Ferrara, August 29th 2018

Applications of Statistical Relational AI

Advanced Course in Artificial Intelligence (ACAI 2018)

Marco Lippi
marco.lippi@unimore.it

Hands-On Lecture

Goal of the lecture

Use some StaRAI frameworks to build models, perform learning and inference, upon some classic applications, such as entity classification and link prediction.

Software

- Alchemy (Markov Logic Networks)
- ProbLog (lecture by Prof. Luc De Raedt)
- cplint (lecture by Prof. Fabrizio Riguzzi)

Hands-On Lecture

Also demos running on browsers (fewer features)

- http://pracmln.open-ease.org/
- https://dtai.cs.kuleuven.be/problog/editor.html
- http://cplint.eu/

StaRAI Problems

StaRAI applications typically have to deal with **three** distinct, but strongly **inter-related** problems…

- Inference
- Parameter Learning
- Structure Learning

Inference

Inference in StaRAI lies at the intersection between **logical inference** and **probabilistic inference**

Logical Inference

Inferring the truth value of some logic facts, given a collection of some **facts** and some **rules**

Probabilistic inference

Inferring the **posterior** distribution of unobserved **random** variables, given observed ones

Parameter Learning

Typically, StaRAI models specify a set of **parameters** (probabilities or real values) attached to rules/clauses

These parameters can be **learned** from data

Structure Learning

A much more challenging problem would be that of directly learning the **rules** (the structure) of the model

Different approaches…

- Jointly learn parameters **and** rules
- **First** learn rules (i.e., with ILP), **then** their weights

Tasks

Typical tasks in Statistical Relational AI

- Entity classification
- Entity resolution
- Link prediction
- …

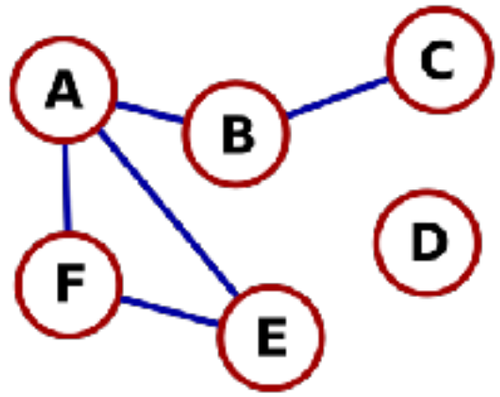
For most of the applications, there might be need to perform **collective (joint) classification**

Entity Classification

- User profiles in a social network
- Gene functions in a regulatory network
- Congestions in a transportation network
- Service requests in p2p networks
- Fault diagnosis in sensor networks
- Hypertext categorization on the Internet ...
- …

Entity Classification

Image from Wikipedia



Which features?

- Use attributes of each **node**
- Use attributes of **neighbourhood**
- Use attributed coming from the **graph structure**
- Use **labels** of other nodes

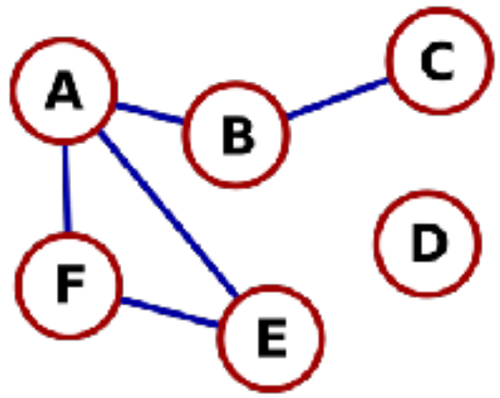
Principle of **co-citation regularity**: similar individuals tend to be related/connected to the same things

Link Prediction

- Friendship in a social network
- Recommendation in a customer-product network
- Interaction in a biological network
- Congestion in a transportation network
- Congestion in a p2p network
- Support/Attack links in argumentation mining
- …

Link Prediction

Image from Wikipedia



Which features?

- Use attributes of **edge**
- Use attributes of **involved nodes**
- Use attributed coming from the **graph structure**
- Use **labels** of other edges

Concept of **homophily**: a link between individuals is correlated with such individuals being similar in nature

Tasks

Statistical Relational AI tasks have some peculiarities

- Examples are typically **not independent**
- Networks are very often **dynamic**
- It might be tricky to perform **model validation**
- …

Tasks

Dynamic networks:

- **Nodes and links** may change over time
- Node and link **properties** may change over time

Shall we predict the **evolution** of the network?

Use the network at time T for **training** and the network at time T+K for **validation/testing**

Tasks

How to perform model validation over network(s), given that examples are not independent?

Possible scenarios:

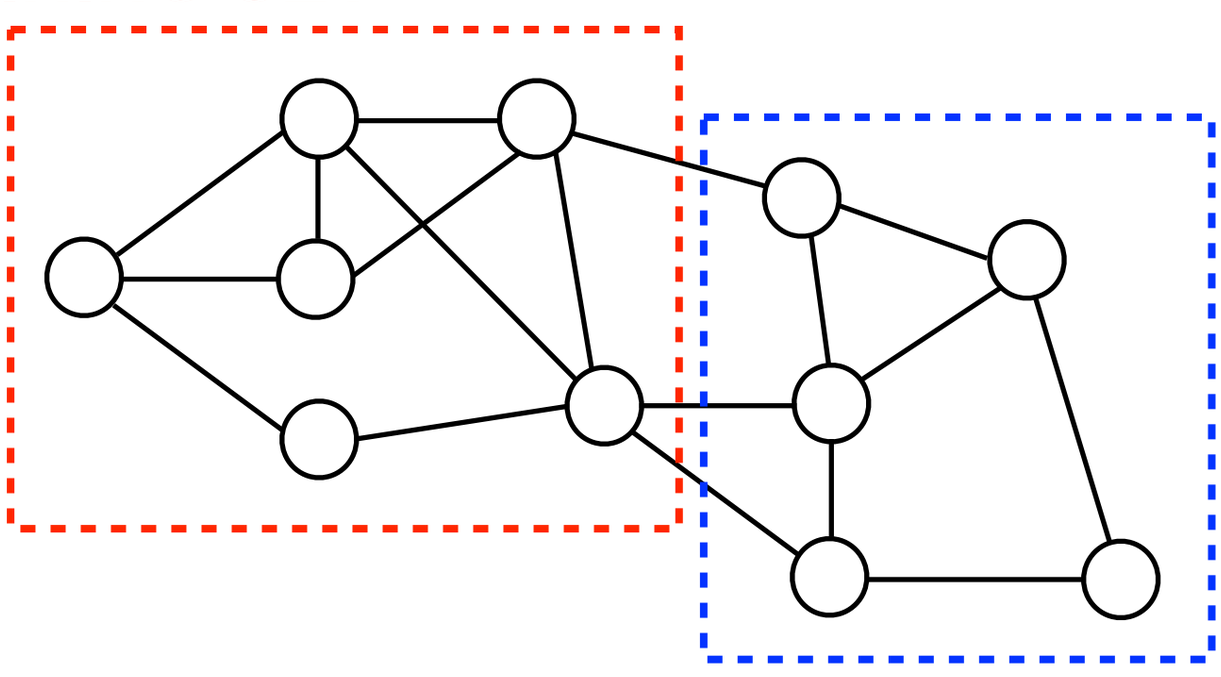
1. **A single static** network (e.g., recommendation)
2. **Many small** networks (e.g., molecules, proteins)
3. **A single evolving** network (e.g., traffic, transport)

