hard in the worst case) than computing $\mathbb{E}_{\mathbf{w}}(n_j)$, and as a result the voted perceptron algorithm is more scalable than the standard gradient descent algorithm. In addition, it converges much faster.

4.3 Testing

In the testing phase, we combine BioMLN with the output of the pipeline model (see Section 3) to obtain a new MLN, which we refer to as BioMLN$^+$. For every candidate trigger, the SVM trigger classifier outputs a vector of signed confidence values (which is proportional to the distance from the separating hyperplane) of dimension Δ_{ttype} with one entry for each trigger type. Similarly, for every candidate argument, the SVM argument classifier outputs a vector of signed confidence values of dimension Δ_{arole} with one entry for each argument role. In BioMLN$^+$, we model the SVM output as soft evidence, using two soft unit clauses, TriggerType(i,+j,+t) and ArgumentRole(i,+k,+j,+a). We use the confidence values to determine the weights of these clauses. Intuitively, higher (smaller) the confidence, higher (smaller) the weight.

Specifically, the weights of the soft unit clauses are set as follows. If the SVM trigger classifier determines that the trigger in sentence i at position j belongs to type t with confidence $C_{i,j}$, then we attach a weight of $\frac{C_{i,j}}{\alpha n_i}$ to the clause TriggerType(i,j,t). Here, n_i denotes the number of trigger candidates in sentence i. Similarly, if the SVM argument classifier determines that the token at position k in sentence i belongs to the argument role a with respect to the token at position j, with confidence $C'_{i,k,j}$, then we attach a weight of $\frac{C'_{i,k,j}}{\beta \sum_{j=1}^{n_i} m_{ij}}$ to the clause ArgumentRole(i, k, j,a). Here, m_{ij} denotes the number of argument candidates for the j^{th} trigger candidate in sentence i. α and β act as *scale parameters* for the confidence values ensuring that the weights don't get too large (or too small).

4.4 Inference

As we need to perform MAP inference, both at training time and at test time, in this subsection we will describe how to do it efficiently by exploiting unique properties of our proposed BioMLN.

Naively, we can perform MAP inference by grounding BioMLN to a Markov network and then reducing the Markov network by removing from it all (grounded propositional) formulas that are inconsistent with the evidence. On the re-

duced Markov network, we can then compute the MAP solution using standard MAP solvers such as MaxWalkSAT (a state-of-the-art local search based MAP solver) (Selman et al., 1996) and Gurobi[3] (a state-of-the-art, parallelized ILP solver).

The problem with the above approach is that grounding the MLN is infeasible in practice; even the reduced Markov network is just too large. For example, assuming a total of $|\Delta_{sid}|$ sentences and a maximum of N tokens in a sentence, Formula (3) alone has $O(|\Delta_{sid}|N^3)$ groundings. Concretely, at training time, assuming 1000 sentences with 10 tokens per sentence, Formula (3) itself yields one million groundings. Clearly, this approach is not scalable. It turns out, however, that the (ground) Markov network can be decomposed into several disconnected components, each of which can be solved independently. This greatly reduces the memory requirement of the inference step. Specifically, for every grounding of sid, we get a set of nodes in the Markov network that are disconnected from the rest of the Markov network and therefore independent of the rest of the network. Formally,

Proposition 1. *For any world ω of the* `BioMLN`*,*

$$P_{\mathcal{M}}(\omega) = P_{\mathcal{M}_i}(\omega_i)P_{\mathcal{M}\setminus\mathcal{M}_i}(\omega \setminus \omega_i) \quad (4)$$

where ω_i is the world ω projected on the groundings of sentence i and $\mathcal{M}_i$ is `BioMLN` *grounded only using sentence i.*

Using Equation (4), it is easy to see that the MLN $\mathcal{M}$ can be decomposed into $|\Delta_{sid}|$ disjoint MLNs, $\{\mathcal{M}_k\}_{k=1}^{|\Delta_{sid}|}$. The MAP assignment to $\mathcal{M}$ can be computed using, $\bigcup_{i=1}^{|\Delta_{sid}|}\left(\arg\max_{\omega_i} P_{\mathcal{M}_i}(\omega_i)\right)$. This result ensures that to approximate the expected counts $\mathbb{E}_{\mathbf{w}}(n_j)$, it is sufficient to keep exactly *one sentence's groundings in memory*. Specifically, $\mathbb{E}_{\mathbf{w}}(n_j)$ can be written as $\sum_{k=1}^{|\Delta_{sid}|}\mathbb{E}_{\mathbf{w}}(n_j^k)$, where $\mathbb{E}_{\mathbf{w}}(n_j^k)$ indicates the expected number of satisfied groundings of the j^{th} formula in the k^{th} sentence. Since the MAP computation is decomposable, we can estimate $\mathbb{E}_{\mathbf{w}}(n_j^k)$ using MAP inference on just the k^{th} sentence.

