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

Deepak Venugopal Chen Chen Vibhav Gogate Vincent Ng
Department of Computer Science
University of Texas at Dallas

Event Extraction

- Event extraction is the task of extracting and labeling all instances in a text document that correspond to pre-defined event types

Event Extraction

- Event extraction is the task of extracting and labeling all instances in a text document that correspond to pre-defined event types
  - BioNLP Genia event extraction task concerns the extraction of instances of bio-molecular event types (Kim et al., 2009)

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

An Example of BioNLP Events

… demonstrated that HOIL-1L interaction protein (HOIP) is recruited to CD40 in a TRAF2-dependent manner …

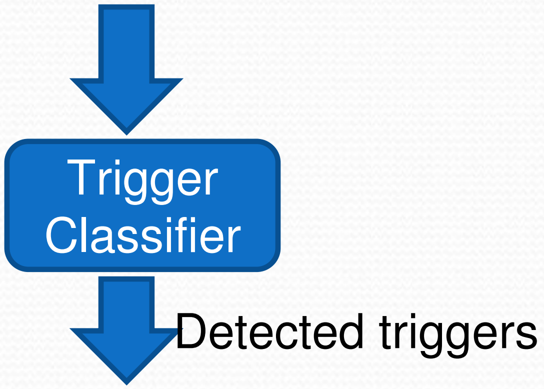
ID	Event Type	Trigger	Arguments
E11	Binding	recruited	Theme={HOIL-1L interaction protein, CD40}
E12	Regulation	dependent	Theme=E11, Cause=TRAF2

Events can be nested.

Standard Pipeline Approach

Standard Pipeline Approach

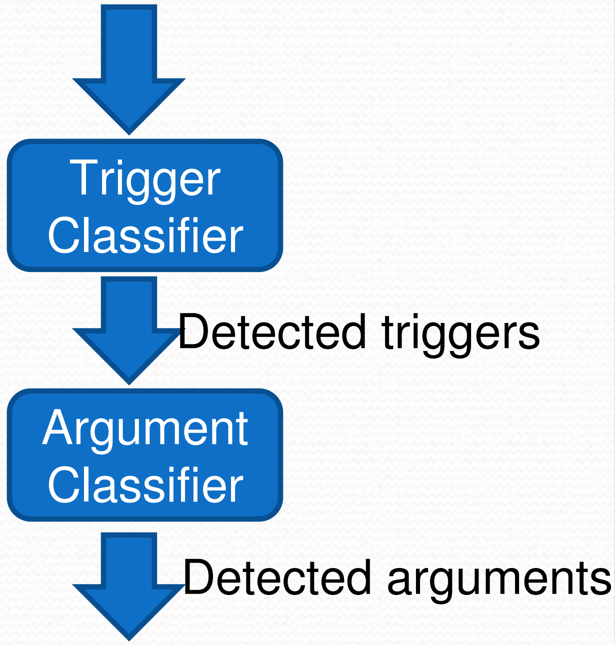
Step 1: Detect if a token is a trigger and if so, assign a event/trigger type to it



Standard Pipeline Approach

Step 1: Detect if a token is a trigger and if so, assign a event/trigger type to it

Step 2: For every detected trigger, determine all its arguments and assign a role to each detected argument

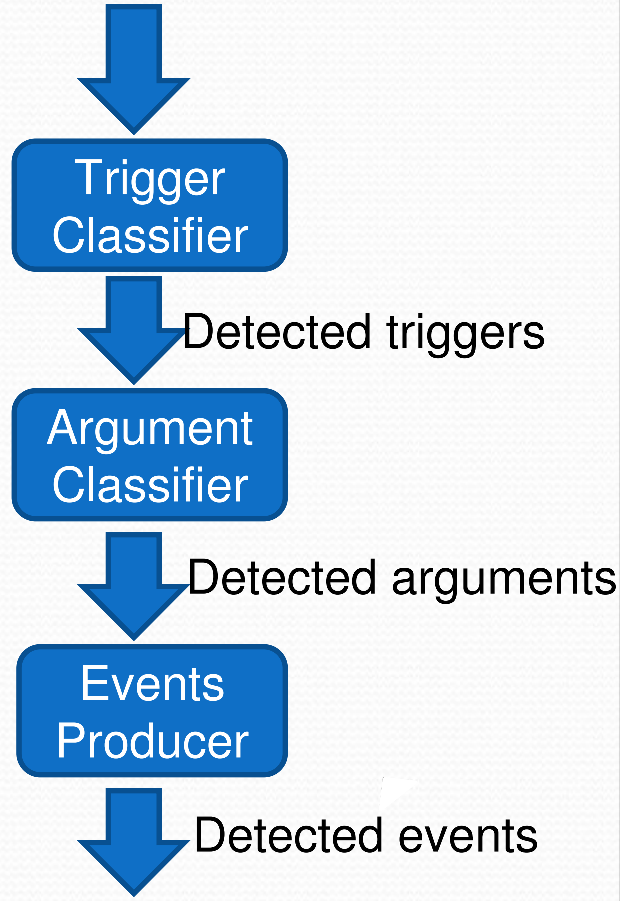


Standard Pipeline Approach

Step 1: Detect if a token is a trigger and if so, assign a event/trigger type to it

Step 2: For every detected trigger, determine all its arguments and assign a role to each detected argument

Step 3: Combine the extracted triggers and arguments to produce events

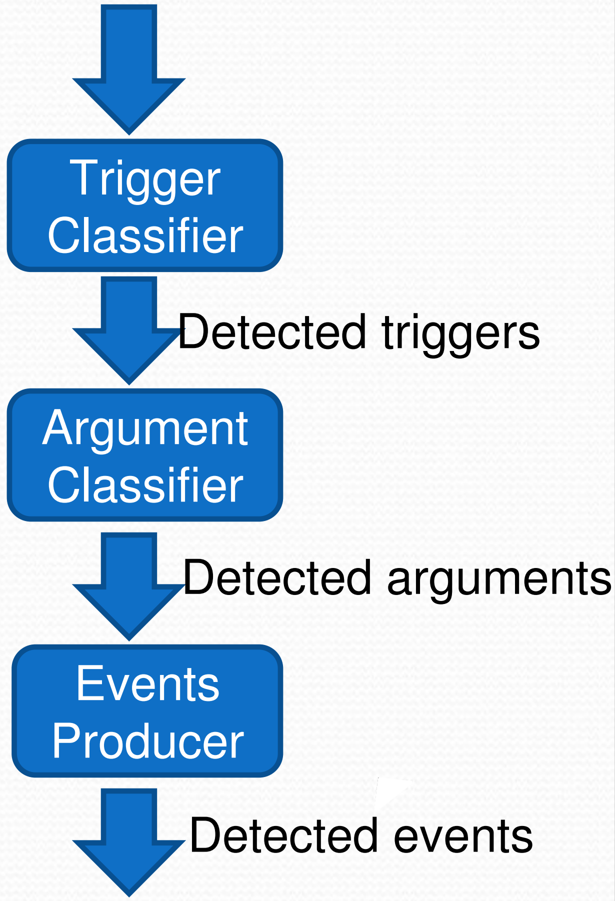


Standard Pipeline Approach

Step 1: Detect if a token is a trigger and if so, assign a event/trigger type to it

Step 2: For every detected trigger, determine all its arguments and assign a role to each detected argument

Step 3: Combine the extracted triggers and arguments to produce events



Steps 1 and 2 are difficult, while Step 3 is trivial

Pipeline Approach: Pros and Cons

- **Pros**
  - Approach is simple and straightforward
  - Uses an efficient learner (e.g., SVMs) in each step, thus enabling the use of high-dimensional features
    - Features such as n-grams of context words, n-grams of words/POS/dependency relations extracted from dependency paths are important for event extraction

Pipeline Approach: Pros and Cons

- **Pros**
  - Approach is simple and straightforward
  - Uses an efficient learner (e.g., SVMs) in each step, thus enabling the use of high-dimensional features
    - Features such as n-grams of context words, n-grams of words/POS/dependency relations extracted from dependency paths are important for event extraction
- **Cons**
  - Error may propagate from one stage to the next
  - Each trigger/argument is detected independently, thus failing to capture the relationships between neighboring triggers, neighboring arguments, etc.