Tasks

Validation with a single static network

TRAINING SET

SPLIT THE NETWORK BY CUTTING SOME EDGES

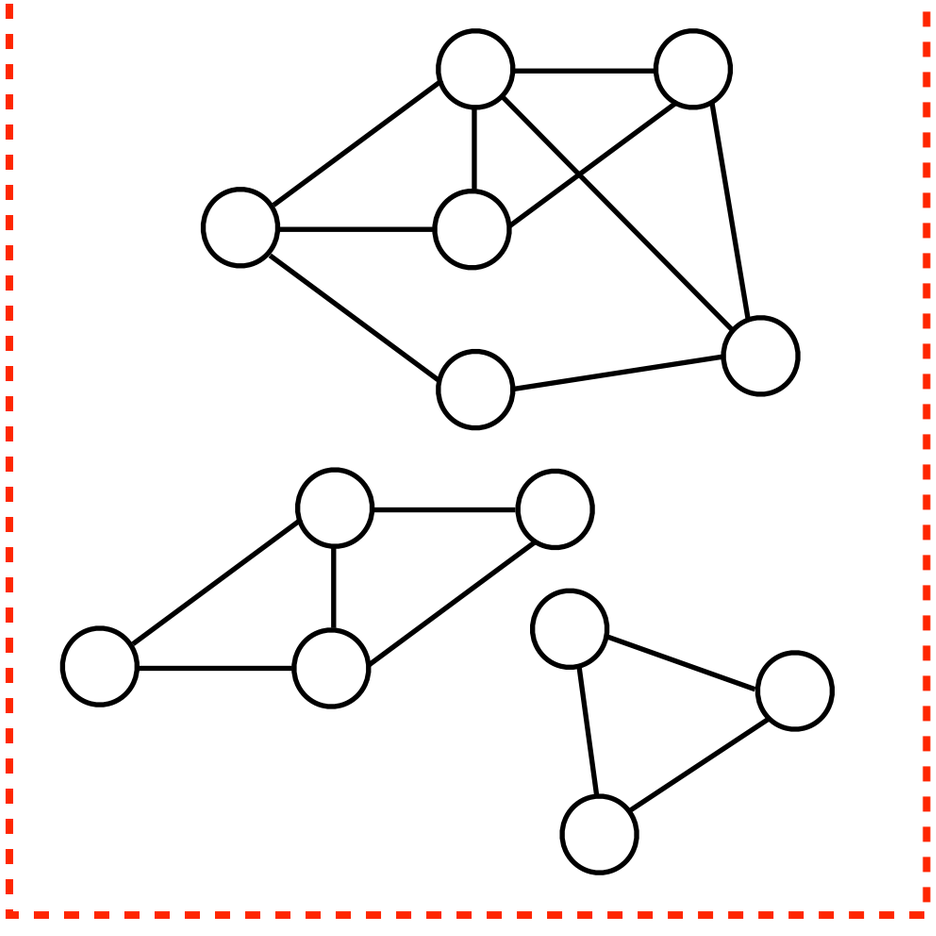


TEST SET

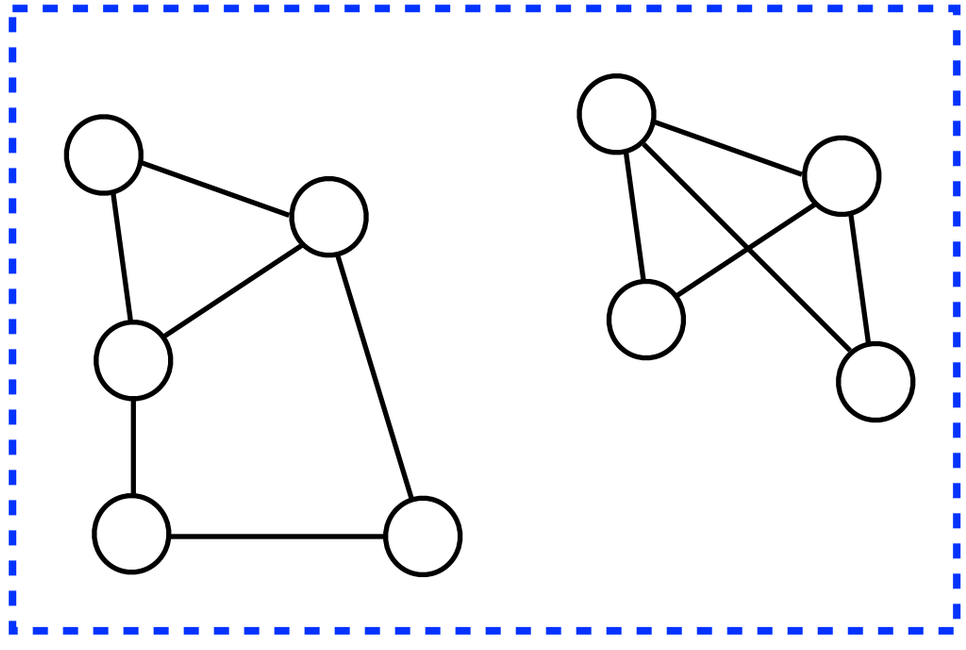
Tasks

Validation with many small networks

TRAINING SET



SPLIT THE NETWORKS INTO DISJOINT SETS

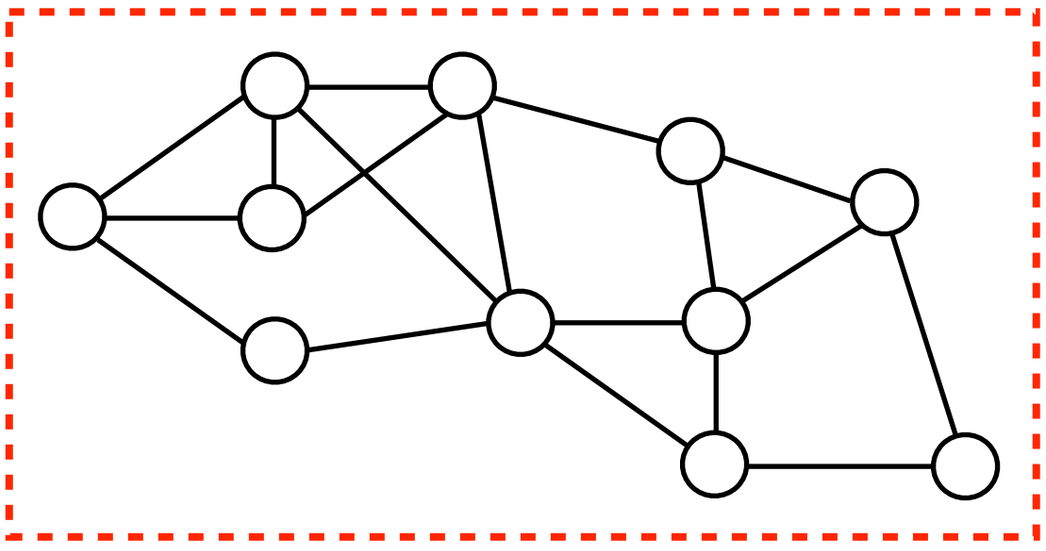


TEST SET

Tasks

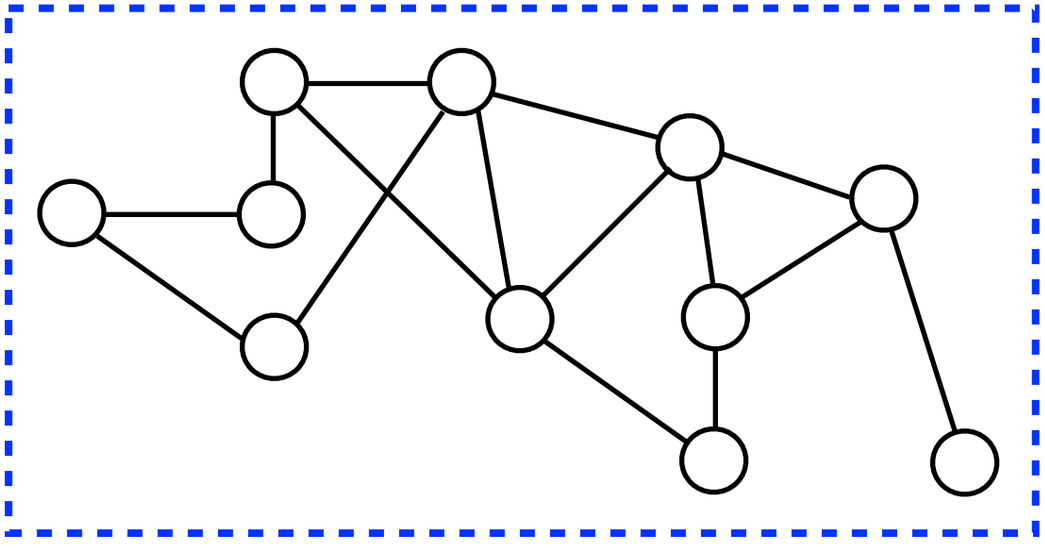
Validation with a single evolving network

TRAINING SET



CONSIDER DIFFERENT TIMES FOR TRAINING AND TEST

TEST SET



Markov Logic Networks

Logic imposes **hard** constraints on the set of possible worlds. Markov logic exploits **soft** constraints.

A Markov Logic Network is defined by:

- a set of first-order **formulae**
- a set of **weights**, one attached to each formula

A world violating a formula becomes **less probable** but **not** impossible!