5 Evaluation

5.1 Experimental Setup

We evaluate our system on the BioNLP'13 (Kim et al., 2013), '11 (Kim et al., 2011a) and '09 (Kim et al., 2009) Genia datasets for the main event extraction shared task. Note that this task is the most important one for Genia and therefore has the most active participation. Statistics on the datasets are shown in Table 2. All our evaluations use the online tool provided by the shared task organizers. We report scores obtained using the *approximate span, recursive* evaluation.

Dataset	#Papers	#Abstracts	#TT	#Events
BioNLP'13	(10,10,14)	(0,0,0)	13	(2817,3199,3348)
BioNLP'11	(5,5,4)	(800,150,260)	9	(10310,4690,5301)
BioNLP'09	(0,0,0)	(800,150,260)	9	(8597,1809,3182)

Table 2: Statistics on the BioNLP datasets, which consist of annotated papers/abstracts from PubMed. (x, y, z): x in training, y in development and z in test. #TT indicates the total number of trigger types. The total number of argument types is 2.

To generate features, we employ the supporting resources provided by the organizers. Specifically, sentence split and tokenization are done using the GENIA tools, while part-of-speech information is provided by the BLLIP parser that uses the self-trained biomedical model (McClosky, 2010). Also, we create dependency features from the parse trees provided by two dependency parsers, the Enju parser (Miyao and Tsujii, 2008) and the aforementioned BLLIP parser that uses the self-trained biomedical model, which results in two sets of dependency features.

For MAP inference, we use Gurobi, a parallelized ILP solver. After inference, a post-processing step is required to generate biomedical events from the extracted triggers and arguments. Specifically, for binding events, we employ a learning-based method similar to Björne and Salakoski (2011), while for the other events, we employ a rule-based approach similar to Björne et al. (2009). Both the SVM baseline system and the combined MLN+SVM system employ the same post-processing strategy.

During weight learning, in order to combat the problem of different initializations yielding radically different parameter estimates, we start at several different initialization points and average the weights obtained after 100 iterations of gradient descent. However, we noticed that if we simply choose random initialization points, the variance of the weights was quite high and some initialization points were much worse than others. To counter this, we use the following method to systematically

[3] http://www.gurobi.com/

System	Rec.	Prec.	F1
Our System	48.95	59.24	**53.61**
EVEX (Hakala et al., 2013)	45.44	58.03	50.97
TEES-2.1 (Björne and Salakoski, 2013)	46.17	56.32	50.74
BIOSEM (Bui et al., 2013)	42.47	62.83	50.68
NCBI (Liu et al., 2013)	40.53	61.72	48.93
DLUTNLP (Li et al., 2013a)	40.81	57.00	47.56

Table 3: Recall (Rec.), Precision (Prec.) and F1 score on the BioNLP'13 test data.

initialize the weights. Let n_i be the number of satisfied groundings of formula f_i in the training data and m_i be the total number of possible groundings of f_i. We use a threshold γ to determine whether we wish to make the initial weight positive or negative. If $\frac{n_i}{m_i} \leq \gamma$, then we choose the initial weight uniformly at random from the range $[-0.1, 0]$. Otherwise, we chose it from the range $[0, 0.1]$. These steps ensure that the weights generated from different initialization points have smaller variance. Also, in the testing phase, we set the scale parameters for the soft evidence as $\alpha = \beta = \max_{c \in \mathbf{C}} |c|$, where $\mathbf{C}$ is the set of SVM confidence values.

5.2 Results on the BioNLP'13 Dataset

Among the three datasets, the BioNLP'13 dataset is most "realistic" one because it is the only one that contains full papers and no abstracts. As a result, it is also the most challenging dataset among the three. Table 3 shows the results of our system along with the results of other top systems published in the official evaluation of BioNLP'13. Our system achieves the best F1-score (an improvement of 2.64 points over the top-performing system) and has a much higher recall (mainly because our system detects more regulation events which outnumber other event types in the dataset) and a slightly higher precision than the winning system. Of the top five teams, NCBI is the only other joint inference system, which adopts joint pattern matching to predict triggers and arguments at the same time. These results illustrate the challenge in using joint inference effectively. NCBI performed much worse than the SVM-based pipeline systems, EVEX and TEES2.1. It was also worse than BIOSEM, a rule-based system that uses considerable domain expertise. Nevertheless, it was better than DLUTNLP, another SVM-based system.

Figure 3 compares our baseline pipeline model with our combined model. We can clearly see that the combined model has a significantly better F1 score than the pipeline model on most event types.