Pipeline Approach

- achieved state-of-the-art results in BioNLP Genia event extraction despite its weaknesses
  - BioNLP’13: Hakala et al.(2013)
  - BioNLP’09 and BioNLP’11: Miwa et al. (2012)

Pipeline Approach

- achieved state-of-the-art results in BioNLP Genia event extraction despite its weaknesses
  - BioNLP'13: Hakala et al.(2013)
  - BioNLP'09 and BioNLP'11: Miwa et al. (2012)

Can we improve further? If so, how?

Pipeline Approach

- achieved state-of-the-art results in BioNLP Genia event extraction despite its weaknesses
  - BioNLP’13: Hakala et al.(2013)
  - BioNLP’09 and BioNLP’11: Miwa et al. (2012)

Can we improve further? If so, how?

Try to overcome the weaknesses of the pipeline approach

Joint Inference

- **Markov Logic Networks (MLNs)**
  - Riedel et al. (2009), Poon and Vanderwende (2010)

- **Pros**
  - can avoid error propagation
    - by jointly detecting triggers and arguments
  - can model dependencies between triggers/arguments

MLNs for Event Extraction

- The performance of existing MLN approaches to event extraction lags behind that of state-of-the-art pipeline approaches

MLNs for Event Extraction

- The performance of existing MLN approaches to event extraction lags behind that of state-of-the-art pipeline approaches
  - Because they include a highly simplified model ignoring powerful high-dimensional features

MLNs for Event Extraction

- The performance of existing MLN approaches to event extraction lags behind that of state-of-the-art pipeline approaches
  - Because they include a highly simplified model ignoring powerful high-dimensional features
  - Modeling high-dimensional features using MLNs is difficult

MLNs for Event Extraction

- The performance of existing MLN approaches to event extraction lags behind that of state-of-the-art pipeline approaches
  - Because they include a highly simplified model ignoring powerful high-dimensional features
  - Modeling high-dimensional features using MLNs is difficult
  - The complexity of inference is high

MLNs for Event Extraction

- The performance of existing MLN approaches to event extraction lags behind that of state-of-the-art pipeline approaches
  - Because they include a highly simplified model ignoring powerful high-dimensional features
  - Modeling high-dimensional features using MLNs is difficult
  - The complexity of inference is high
    - E.g.: $Word(w_1,p-1) \wedge Word(w_2,p) \wedge Word(w_3,p+1) \rightarrow Type(p, T)$
      - A simple trigram feature requires $|T|*|W|^3$ groundings for each position p

MLNs for Event Extraction

- The performance of existing MLN approaches to event extraction lags behind that of state-of-the-art pipeline approaches
  - Because they include a highly simplified model ignoring powerful high-dimensional features
  - Modeling high-dimensional features using MLNs is difficult
  - The complexity of inference is high
    - E.g.: $Word(w_1,p-1) \wedge Word(w_2,p) \wedge Word(w_3,p+1) \rightarrow Type(p, T)$
      - A simple trigram feature requires $|T|*|W|^3$ groundings for each position p

assignment of values to variables

Goal

- Combine the strengths of the pipeline approach and MLNs for event extraction

Goal

- Combine the strengths of the pipeline approach and MLNs for event extraction

can handle high-dimensional features

Goal

- Combine the strengths of the pipeline approach and MLNs for event extraction

can handle high-dimensional features

can capture relational dependencies

Goal

- Combine the strengths of the pipeline approach and MLNs for event extraction

can handle high-dimensional features

can capture relational dependencies

Propose a model for event extraction based on **MLNs** that can handle **high-dimensional** features

Plan for the Talk

- Preliminaries
  - The Genia event extraction task
  - Markov Logic Networks
- Baseline system
- Our MLN approach
- Evaluation

Plan for the Talk

- Preliminaries
  - The Genia event extraction task
  - Markov Logic Networks
- Baseline system
- Our MLN approach
- Evaluation

BioNLP Genia Event Extraction Task

- organized in 2009, 2011 and 2013
  - 2009: abstracts only
  - 2011: abstracts and some full-text articles
  - 2013: full-text articles only

BioNLP Genia Event Extraction Task

- organized in 2009, 2011 and 2013
  - 2009: abstracts only
  - 2011: abstracts and some full-text articles
  - 2013: full-text articles only

- concerned with extracting instances of 9 fine-grained event subtypes that can be categorized into 3 main types:
  - **Simple** event
  - **Binding** event
  - **Regulation** event

BioNLP Genia Event Extraction Task

- organized in 2009, 2011 and 2013
  - 2009: abstracts only
  - 2011: abstracts and some full-text articles
  - 2013: full-text articles only

- concerned with extracting instances of 9 fine-grained event subtypes that can be categorized into 3 main types:
  - **Simple** event
    - One protein as its ***THEME*** argument
  - **Binding** event
  - **Regulation** event

BioNLP Genia Event Extraction Task

- organized in 2009, 2011 and 2013
  - 2009: abstracts only
  - 2011: abstracts and some full-text articles
  - 2013: full-text articles only

- concerned with extracting instances of 9 fine-grained event subtypes that can be categorized into 3 main types:
  - **Simple** event
    - One protein as its ***THEME*** argument
  - **Binding** event
    - One or more proteins as its ***THEME*** argument
  - **Regulation** event

BioNLP Genia Event Extraction Task

- organized in 2009, 2011 and 2013
  - 2009: abstracts only
  - 2011: abstracts and some full-text articles
  - 2013: full-text articles only

- concerned with extracting instances of 9 fine-grained event subtypes that can be categorized into 3 main types:
  - **Simple** event
    - One protein as its ***THEME*** argument
  - **Binding** event
    - One or more proteins as its ***THEME*** argument
  - **Regulation** event
    - One protein or event as its ***THEME*** argument and optionally one protein or event as its ***CAUSE*** argument

MLNs: Preliminaries

- 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

MLNs: Preliminaries

- 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
  - A formula specifies a hard or soft **constraint** on predicates

  A weight specifies how important it is to satisfy the constraint
    - **hard formula**: infinite weight
      - Father(x,y) → Child(y,x)
    - **soft formula**: finite weight
      - Mother(x,y) → Below45(x)

MLNs: Preliminaries

- 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
  - A formula specifies a hard or soft **constraint** on predicates