Markov Logic Networks

Example

```
1.2 Friends(x,y) ^ WatchedMovie(x,m) => WatchedMovie(y,m)
2.3 Friends(x,y) ^ Friends(y,z) => Friends(x,z)
0.8 LikedMovie(x,m) ^ Friends(x,y) => LikedMovie(y,m)
```

The **higher** the weight of a clause =>
  => The **lower** the probability for a world violating that clause

What is a world or Herbrand interpretation?
  => **A truth assignment** to all **ground** predicates

Markov Logic Networks

Beware of the **differences** in the **syntax**…

- In MLN, constants are uppercase (e.g., Alice) and variables are lowercase (e.g., person)
- In ProbLog, constants are lowercase (e.g., alice) and variables are uppercase (e.g., Person)

Markov Logic Networks

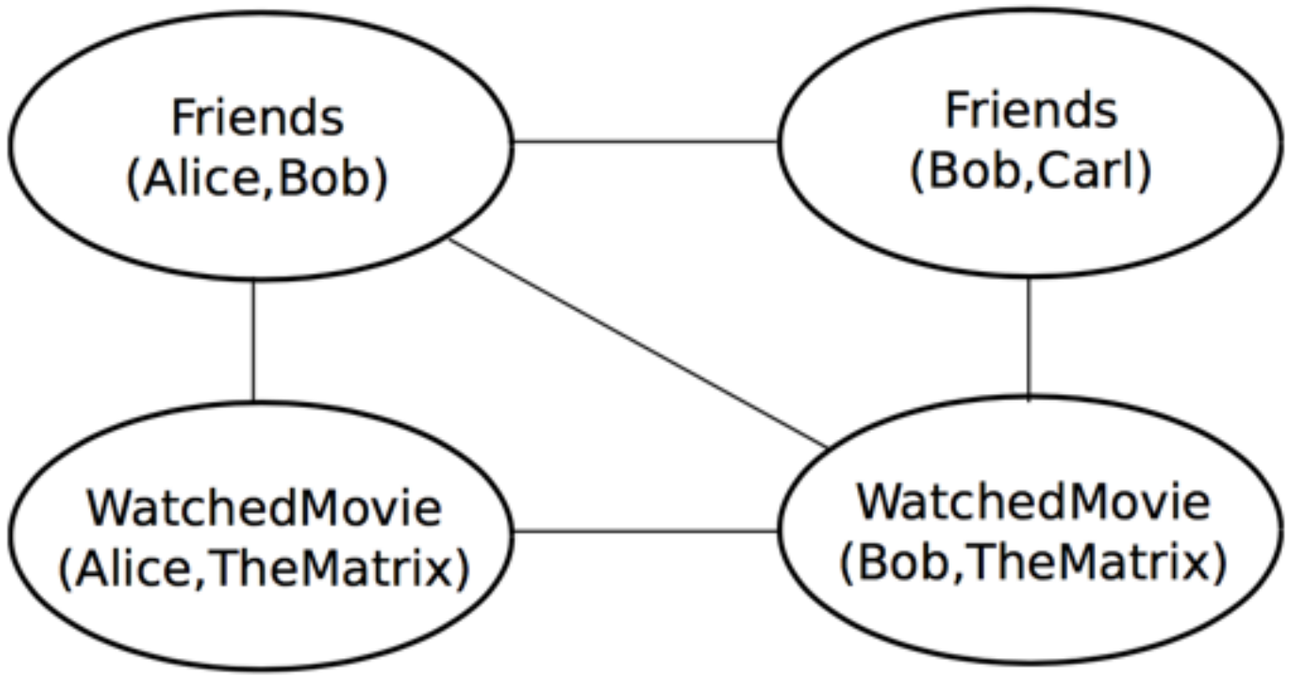
Together with a (finite) set of (unique and possibly **typed**) constants, an MLN defines a **Markov Network** which contains:

1. a binary **node** for each predicate grounding in the MLN, with value 0/1 if the **atom** is false/true
2. an **edge** between two nodes appearing together in (at least) one **formula** on the MLN
3. a **feature** for each formula grounding in the MLN, whose value is 0/1 if the formula is false/true, and whose **weight** is the weight of the formula

Markov Logic Networks

Set of constants:

```
people = {Alice,Bob,Carl,David}
movie = {BladeRunner,ForrestGump,PulpFiction,TheMatrix}
```



Markov Logic Networks

Special cases of MLNs include:

- Markov networks
- Log-linear models
- Exponential models
- Gibbs distributions
- Boltzmann machines
- Logistic regression
- Hidden Markov Models
- Conditional Random Fields
- …

Markov Logic Networks

The semantics of MLNs induces a **probability distribution** over all possible worlds. We indicate with X a set of random variables represented in the model, then we have:

$$P(X = x) = \frac{\exp\left(\sum_{F_i \in \mathcal{F}} w_i n_i(x)\right)}{Z}$$

being $n_i(x)$ the number of true groundings of formula i in world x and Z is the partition function

$$Z = \sum_{x \in \mathcal{X}} \exp\left(\sum_{F_i \in \mathcal{F}} w_i n_i(x)\right)$$

Markov Logic Networks

The definition is similar to the joint probability distribution induced by a **Markov network** and expressed with a log-linear model:

$$P(X = x) = \frac{\exp\left(\sum_{F_i \in \mathcal{F}} w_i n_i(x)\right)}{Z}$$

$$P(X = x) = \frac{\exp\left(\sum_j w_j f_j(x)\right)}{Z}$$

Markov Logic Networks

Discriminative setting: typically, some atoms are always observed (evidence X), while others are unknown at prediction time (query Y)

EVIDENCE	QUERY
Friends(Alice,Bob)	LikedMovie(Alice,BladeRunner) ?
Friends(Bob,Carl)	LikedMovie(Alice,PulpFiction) ?
WatchedMovie(Alice,PulpFiction)	LikedMovie(Bob,BladeRunner) ?
WatchedMovie(David,BladeRunner)	LikedMovie(Bob,TheMatrix) ?
...	...

$$P(Y = y|X = x) = \frac{\exp\left(\sum_{F_i \in \mathcal{F}} w_i n_i(x, y)\right)}{Z_x}$$

Markov Logic Networks

In the discriminative setting, **inference** corresponds to finding the most likely interpretation (MAP – Maximum A Posteriori) **given the observed evidence**

$$y^* = \underset{y}{\operatorname{argmax}}\, P(Y = y | X = x)$$

- **#P-complete** problem => approximate algorithms
- MaxWalkSAT [Kautz et al., 1996], **stochastic local search** => minimize the sum of unsatisfied clauses

Markov Logic Networks

MaxWalkSAT algorithm

```
for i ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if sum of weights (satisfied clauses) > threshold then
            return solution
        c ← random unsatisfied clause
        with probability p
            flip a random variable in c
        else
            flip variable in c that maximizes sum of weights (satisfied clauses)
return failure, best solution found
```

Markov Logic Networks

MaxWalkSAT: key ideas…

- start with a **random** truth value assignment
- **flip** the atom giving the highest **improvement** (greedy)
- can get stuck in **local minima**
- **sometimes** perform a **random flip**
- **stochastic** algorithm (many runs often needed)
- need to build the **whole ground network**!

Markov Logic Networks

Besides MAP inference, Markov Logic allows to compute also the **probability** that each atom is true

Key idea: employ a MonteCarlo approach

- MCMC with Gibbs sampling
- MC-SAT (**sample** over satisfying assignments)
- …

Now moving towards **lifted inference**!

Markov Logic Networks

MC-SAT Algorithm

X(0) ← A **random solution** satisfying all **hard** clauses
for k ← 1 to num_samples
    M ← Ø
    forall C satisfied by X(k–1)
        With probability 1 – exp(–w) add C to M
    endfor
    X (k) ← A **uniformly random solution** satisfying M
endfor

Lazy variant: only ground what is needed (active)

Markov Logic Networks

Parameter learning: maximize conditional log likelihood (CLL) of query predicates given evidence: **inference as subroutine**!

$$\frac{\partial}{\partial w_i} \log P(Y = y | X = x) = n_i - E_w[n_i]$$

Several algorithms for this task:

- Voted Perceptron
- Contrastive Divergence
- Diagonal Newton
- (Preconditioned) Scaled Conjugate Gradient

Markov Logic Networks

Directly **infer the rules** from the data

Classic task for Inductive Logic Programming (ILP), to be addressed jointly or separately wrt parameter learning

- Modified ILP algorithms (e.g., Aleph)
- Bottom-Up Clause Learning
- Iterated Local Search
- Structural Motifs

Still much an **open problem**!

Markov Logic Networks

Remarks on expressivity

MLNs exploit first-order logic clauses

- Infinite weights for hard constraints (pure FOL rules)
- Existential and universal quantifiers
- Contradictions are allowed

Existential quantifiers are translated into a **disjunction**, with the caveat that it can make groundings explode!