System	Rec.	Prec.	F1
Our System	53.42	63.61	**58.07**
Miwa12 (Miwa et al., 2012)	53.35	63.48	57.98
Riedel11 (Riedel et al., 2011)	–	–	56
UTurku (Björne and Salakoski, 2011)	49.56	57.65	53.30
MSR-NLP (Quirk et al., 2011)	48.64	54.71	51.50

Table 4: Results on the BioNLP'11 test data.

The regulation events are considered the most complex events to detect because they have a recursive structure. At the same time, this structure yields a large number of joint dependencies. The advantage of using a rich model such as MLNs can be clearly seen in this case; the combined model yields a 10 point and 6 point increase in F1-score on the test data and development data respectively compared to the pipeline model.

5.3 Results on the BioNLP'11 Dataset

Table 4 shows the results on the BioNLP'11 dataset. We can see that our system is marginally better than Miwa12, which is a pipeline-based system. It is also more than two points better than Riedel11, a state-of-the-art structured prediction-based joint inference system. Reidel11 incorporates the Stanford predictions (McClosky et al., 2011b) as features in the model. On the two hardest, most complex tasks, detecting regulation events (which have recursive structures and more joint dependencies than other event types) and detecting binding events (which may have multiple arguments), our system performs better than both Miwa12 and Riedel11.[4] Specifically, our system's F1 score for regulation events is 46.84, while those of Miwa12 and Riedel11 are 45.46 and 44.94 respectively. Our system's F1 score for the binding event is 58.79, while those of Miwa12 and Riedel11 are 56.64 and 48.49 respectively. These results clearly demonstrate the effectiveness of enforcing joint dependencies along with high-dimensional features.

5.4 Results on the BioNLP'09 Dataset

Table 5 shows the results on the BioNLP'09 dataset. Our system has a marginally lower score (by 0.11 points) than Miwa12, which is the best performing system on this dataset. Specifically, our system achieves a higher recall but a lower precision than Miwa12. However, note that Miwa12 used co-reference features while we are able to achieve

[4]Detailed results are not shown for any of these three datasets due to space limitations.

	SVM			MLN+SVM		
Type	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
Simple	64.47	87.89	74.38	73.11	78.99	75.94
Protein-Mod	66.49	79.87	72.57	72.25	69.70	70.95
Binding	39.04	50.00	43.84	48.05	43.84	45.85
Regulation	23.51	56.21	33.15	36.47	50.86	42.48
Overall	37.90	67.88	**48.64**	48.95	59.24	**53.61**

(a) Test

	SVM			MLN+SVM		
Type	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
Simple	55.79	81.63	66.28	63.21	75.10	68.64
Protein-Mod	64.47	87.89	74.38	71.14	85.63	77.72
Binding	31.90	48.77	38.57	47.99	50.00	48.97
Regulation	20.13	52.46	29.10	28.57	43.41	34.46
Overall	34.42	66.14	**45.28**	43.50	57.45	**49.51**

(b) Development

Figure 3: Comparison of the combined model (MLN+SVM) with the pipeline model on the BioNLP'13 test and development data.

System	**Rec.**	**Prec.**	**F1**
Miwa12 (Miwa et al., 2012)	52.67	65.19	58.27
Our System	53.96	63.08	**58.16**
Riedel11 (Riedel et al., 2011)	–	–	57.4
Miwa10 (Miwa et al., 2010a)	50.13	64.16	56.28
Bjorne (Björne et al., 2009)	46.73	58.48	51.95
PoonMLN (Poon&Vanderwende,2010)	43.7	58.6	50.0
RiedelMLN (Riedel et al., 2009)	36.9	55.6	44.4

Table 5: Results on the BioNLP'09 test data. "–" indicates that the corresponding values are not known.

similar accuracy without the use of co-reference data. The F1 score of Miwa10, which does not use co-reference features, is nearly 2 points lower than that of our system. Our system also has a higher F1 score than Reidel11, which is the best joint inference-based system for this task.

On the regulation events, our system (47.55) outperforms both Miwa12 (45.99) and Riedel11 (46.9), while on the binding event, our system (59.88) is marginally worse than Miwa12 (59.91) and significantly better than Riedel11 (52.6). As mentioned earlier, these are the hardest events to extract. Also, existing MLN-based joint inference systems such as RiedelMLN and PoonMLN do not achieve state-of-the-art results because they do not leverage complex, high-dimensional features.

6 Summary and Future Work

Markov logic networks (MLNs) are a powerful representation that can compactly encode rich relational structures and ambiguities (uncertainty). As a result, they are an ideal representation for complex NLP tasks that require joint inference, such as event extraction. Unfortunately, the superior representational power greatly complicates inference and learning over MLN models. Even the most advanced methods for inference and learning in MLNs (Gogate and Domingos, 2011) are unable to handle complex, high-dimensional features, and therefore existing MLN systems primarily use low-dimensional features. This limitation severely affects the accuracy of MLN-based NLP systems, and as a result, in some cases their performance is inferior to pipeline methods that do not employ joint inference.