  A weight specifies how important it is to satisfy the constraint
    - **hard formula**: infinite weight
      - Father(x,y) → Child(y,x)
    - **soft formula**: finite weight
      - Mother(x,y) → Below45(x)
  - A **grounding** of a formula is an assignment of values to the variables in the formula
    - In Mother(x,y) → Below45(x), if each of x and y has 10 values, then the formula has 10 x 10 = 100 groundings

MLNs: Preliminaries

- A world $\boldsymbol{\omega}$ is an assignment of values to all ground predicates
  - Father(Bob,John)=T, Father(Bob,Ted)=F, Father(Jack,Matt)=T, Male(Bob)=T, Male(Jesse)=F, …

MLNs: Preliminaries

- A world $\boldsymbol{\omega}$ is an assignment of values to all ground predicates
  - Father(Bob,John)=T, Father(Bob,Ted)=F, Father(Jack,Matt)=T, Male(Bob)=T, Male(Jesse)=F, …

- The probability of a world $\boldsymbol{\omega}$ is given by:

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

MLNs: Preliminaries

- A world $\boldsymbol{\omega}$ is an assignment of values to all ground predicates
  - Father(Bob,John)=T, Father(Bob,Ted)=F, Father(Jack,Matt)=T, Male(Bob)=T, Male(Jesse)=F, …

- The probability of a world $\boldsymbol{\omega}$ is given by:

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

the number of groundings of f_i that evaluates to True in ω

MLNs: Preliminaries

- A world $\boldsymbol{\omega}$ is an assignment of values to all ground predicates
  - Father(Bob,John)=T, Father(Bob,Ted)=F, Father(Jack,Matt)=T, Male(Bob)=T, Male(Jesse)=F, …

- The probability of a world $\boldsymbol{\omega}$ is given by:

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

normalization constant

the number of groundings of f_i that evaluates to True in ω

MLNs: Preliminaries

- A world $\boldsymbol{\omega}$ is an assignment of values to all ground predicates
  - Father(Bob,John)=T, Father(Bob,Ted)=F, Father(Jack,Matt)=T, Male(Bob)=T, Male(Jesse)=F, …

- The probability of a world $\boldsymbol{\omega}$ is given by:

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

ω is more probable if more formulas are satisfied more often

normalization constant

the number of groundings of f_i that evaluates to True in ω

MLNs: Preliminaries

- The key inference task over MLNs is finding the most probable world (a.k.a. the MAP task):

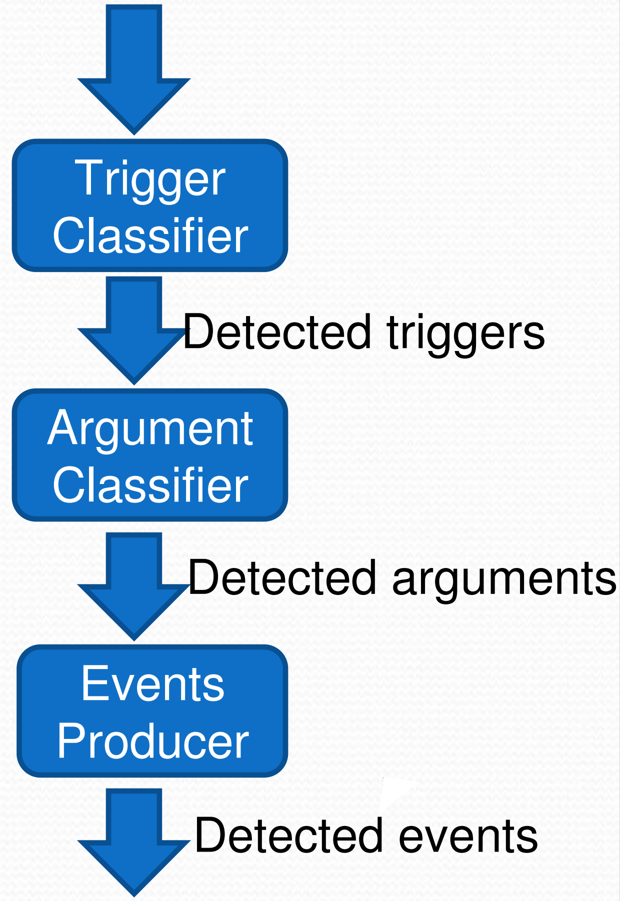
$$\arg\max_{\omega} P(\omega) = \arg\max_{\omega} \sum_{i} w_i N(f_i, \omega)$$

Plan for the Talk

- Preliminaries
  - The Genia event extraction task
  - Markov Logic Networks
- Baseline system
- Our MLN approach
- Evaluation

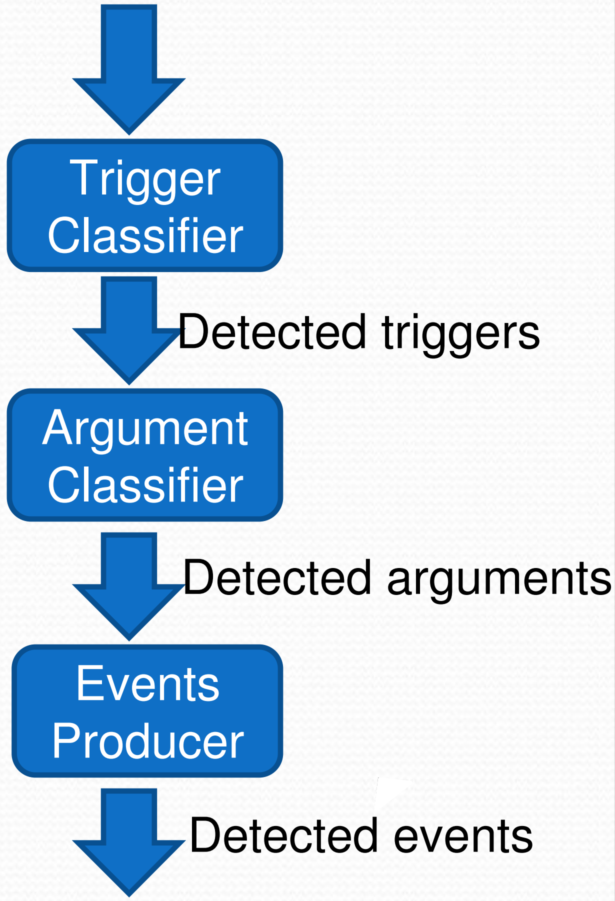
Baseline System

- Adopts the standard pipeline approach
  - trigger classification followed by argument classification



Baseline System

- Adopts the standard pipeline approach
  - trigger classification followed by argument classification



Training the Trigger Classifier

- **Training instance creation**
  - Create one training instance for each **candidate trigger** in each training document
    - Candidate triggers are verbs, nouns and adjectives in the text
  - Class label is
    - Trigger/Event type if the candidate trigger is a true trigger
    - None otherwise

Training the Trigger Classifier

- **Training instance creation**
  - Create one training instance for each **candidate trigger** in each training document
    - Candidate triggers are verbs, nouns and adjectives in the text
  - Class label is
    - Trigger/Event type   if the candidate trigger is a true trigger
    - None   otherwise
- **Learning algorithm**
  - SVM-multiclass