Tractable Markov Logic

- Exploit **tractable subsets** of first-order logic!
- Relations such as **subclass**, subpart, instance of, …
- Use **probabilistic theorem proving** for inference
- Compute partition function in **polynomial** time/space

http://alchemy.cs.washington.edu/lite

MLN vs. ProbLog vs. LPAD

Weights vs. probabilities

- In an MLN, the weight of formula F is the **log odds** between a world where F is true and a world where F is false, **other things being equal**
- In ProbLog and LPAD, we model **directly** the **probability** that a rule is true

Interpreting MLN Weights

Back to the probability distribution induced by an MLN

$$P(X = x) = \frac{\exp\left(\sum_{F_i \in \mathcal{F}} w_i n_i(x)\right)}{Z}$$

Suppose to have **four** rules with **one** grounding each

Suppose to have two distinct MLNs, where the only difference is that one of the rules has **double** weight

What happens to the probability distribution?

Interpreting MLN Weights

MLN #1

$$P(X = x) = \frac{\exp(w_0 + w_1 + w_2 + w_3)}{Z}$$

MLN #2

$$P(X = x) = \frac{\exp(w_0 + w_1 + w_2 + 2 \cdot w_3)}{Z}$$

Odd Ratio

$$\frac{\exp(w_0 + w_1 + w_2 + 2 \cdot w_3)}{w_0 + w_1 + w_2 + w_3} = e^{w_3}$$

Alchemy Data Format

.mln file

- Predicate definition (including types)
- Rules (possibly including weights)

.db file

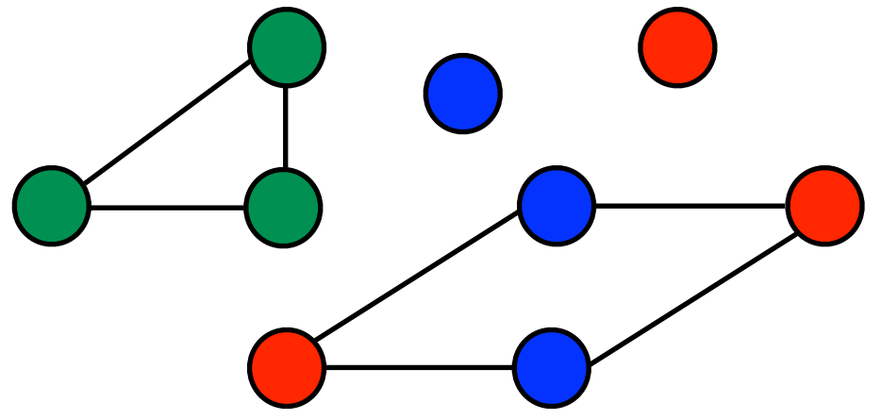
Ground evidence predicates (during training and test)

Ground query predicates (during training only)

Open vs. Closed world assumption!

Example 1

Toy Link Prediction Problem



- Tiny network
- Nodes have a color
- The probability of a link between two nodes depend on the colours of such nodes

Example 1

Toy Link Prediction Problem (MLN)

.mln file (version 1)

```
Red(node)
Blue(node)
Green(node)
Link(node,node)

Link(x,y) <=> Link(y,x).
Red(x) ^ Red(y) => Link(x,y)
Green(x) ^ Green(y) => Link(x,y)
Blue(x) ^ Blue (y) => Link(x,y)
Red(x) ^ Green(y) => Link(x,y)
Green(x) ^ Red(y) => Link(x,y)
. . .
```

Example 1

Toy Link Prediction Problem (MLN)

.db file (version 1)

```
Red(N1)
Green(N2)
Green(N3)
Blue(N4)
Red(N5)
. . .
Link(N2,N3)
Link(N3,N2)
Link(N2,N10)
. . .
!Link(N1,N1)
!Link(N1,N2)
. . .
```

! indicates the negation sign in Alchemy

Example 1

Toy Link Prediction Problem (MLN)

.mln file (version 2)

```
Color(node,value)
Link(node,node)

Link(x,y) <=> Link(y,x).
Color(x,+c1) ^ Color(y,+c2) => Link(x,y)
```

Using the + is a shortcut of the Alchemy language to indicate all possible combinations of constants!

Example 1

Toy Link Prediction Problem (MLN)

.db file (version 2)

```
Color(N1,Red)
Color(N2,Green)
Color(N3,Green)
Color(N4,Blue)
Color(N5,Red)
. . .
Link(N2,N3)
Link(N3,N2)
Link(N2,N10)
. . .
!Link(N1,N1)
!Link(N1,N2)
. . .
```

Example 1

Toy Link Prediction Problem (ProbLog)

model file

```
t(_)::link(X,Y) :- red(X), red(Y).
t(_)::link(X,Y) :- green(X), green(Y).
t(_)::link(X,Y) :- blue(X), blue(Y).
t(_)::link(X,Y) :- red(X), blue(Y).
. . .

1::link(X,Y) :- link(Y,X).

red(n1).
green(n2).
green(n3).
. . .
```

Example 1

Toy Link Prediction Problem (ProbLog)

data file

```
evidence(link(n2,n3),true).
evidence(link(n3,n2),true).
evidence(link(n2,n10),true).
evidence(link(n10,n2),true).
evidence(link(n3,n10),true).
evidence(link(n10,n3),true).
. . .
evidence(link(n1,n1),false).
evidence(link(n1,n2),false).
evidence(link(n1,n3),false).
. . .
```

Example 1

Toy Link Prediction Problem (ProbLog)

command line

```
> problog lfi model.pl data.pl -O output.pl


> problog -h

> problog lfi -h
```

Example 1

Toy Link Prediction Problem (cplint)

Load splicover and initialize input theory

```
:- use_module(library(slipcover)).
:- sc.
:- set_sc(verbosity,3).

:- begin_in.
link(X,Y):0.1 :- red(X), red(Y).
link(X,Y):0.1 :- green(X), green(Y).
link(X,Y):0.1 :- blue(X), blue(Y).
link(X,Y):0.1 :- red(X), blue(Y).
link(X,Y):0.1 :- blue(X), red(Y).
. . .
:- end_in.
```

NOTE: the value of the probability is not important, but necessary for the learning!