In this paper, we presented a general approach for exploiting the power of high-dimensional linguistic features in MLNs. Our approach involves reliably processing and learning high-dimensional features using SVMs and encoding their output as low-dimensional features in MLNs. We showed that we could achieve scalable learning and inference in our proposed MLN model by exploiting decomposition. Our results on the BioNLP shared tasks from '13, '11, and '09 clearly show that our proposed combination is extremely effective, achieving the best or second best score on all three datasets.

In future work, we plan to (1) improve our joint model by incorporating co-reference information and developing model ensembles; (2) transfer the results of this investigation to other complex NLP tasks that can potentially benefit from joint inference; and (3) develop scalable inference and learning algorithms (Ahmadi et al., 2013).

Acknowledgments

This work was supported in part by the AFRL under contract number FA8750-14-C-0021, by the ARO MURI grant W911NF-08-1-0242, and by the DARPA Probabilistic Programming for Advanced-Machine Learning Program under AFRL prime contract number FA8750-14-C-0005. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of DARPA, AFRL, ARO or the US government.

References

Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan. 2013. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning*, 92(1):91–132.

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*, pages 1–8.

Jari Björne and Tapio Salakoski. 2011. Generalizing biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 183–191.

Jari Björne and Tapio Salakoski. 2013. TEES 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25.

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 10–18.

Quoc-Chinh Bui, David Campos, Erik van Mulligen, and Jan Kors. 2013. A fast rule-based approach for biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 104–108.

Chen Chen and Vincent Ng. 2012. Joint modeling for Chinese event extraction with rich linguistic features. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 529–544.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8.

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

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

Vibhav Gogate and Pedro Domingos. 2011. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 256–265.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 system description. In *Proceedings of the ACE 2005 Evaluation Workshop*. Washington.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 369–372.

Kai Hakala, Sofie Van Landeghem, Tapio Salakoski, Yves Van de Peer, and Filip Ginter. 2013. EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 26–34.

Ruihong Huang and Ellen Riloff. 2012a. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295.

Ruihong Huang and Ellen Riloff. 2012b. Modeling textual cohesion for event extraction. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 254–262.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10.

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

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011a. Overview of BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 1–6.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011b. Overview of Genia event task in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 7–15.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The Genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in Chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1006–1016.

Lishuang Li, Yiwen Wang, and Degen Huang. 2013a. Improving feature-based biomedical event extraction system by integrating argument information. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 109–115.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013b. Argument inference from relevant event mentions in Chinese argument extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1477–1487.

Qi Li, Heng Ji, and Liang Huang. 2013c. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in preselected and balanced collections. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 9–16.

Haibin Liu, Karin Verspoor, Donald C. Comeau, Andrew MacKinlay, and W John Wilbur. 2013. Generalizing an approximate subgraph matching-based system to extract events in molecular biology and cancer genetics. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 76–85.

Daniel Lowd and Pedro Domingos. 2007. Efficient weight learning for markov logic networks. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 835–844.

Radu Marinescu and Rina Dechter. 2009. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491.

David McClosky, Mihai Surdeanu, and Chris Manning. 2011a. Event extraction as dependency parsing. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies*, pages 1626–1635.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011b. Event extraction as dependency parsing for BioNLP 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 41–45.

David McClosky. 2010. *Any domain parsing: Automatic domain adaptation for natural language parsing*. Ph.D. thesis, Ph.D. thesis, Brown University, Providence, RI.

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

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010b. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology*, 8(01):131–146.

Makoto Miwa, Paul Thompson, and Sophia Ananiadou. 2012. Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics*, 28(13):1759–1765.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160.

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 913–918.

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

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Chris Quirk, Pallavi Choudhury, Michael Gamon, and Lucy Vanderwende. 2011. MSR-NLP entry in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 155–163.

Sebastian Riedel and Andrew McCallum. 2011a. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12.

Sebastian Riedel and Andrew McCallum. 2011b. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 46–50.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A Markov logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 41–49.

Sebastian Riedel, David McClosky, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. 2011. Model combination for event extraction in bionlp 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 51–55.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1104–1112.

Bart Selman, Henry Kautz, and Bram Cohen. 1996. Local Search Strategies for Satisfiability Testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pages 521–532. American Mathematical Society, Washington, DC.

Parag Singla and Pedro Domingos. 2005. Discriminative training of Markov logic networks. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 868–873.

David Sontag and Amir Globerson. 2011. Introduction to Dual Decomposition for Inference. *Optimization for Machine Learning*.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, pages 104–112.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York, NY.