Training the Trigger Classifier

- Features computed on a candidate trigger t

Token features	The lexical string, lemma, stem, POS of t and its surrounding tokens in a window of 2; word n-grams (n=1,2,3) of t and its context words; whether t contains an uppercase letter or a symbol; …
Dependency features	Compute features based on the shortest dependency path p from t to the nearest protein: vertex walk in p; edge walk in p; n-grams (n-2,3,4) of the stemmed words associated with p's vertices; n-grams (n=2,3,4) of the POS tags of the words associated with p's vertices; …

Training the Trigger Classifier

- Features computed on a candidate trigger t

Token features	The lexical string, lemma, stem, POS of t and its surrounding tokens in a window of 2; word n-grams (n=1,2,3) of t and its context words; whether t contains an uppercase letter or a symbol; …
Dependency features	Compute features based on the shortest dependency path p from t to the nearest protein: vertex walk in p; edge walk in p; n-grams (n-2,3,4) of the stemmed words associated with p's vertices; n-grams (n=2,3,4) of the POS tags of the words associated with p's vertices; …

Training the Trigger Classifier

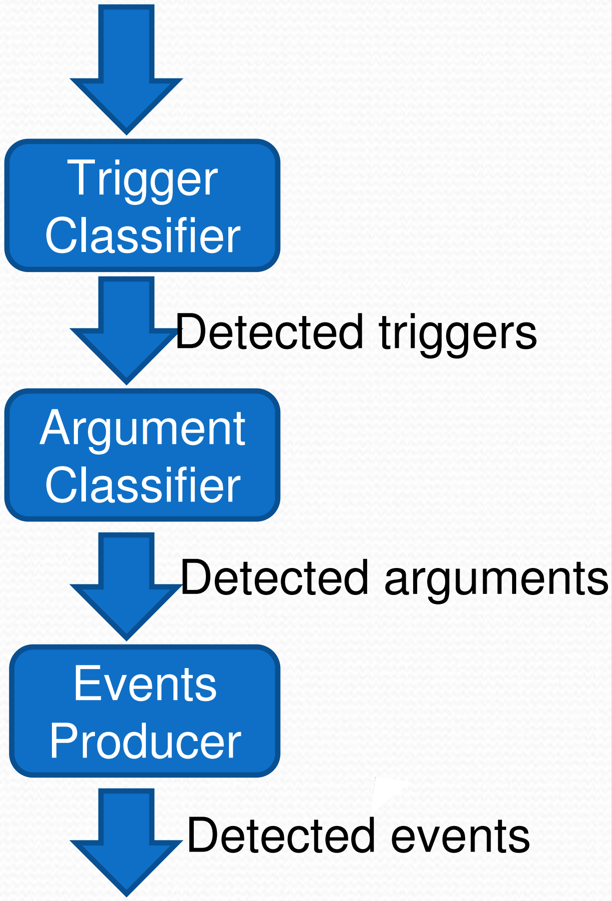
- Features computed on a candidate trigger t

Token features	The lexical string, lemma, stem, POS of t and its surrounding tokens in a window of 2; word n-grams (n=1,2,3) of t and its context words; whether t contains an uppercase letter or a symbol;
Dependency features	Compute features based on the shortest dependency path p from t to the nearest protein: vertex walk in p; edge walk in p; n-grams (n-2,3,4) of the stemmed words associated with p's vertices; n-grams (n=2,3,4) of the POS tags of the words associated with p's vertices; …

Over 3 million features

Baseline System

- Adopts the standard pipeline approach
  - trigger classification followed by argument classification



Training the Argument Classifier

- **Training instance creation**
  - Create one training instance by pairing a candidate trigger *t* with one of its candidate arguments
    - A candidate argument is either a protein or a candidate argument that appears in the same sentence as candidate trigger *t*
  - Class label is
    - Argument role if the candidate argument is a true argument of *t*
    - None otherwise

Training the Argument Classifier

- **Training instance creation**
  - Create one training instance by pairing a candidate trigger *t* with one of its candidate arguments
    - A candidate argument is either a protein or a candidate argument that appears in the same sentence as candidate trigger *t*
  - Class label is
    - Argument role if the candidate argument is a true argument of *t*
    - None otherwise

- **Learning algorithm**
  - SVM-multiclass

Training the Argument Classifier

- Features computed on candidate trigger t and candidate argument a

Token features	Word n-grams (n=1,2,3) of t and its surrounding text; Word n-grams (n=1,2,3) of a and its surrounding context; The lexical string, lemma, stem, POS of t and its surrounding tokens in a window of 2; …
Dependency features	Compute features based on the shortest dependency path p from t to a: n-grams (n-2,3,4) of the stemmed words, POS tags, and dependency types associated with p's vertices; …
Other features	Distance between t and a; Number of proteins between t and a; Concatenation of t and a; Concatenation of t's event type and a

Training the Argument Classifier

- Features computed on candidate trigger t and candidate argument a

Token features	Word n-grams (n=1,2,3) of t and its surrounding text; Word n-grams (n=1,2,3) of a and its surrounding context; The lexical string, lemma, stem, POS of t and its surrounding tokens in a window of 2; …
Dependency features	Compute features based on the shortest dependency path p from t to a: n-grams (n-2,3,4) of the stemmed words, POS tags, and dependency types associated with p's vertices; …
Other features	Distance between t and a; Number of proteins between t and a; Concatenation of t and a; Concatenation of t's event type and a

Plan for the Talk

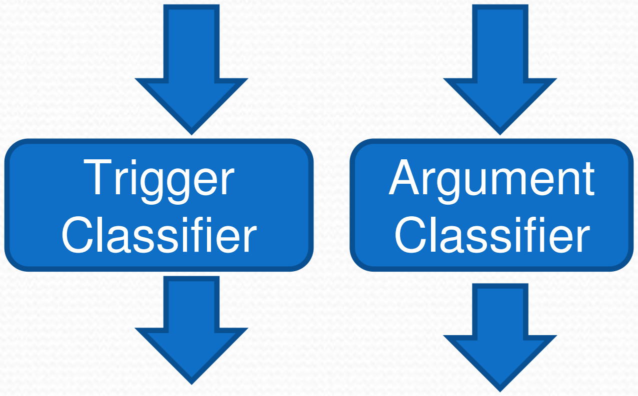
- Preliminaries
  - The Genia event extraction task
  - Markov Logic Networks
- Baseline system
- Our MLN approach
- Evaluation

Our MLN Approach to Event Extraction

- **Goal**: design a model for event extraction that combines the strengths of SVMs and MLNs
  - can employ **high-dimensional** features
  - can model the **dependencies** between different data instances

Our Approach: An Overview

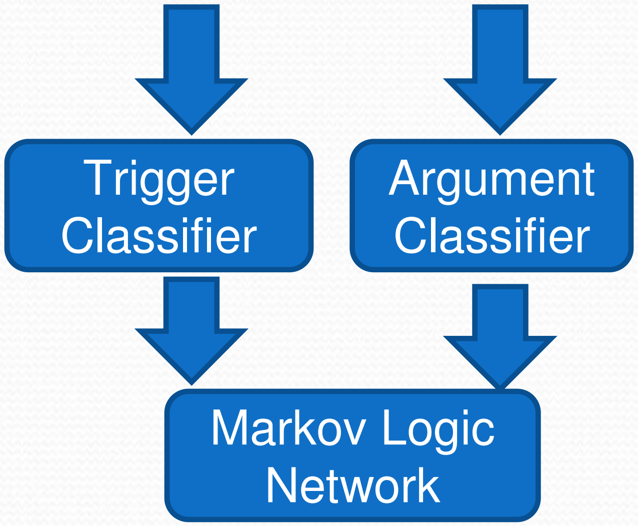
Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**