Example 1

Toy Link Prediction Problem (cplint)

Background knowledge (if any) and language bias

```
output(link/2).
input_cw(red/1).
input_cw(green/1).
input_cw(blue/1).

determination(link/2,red/1).
determination(link/2,green/1).
determination(link/2,blue/1).

modeh(*,link(node,node)).
modeb(*,red(-node)).
modeb(*,blue(-node)).
modeb(*,green(-node)).
```

Example 1

Toy Link Prediction Problem (cplint)

Training data

```
fold(train,[training_set]).

begin(model(training_set)).
red(n1).
green(n2).
. . .
link(n2,n3).
link(n3,n2).
. . .
neg(link(n1,n1)).
neg(link(n1,n2)).
end(model(training_set)).
```

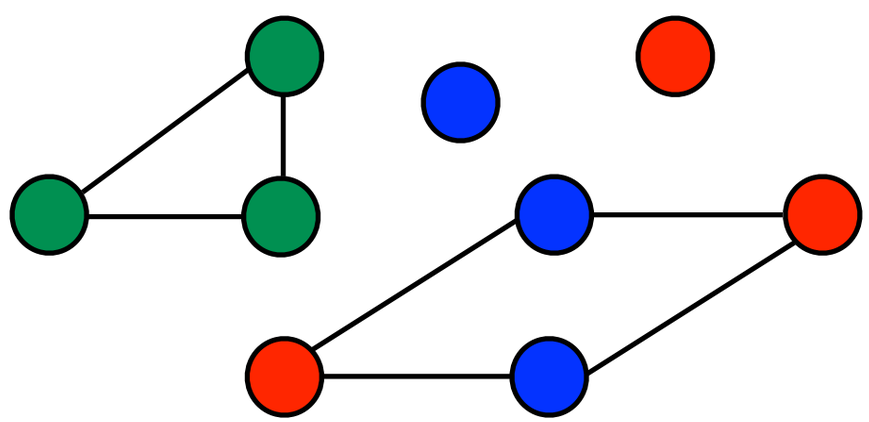
Example 1

Let us consider the model again…

Is this **really** relational learning?

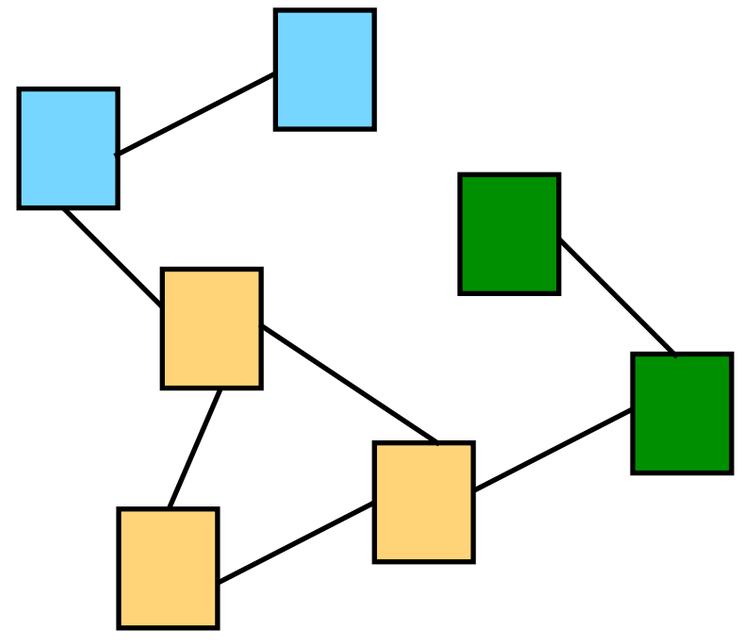
Did we **really** perform collective classification?

Which rules did **spread** information among nodes?



Example 2

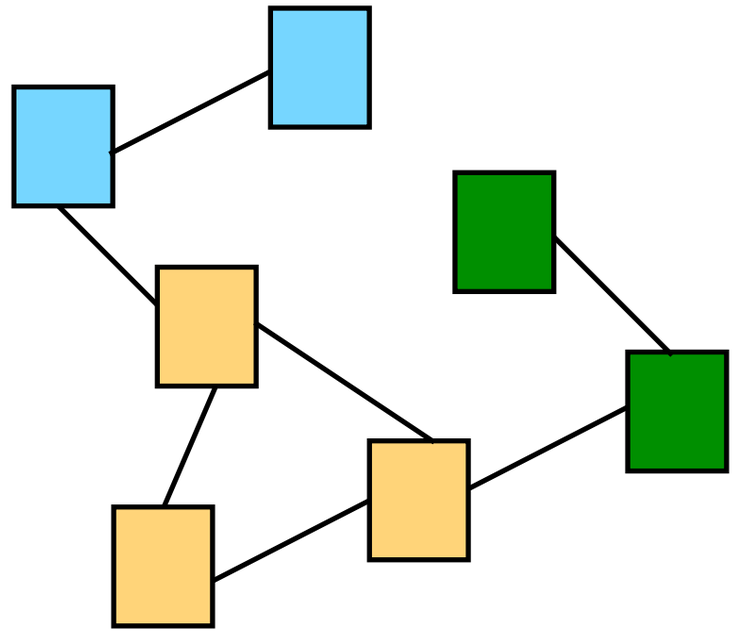
Hypertext classification (MLN)



```
Link(page,page)
HasWord(page,word)
Topic(page,topic)

HasWord(p,+w) => Topic(p,+t)
Topic(p,t) ^ Link(p,q) => Topic(q,t)
```

Example 2

Hypertext classification (ProbLog)



We can use a trick similar to the use of + for the Alchemy syntax!

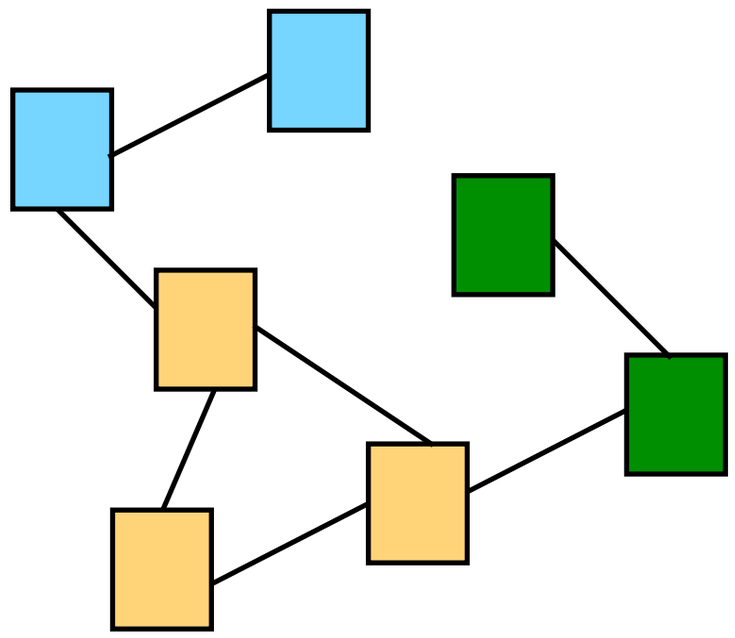
```
t(_)::topic(P,T) :- link(P,Q), topic(Q,T).
t(_,W,T)::topic(P,T) :- hasword(P,W).
```