Our Approach: An Overview

Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**

Step 2: Design an MLN whose formulas encode the soft and hard constraints on the predicates

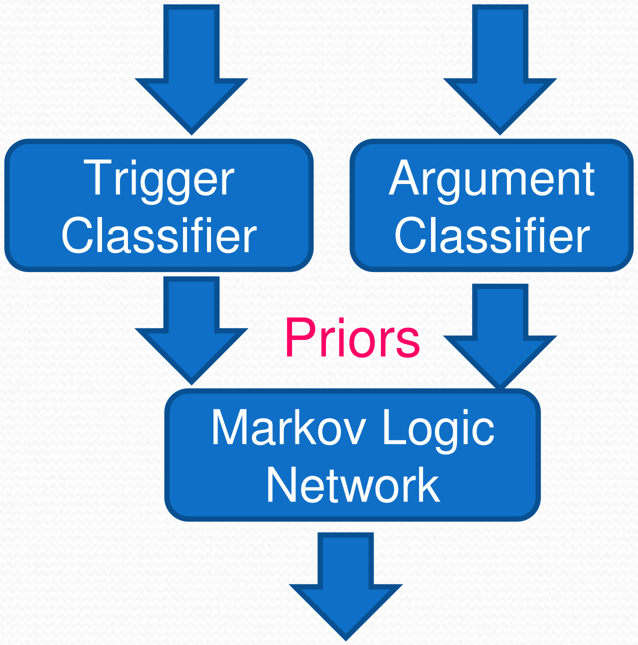


Our Approach: An Overview

Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**

Step 2: Design an MLN whose formulas encode the soft and hard constraints on the predicates

Step 3: Encode SVM output as prior knowledge in the MLNs

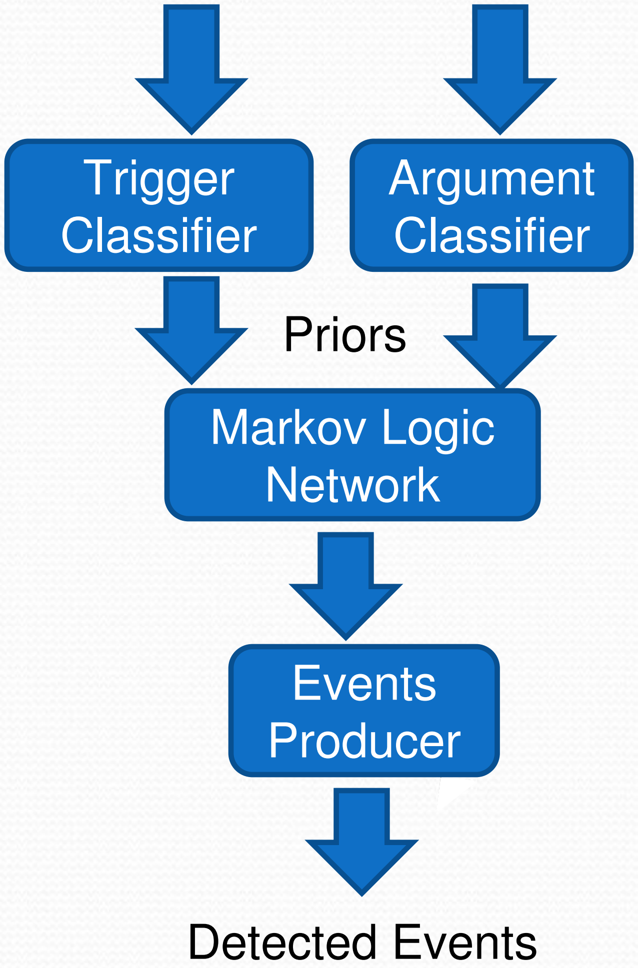


Our Approach: An Overview

Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**

Step 2: Design an MLN whose formulas encode the soft and hard constraints on the predicates

Step 3: Encode SVM output as prior knowledge in the MLNs

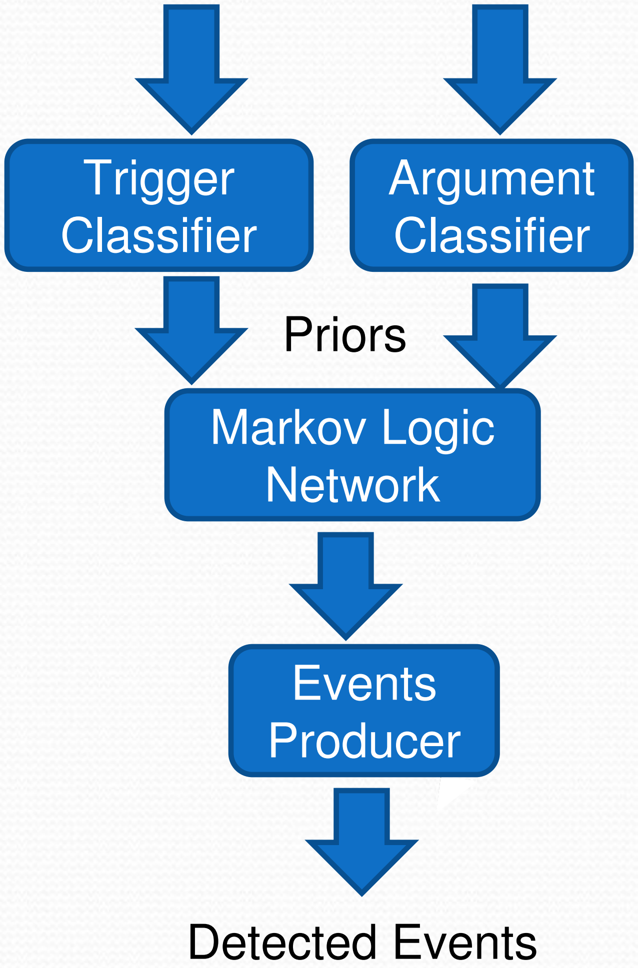


Our Approach: An Overview

Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**

Step 2: Design an MLN whose formulas encode the soft and hard constraints on the predicates

Step 3: Encode SVM output as prior knowledge in the MLNs



BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

$$\texttt{TriggerType}(sid, tid, ttype!)$$
$$\texttt{ArgumentRole}(sid, aid, tid, arole!)$$

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

$\mathtt{TriggerType}(sid,tid,ttype!)$
$\mathtt{ArgumentRole}(sid,aid,tid,arole!)$

The token located in sentence *sid* at position *tid* has trigger type *ttype*

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

$\mathtt{TriggerType}(sid, tid, ttype!)$
$\mathtt{ArgumentRole}(sid, aid, tid, arole!)$

The token located in sentence *sid* at position *tid* has trigger type *ttype*

The token in sentence *sid* at position aid plays the argument role *arole* w.r.t. the token at position *tid*

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known

```
TriggerType(sid,tid,ttype!)
ArgumentRole(sid,aid,tid,arole!)
```

"!" means that exactly one *ttype* makes the predicate true for each combination of *sid* and *tid*