Example 2

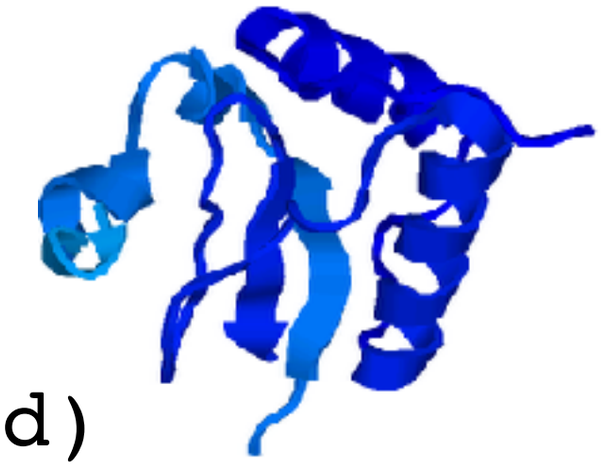
Now, this model **does exploit** relational information!

Could we model the same problem with standard machine learning classifiers (i.e., SVM, NN, RF)?

Yes? No? Maybe?



Example 3



Protein Secondary Structure

```
Residue(sequence,position,aminoacid)
SecondaryStructure(sequence,position,class)

Residue(s,p,+a) => SecondaryStructure(s,p,+c)

SecondaryStructure(s,p1,c) =>
        SecondaryStructure(s,succ(p1),c)
```

Example 3

Beware of unwanted (spurious) groundings with MLN!
If the knowledge base contains a predicate such as:

`Residue(sequence,position,aminoacid)`

then Alchemy will **expect** ground predicates for all possible combinations of sequences and positions, even if a position is not part of a sequence!

This is not important for **evidence** predicates (since they are closed world) but for **query** predicates!

Example 3

For example, with the following database:

```
Residue(S1,1,C)
. . .
Residue(S1,72,A)
Residue(S2,1,R)
. . .
Residue(S2,66,S)
```

then Alchemy will also build/expect the **query** predicate `SecondaryStructure(S2,P72,CLASS)`

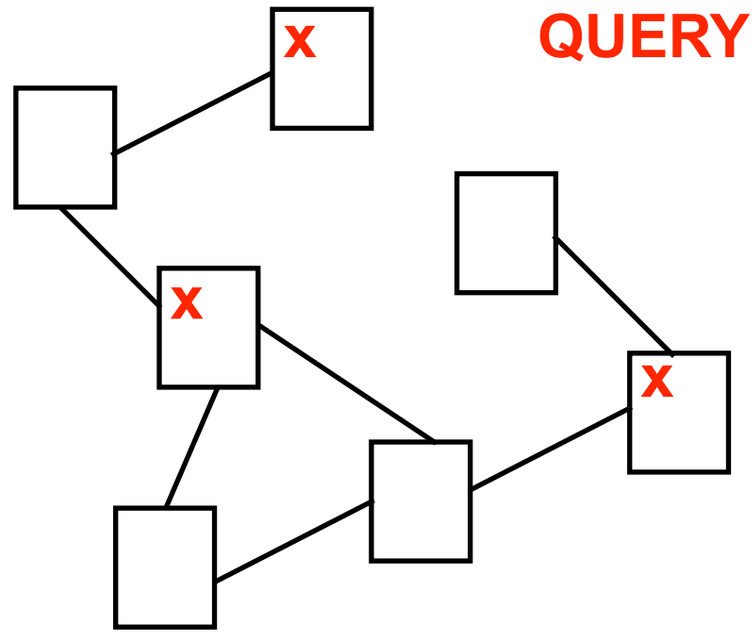
Example 3

This problem can be circumvented by using the `multipleDatabases` option, which allows for multiple `.db` files with **independent** constant sets

With ProbLog and LPAD this problem does not occur because we perform **learning from interpretations**: you basically can have a different interpretation for each training “world”.

Example 4

Information Retrieval — MLN



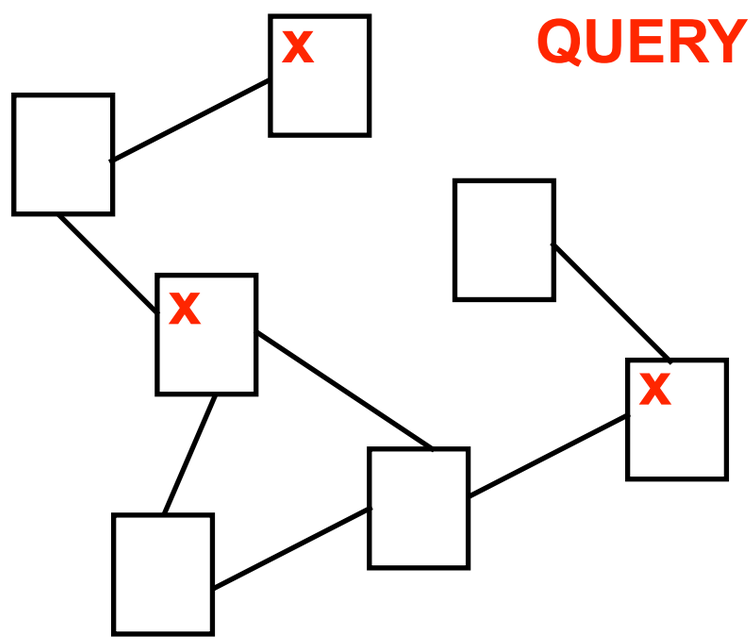

```
InQuery(word)
HasWord(page,word)
Link(page,page)
Relevant(page)

HasWord(p,+w) ^ InQuery(w) => Relevant(p)
Relevant(p) ^ Link(p,q) => Relevant(q)
```

Example 4

Information Retrieval

Try to perform **weight** learning and then **inference** with Alchemy with default parameters…

What is the problem?