The token located in sentence *sid* at position *tid* has trigger type *ttype*

The token in sentence *sid* at position aid plays the argument role *arole* w.r.t. the token at position *tid*

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

  $\texttt{TriggerType}(sid, tid, ttype!)$
  $\texttt{ArgumentRole}(sid, aid, tid, arole!)$

- **Hidden predicates:** model latent random processes

  $\texttt{Simple}(sid, tid)$
  $\texttt{Regulation}(sid, tid)$

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

  TriggerType(sid,tid,$ttype!$)
  ArgumentRole(sid,aid,tid,$arole!$)

- **Hidden predicates:** model latent random processes

  Simple(sid,tid)
  Regulation(sid,tid)

The token in sentence *sid* at position *tid* corresponds to a **Simple** or **Binding** event

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

  $\texttt{TriggerType}(sid, tid, ttype!)$

  $\texttt{ArgumentRole}(sid, aid, tid, arole!)$

- **Hidden predicates:** model latent random processes

  $\texttt{Simple}(sid, tid)$

  $\texttt{Regulation}(sid, tid)$

The token in sentence *sid* at position *tid* corresponds to a **Simple** or **Binding** event trigger

The token in sentence *sid* at position *tid* corresponds to a **Regulation** event trigger

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

  $\mathtt{TriggerType}(sid,tid,ttype!)$
  $\mathtt{ArgumentRole}(sid,aid,tid,arole!)$

- **Hidden predicates:** model latent random processes

  $\mathtt{Simple}(sid,tid)$
  $\mathtt{Regulation}(sid,tid)$

- **Evidence predicates:** assumed to be known during inference

  $\mathtt{Word}(sid,tid,word)$
  $\mathtt{DepType}(sid,aid,tid,dtype)$

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

  $\texttt{TriggerType}(sid, tid, ttype!)$
  $\texttt{ArgumentRole}(sid, aid, tid, arole!)$

- **Hidden predicates:** model latent random processes

  $\texttt{Simple}(sid, tid)$
  $\texttt{Regulation}(sid, tid)$

- **Evidence predicates:** assumed to be known during inference

  $\texttt{Word}(sid, tid, word)$
  $\texttt{DepType}(sid, aid, tid, dtype)$

The word in sentence ***sid*** at position ***tid*** is equal to word

BioMLN: 3 kinds of Predicates

- **Query predicates:** assignments not known & need to be predicted

  TriggerType($sid,tid,ttype!$)
  ArgumentRole($sid,aid,tid,arole!$)

- **Hidden predicates:** model latent random processes

  Simple(sid,tid)
  Regulation(sid,tid)

- **Evidence predicates:** assumed to be known during inference

  Word($sid,tid,word$)
  DepType($sid,aid,tid,dtype$)

The word in sentence ***sid*** at position ***tid*** is equal to word

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***

BioMLN: 9 Formulas

1. $\exists t$ $\texttt{TriggerType}(i,j,t)$.

2. $\exists a$ $\texttt{ArgumentRole}(i,k,j,a)$.

3. $\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k$ $\texttt{ArgumentRole}(i,k,j,Theme)$.

4. $\texttt{Simple}(i,j) \Rightarrow \neg \exists k$ $\texttt{ArgumentRole}(i,k,j,Cause)$.

5. $\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None)$.

6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.

7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.

8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$
   $\vee\ \texttt{TriggerType}(i,j,NegReg)$.

9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

BioMLN: 9 Formulas

1. $\exists t$ $\texttt{TriggerType}(i,j,t)$.
2. $\exists a$ $\texttt{ArgumentRole}(i,k,j,a)$.
3. $\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k$ $\texttt{ArgumentRole}(i,k,j,Theme)$.
4. $\texttt{Simple}(i,j) \Rightarrow \neg$ $\exists k$ $\texttt{ArgumentRole}(i,k,j,Cause)$.
5. $\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None)$.
6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.
7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.
8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$
   $\vee$ $\texttt{TriggerType}(i,j,NegReg)$.
9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

Each candidate trigger/argument has a type/role

BioMLN: 9 Formulas

1. $\exists t$ $\texttt{TriggerType}(i,j,t)$.
2. $\exists a$ $\texttt{ArgumentRole}(i,k,j,a)$.
3. $\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k$ $\texttt{ArgumentRole}(i,k,j,Theme)$.
4. $\texttt{Simple}(i,j) \Rightarrow \neg\ \exists k$ $\texttt{ArgumentRole}(i,k,j,Cause)$.
5. $\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None)$.
6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.
7. $\boxed{\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding).}$
8. $\boxed{\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg) \vee \texttt{TriggerType}(i,j,NegReg).}$
9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

Hidden predicates are clusters of trigger types

BioMLN: 9 Formulas

1. $\exists t\ \texttt{TriggerType}(i,j,t)$.
2. $\exists a\ \texttt{ArgumentRole}(i,k,j,a)$.
3. $\boxed{\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k\ \texttt{ArgumentRole}(i,k,j,Theme).}$
4. $\texttt{Simple}(i,j) \Rightarrow \neg\ \exists k\ \texttt{ArgumentRole}(i,k,j,Cause)$.
5. $\boxed{\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None).}$
6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.
7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.
8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$
   $\vee\ \texttt{TriggerType}(i,j,NegReg)$.
9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

Capture dependencies between triggers and arguments

BioMLN: 9 Formulas

1. $\exists t$ $\texttt{TriggerType}(i,j,t)$.
2. $\exists a$ $\texttt{ArgumentRole}(i,k,j,a)$.
3. $\boxed{\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k\ \texttt{ArgumentRole}(i,k,j,Theme).}$
4. $\boxed{\texttt{Simple}(i,j) \Rightarrow \neg\ \exists k\ \texttt{ArgumentRole}(i,k,j,Cause).}$
5. $\boxed{\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None).}$
6. $\boxed{\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j).}$
7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.
8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$ $\vee\ \texttt{TriggerType}(i,j,NegReg)$.
9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

Capture dependencies between triggers and arguments

MLN Structure – Joint Formulas

Hard constraints:
infinite weights

1. $\exists t$ $\texttt{TriggerType}(i,j,t)$.
2. $\exists a$ $\texttt{ArgumentRole}(i,k,j,a)$.
3. $\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k$ $\texttt{ArgumentRole}(i,k,j,Theme)$.
4. $\texttt{Simple}(i,j) \Rightarrow \neg \exists k$ $\texttt{ArgumentRole}(i,k,j,Cause)$.
5. $\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None)$.
6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.
7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.
8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$ $\vee\ \texttt{TriggerType}(i,j,NegReg)$.
9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

MLN Structure – Joint Formulas

1. $\exists t\ \texttt{TriggerType}(i,j,t)$.
2. $\exists a\ \texttt{ArgumentRole}(i,k,j,a)$.
3. $\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k\ \texttt{ArgumentRole}(i,k,j,Theme)$.
4. $\texttt{Simple}(i,j) \Rightarrow \neg\ \exists k\ \texttt{ArgumentRole}(i,k,j,Cause)$.
5. $\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None)$.
6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.
7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.
8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$ $\vee\ \texttt{TriggerType}(i,j,NegReg)$.
9. $\boxed{\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)}$

Encode how a word w with trigger type *t* is related to the role *a* of its argument via dependency type *d*

MLN Structure – Joint Formulas

1. $\exists t$ $\texttt{TriggerType}(i,j,t)$.
2. $\exists a$ $\texttt{ArgumentRole}(i,k,j,a)$.
3. $\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k$ $\texttt{ArgumentRole}(i,k,j,Theme)$.
4. $\texttt{Simple}(i,j) \Rightarrow \neg\, \exists k$ $\texttt{ArgumentRole}(i,k,j,Cause)$.
5. $\texttt{TriggerType}(i,j,None) \Leftrightarrow \texttt{ArgumentRole}(i,k,j,None)$.
6. $\neg\texttt{ArgumentRole}(i,k,j,None) \wedge \neg\texttt{TriggerType}(i,k,None) \Rightarrow \texttt{Regulation}(i,j)$.
7. $\texttt{Simple}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Simple1) \vee \ldots \vee \texttt{TriggerType}(i,j,Binding)$.
8. $\texttt{Regulation}(i,j) \Leftrightarrow \texttt{TriggerType}(i,j,Reg) \vee \texttt{TriggerType}(i,j,PosReg)$ $\vee\, \texttt{TriggerType}(i,j,NegReg)$.
9. $\texttt{Word}(i,j,+w) \wedge \texttt{TriggerType}(i,j,+t) \wedge \texttt{DepType}(i,k,j,+d) \wedge \texttt{ArgumentRole}(i,k,j,+a)$

Encode how a word w with trigger type *t* is related to the role *a* of its argument via dependency type *d*

Soft constraints: weights need to be learned

Weight Learning for Soft Formulas

- Could use gradient descent to maximize the conditional log-likelihood of the query and the hidden variables given an assignment to the evidence variables

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

Weight Learning for Soft Formulas

- Could use gradient descent to maximize the conditional log-likelihood of the query and the hidden variables given an assignment to the evidence variables

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

number of groundings in which the j-th formula is satisfied in the training data

Weight Learning for Soft Formulas

- Could use gradient descent to maximize the conditional log-likelihood of the query and the hidden variables given an assignment to the evidence variables

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

expected number of groundings in which the j-th formula is satisfied given the current weight vector

number of groundings in which the j-th formula is satisfied in the training data

Weight Learning for Soft Formulas

- Could use gradient descent to maximize the conditional log-likelihood of the query and the hidden variables given an assignment to the evidence variables

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

Computing the expectation requires performing inference over the MLN and is intractable

number of groundings in which the j-th formula is satisfied in the training data

Weight Learning for Soft Formulas

- Could use gradient descent to maximize the conditional log-likelihood of the query and the hidden variables given an assignment to the evidence variables

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

Computing the expectation requires performing inference over the MLN and is intractable

number of groundings in which the j-th formula is satisfied in the training data

Use the **voted perceptron algorithm**:
Approximate the number of satisfied groundings in the MAP assignment

Weight Learning for Soft Formulas

- Could use gradient descent to maximize the conditional log-likelihood of the query and the hidden variables given an assignment to the evidence variables

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

Computing the expectation requires performing inference over the MLN and is intractable

number of groundings in which the j-th formula is satisfied in the training data

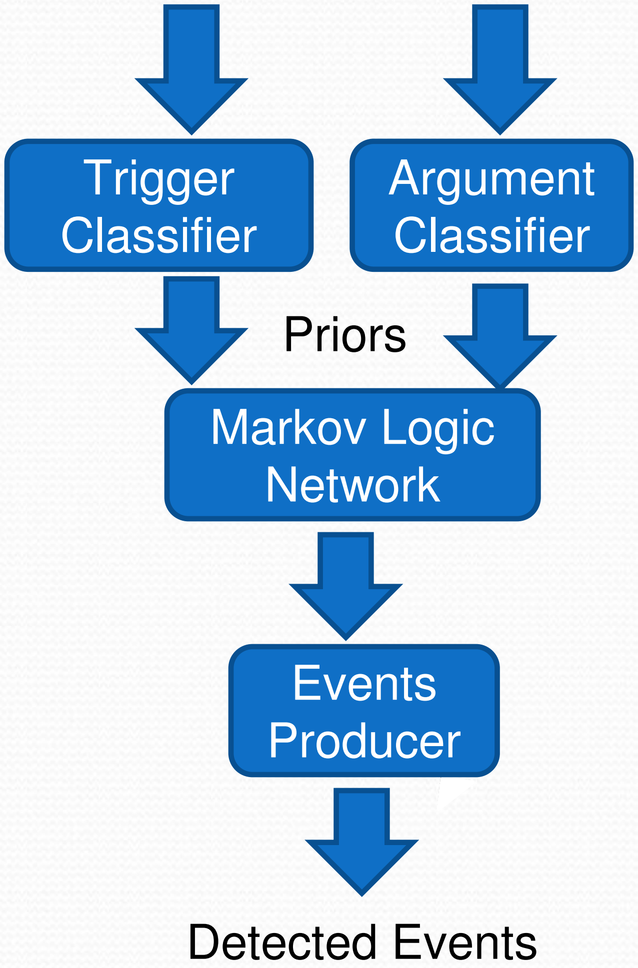
Use the **voted perceptron algorithm**:
Approximate the number of satisfied groundings in the MAP assignment
Easier to compute the MAP assignment than the expectation

Our Approach: An Overview

Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**

Step 2: Design an MLN whose formulas encode the soft and hard constraints on the predicates

Step 3: Encode SVM output as prior knowledge in the MLNs



Using the SVM Output as Priors for the MLN

- Two soft clauses are added into the MLN:

$\texttt{TriggerType}(i,+j,+t)$ $\texttt{ArgumentRole}(i,+k,+j,+a)$

'+' means a separate weight is to be learned for each unique combination of j and t

Using the SVM Output as Priors for the MLN

- Two soft clauses are added into the MLN:

$\texttt{TriggerType}(i,+j,+t)$ $\texttt{ArgumentRole}(i,+k,+j,+a)$

The SVM trigger classifier provides confidence regarding how likely the word at position *j* in sentence *i* has trigger type *t*

Using the SVM Output as Priors for the MLN

- Two soft clauses are added into the MLN:

$\texttt{TriggerType}(i,+j,+t)$ $\texttt{ArgumentRole}(i,+k,+j,+a)$

The SVM trigger classifier provides confidence regarding how likely the word at position *j* in sentence *i* has trigger type *t*

The SVM argument classifier provides confidence regarding how likely the word at position *k* in sentence *i* has argument role a w.r.t. the token at position *j*

Using the SVM Output as Priors for the MLN

- Two soft clauses are added into the MLN:

$\texttt{TriggerType}(i,+j,+t)$ $\texttt{ArgumentRole}(i,+k,+j,+a)$

The SVM trigger classifier provides confidence regarding how likely the word at position *j* in sentence *i* has trigger type *t*

The SVM argument classifier provides confidence regarding how likely the word at position *k* in sentence *i* has argument role a w.r.t. the token at position *j*

- Use these confidence values as the weights of these soft formulas

Using the SVM Output as Priors for the MLN

- Two soft clauses are added into the MLN:

  $\texttt{TriggerType}(i,+j,+t)$ $\texttt{ArgumentRole}(i,+k,+j,+a)$

  The SVM trigger classifier provides confidence regarding how likely the word at position *j* in sentence *i* has trigger type *t*

  The SVM argument classifier provides confidence regarding how likely the word at position *k* in sentence *i* has argument role a w.r.t. the token at position *j*

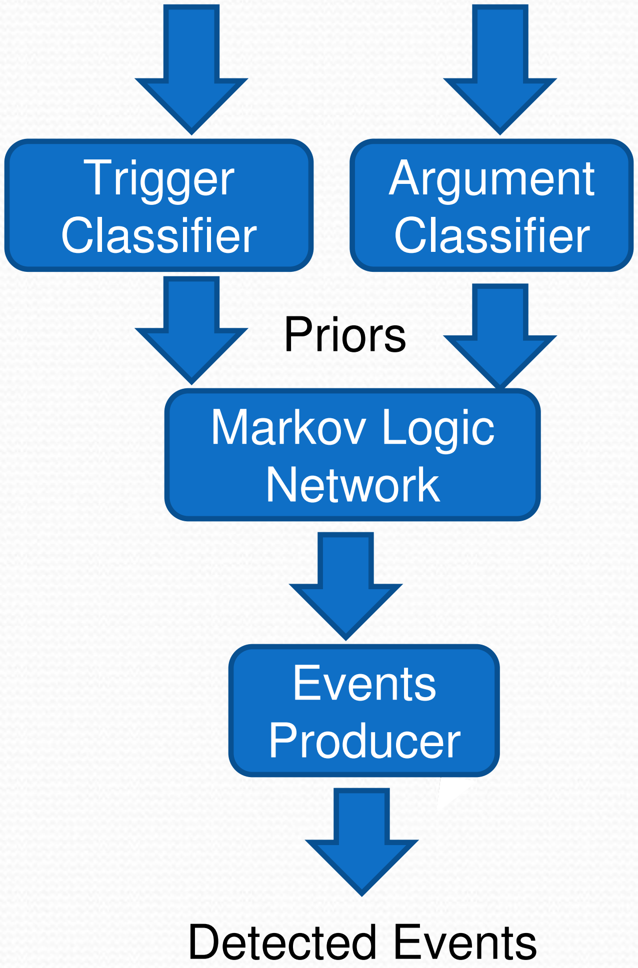
- Use these confidence values as the weights of these soft formulas
  - Provide prior knowledge for the MLN
    - High-dimensional features implicitly used by the MLN
  - In practice, we need to **scale** these confidence values

Our Approach: An Overview

Step 1: Learn the SVM trigger and argument classifiers using high-dimensional features as in the **Baseline**

Step 2: Design an MLN whose formulas encode the soft and hard constraints on the predicates

Step 3: Encode SVM output as prior knowledge in the MLNs



What's next?

What's next?

- Perform MAP inference
  - Needed not only in testing but also in training (weight learning)

What's next?

- Perform MAP inference
  - Needed not only in testing but also in training (weight learning)

How?

Naïve MAP Inference

1. Ground the whole MLN and then reduce it by removing formulas that are inconsistent with the evidence

2. Compute the MAP solution using standard solvers

Naïve MAP Inference

1. Ground the whole MLN and then reduce it by removing formulas that are inconsistent with the evidence

2. Compute the MAP solution using standard solvers

- But.. this naive approach is infeasible because of the huge size of the network:
  - Assuming 1000 sentences and 10 tokens per sentence, 100K groundings will be generated for one formula:

$$\neg\texttt{TriggerType}(i,j,None) \Rightarrow \exists k\ \texttt{ArgumentRole}(i,k,j,Theme).$$

Efficient MAP Inference

- **Idea**
  - Decompose the network into several disconnected components

Efficient MAP Inference

- **Idea**
  - Decompose the network into several disconnected components

- **Observation**
  - All our predicates are sentence-dependent

Efficient MAP Inference

- **Idea**
  - Decompose the network into several disconnected components
- **Observation**
  - All our predicates are sentence-dependent, so
    - the MAP assignment to the MLN over **all** sentences is the same as the **union** of the MAP assignments to the MLN over **each** sentence

Efficient MAP Inference

- **Idea**
  - Decompose the network into several disconnected components
- **Observation**
  - All our predicates are sentence-dependent, so
    - the MAP assignment to the MLN over **all** sentences is the same as the **union** of the MAP assignments to the MLN over **each** sentence
    - The MAP computation is decomposable
      - can perform MAP inference for each sentence independently

Efficient MAP Inference

- **Idea**
  - Decompose the network into several disconnected components
- **Observation**
  - All our predicates are sentence-dependent, so
    - the MAP assignment to the MLN over **all** sentences is the same as the **union** of the MAP assignments to the MLN over **each** sentence
    - The MAP computation is decomposable
      - can perform MAP inference for each sentence independently
- So.. we can create one MLN per sentence
  - keep only one sentence's grounding in memory

Plan for the Talk

- Preliminaries
  - The Genia event extraction task
  - Markov Logic Networks
- Baseline system
- Our MLN approach
- Evaluation

Evaluation: Goal

- Evaluate our MLN approach to event extraction

Evaluation: Datasets and Statistics

Dataset	#Papers	#Abstracts	#Trigger types	#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)

- (x,y,z): x in training, y in development, and z in test

Evaluation: Scoring Setting

- Scores obtained by submitting our output to the official online evaluation tool under the **approximate span**, **recursive evaluation** setting

Results on the BioNLP'13 Test Data

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

- Best systems on this dataset:
  - EVEX, TEES-2.1 and DLUTNLP: pipeline learning systems
  - BIOSEM: a rule based system
  - NCBI: the only joint model using subgraph isomorphism
- Our system beats the best-performing system

Comparison with Baseline on BioNLP'13

	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**

Comparison with Baseline on BioNLP'13

	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**

- MLN+SVM beats our SVM baseline by nearly 5 points
  - Joint inference using MLN is important
  - Performance on **Regulation** events improves by 9.33 points

Results on the BioNLP'11 Test Data

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

- Best systems on this dataset:
  - Miwa12: a pipeline system using coreference features
  - Riedel11: a joint model using dual decomposition
  - Uturku and MSR-NLP: pipeline systems

Results on the BioNLP'11 Test Data

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

- Best systems on this dataset:
  - Miwa12: a pipeline system using coreference features
  - Riedel11: a joint model using dual decomposition
  - Uturku and MSR-NLP: pipeline systems
- Even without using coreference features, our system
  - performs marginally better than Miwa12
  - beats the best joint model and other two best pipeline models

Results on the BioNLP'09 Test 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

- Best systems on this dataset:
  - Miwa12: a pipeline system with a coreference feature
  - Riedel11: a joint model using dual decomposition
  - Miwa10 and Bjorne: pipeline systems without coreference features
  - PoonMLN and RiedelMLN: MLN-based systems

Results on the BioNLP'09 Test 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

- Best systems on this dataset:
  - Miwa12: a pipeline system with a coreference feature
  - Riedel11: a joint model using dual decomposition
  - Miwa10 and Bjorne: pipeline systems without coreference features
  - PoonMLN and RiedelMLN: MLN-based systems

Results on the BioNLP'09 Test 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

- Our system
  - outperforms previous MLN-based systems, the best joint model, and pipeline systems without coreference features
  - is only marginally worse than Miwa12, which uses coreference features

Summary

- Presented a general approach for exploiting the power of high-dimensional features in MLNs
- Obtained the best or second best score on the three Genia event extraction datasets