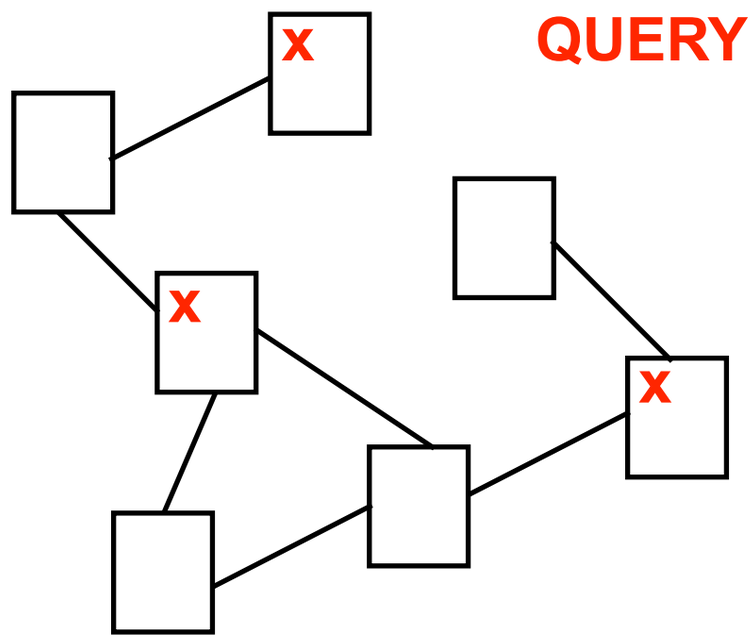


Example 4

Information Retrieval

Try to perform **structure** learning with Alchemy with default parameters…

What is the problem?



Example 4

Information Retrieval — ProbLog

```
t(_)::relevant(P).
t(_)::relevant(P) :- hyperlink(Q,P), relevant(Q).
t(_,W)::relevant(P) :- hasword(P,W), inquery(W).

inquery(apartment).
inquery(rent).
inquery(boston).
hasword(p1,house).
hasword(p1,rentals).
```

Example 4

Information Retrieval — ProbLog

```
evidence(relevant(p1),true).
evidence(relevant(p2),false).
evidence(relevant(p3),true).
evidence(relevant(p4),true).
evidence(relevant(p5),false).
evidence(relevant(p6),false).
```

Example 4

Information Retrieval — ProbFOIL (Structure Learning)

```
% Modes
mode(inquery(+)).
mode(inquery(c)).
mode(hasword(+,c)).
mode(hasword(+,-)).
mode(hyperlink(+,-)).
mode(hyperlink(-,+)).

% Type definitions
base(relevant(page)).
base(hyperlink(page,page)).
base(hasword(page,word)).
base(inquery(word)).
```

Example 4

Information Retrieval — ProbFOIL (Structure Learning)

```
% Target
learn(relevant/1).

% How to generate negative examples
example_mode(auto).
```

Command line

```
probfoil information_retrieval_settings.pl
information_retrieval_data_full.pl
```

Example 4

Information Retrieval — cplint (Parameter Learning)

```
:- use_module(library(slipcover)).
:- sc.
:- set_sc(max_iter,5).
:- set_sc(verbosity,3).

:- begin_in.
relevant(P):0.1 :- hyperlink(Q,P), relevant(Q).
% relevant(P):t(_,W): :- hasword(P,W), inquery(W).
relevant(P):0.1 :- hasword(P,apartment), inquery(apartment).
relevant(P):0.1 :- hasword(P,boston), inquery(boston).
relevant(P):0.1 :- hasword(P,rent), inquery(rent).
:- end_in.
```

Example 4

Information Retrieval — cplint (Parameter Learning)

```
:- begin_bg.
inquery(apartment).
inquery(rent).
inquery(boston).
hasword(p1,house).
hasword(p1,rentals).
hyperlink(p1,p2).
hyperlink(p1,p3).
:- end_bg.

% Fold definition
fold(train,[train1]).
```

Example 4

Information Retrieval — cplint (Parameter Learning)

```
% Language bias
output(relevant/1).
input_cw(hasword/2).
input_cw(hyperlink/2).
input_cw(inquery/1).

determination(relevant/1,hyperlink/2).
determination(relevant/1,hasword/2).
determination(relevant/1,inquery/1).

modeh(*,relevant(page)).
modeb(*,hyperlink(-page,page)).
modeb(*,hasword(-page,word)).
modeb(*,inquery(word)).
```

Example 4

Information Retrieval — cplint (Parameter Learning)

```
% Models / Examples
begin(model(train1)).
relevant(p1).
neg(relevant(p2)).
relevant(p3).
relevant(p4).
neg(relevant(p5)).
neg(relevant(p6)).
end(model(train1)).

induce_par([train],P).
```

Example 4

Information Retrieval — cplint (Structure Learning)

```
:- use_module(library(slipcover)).
:- sc.

:- set_sc(verbosity,3).
:- set_sc(initial_clauses_per_megaex,3).

:- begin_in.
:- end_in.

:- begin_bg.
:- end_bg.

% Fold definition
fold(train,[train1]).
```

Example 4

Information Retrieval — cplint (Structure Learning)

```
output(relevant/1).
input_cw(hasword/2).
input_cw(hyperlink/2).
input_cw(inquery/1).

determination(relevant/1,hyperlink/2).
determination(relevant/1,hasword/2).
determination(relevant/1,inquery/1).

modeh(*,relevant(+page)).
modeb(*,hyperlink(-page,+page)).
modeb(*,hyperlink(+page,-page)).
modeb(*,hasword(+page,-#word)).
modeb(*,inquery(-#word)).
```

Example 4

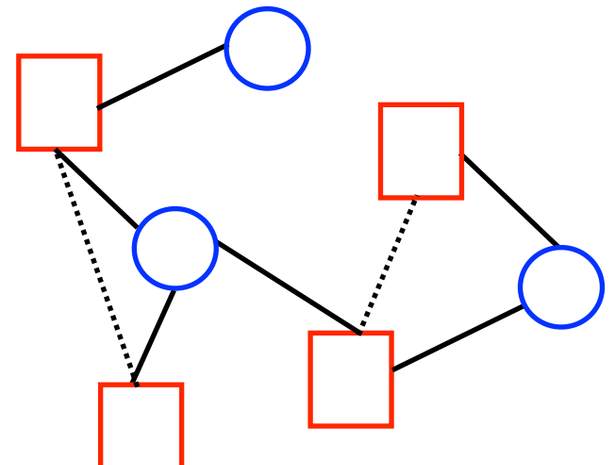
Information Retrieval — cplint (Structure Learning)

```
begin(model(train1)).
inquery(apartment).
inquery(rent).
inquery(boston).
hasword(p1,house).
hasword(p1,rentals).
hasword(p1,massachussets).
. . .
hyperlink(p1,p2).
hyperlink(p1,p3).
hyperlink(p4,p3).
. . .
relevant(p1).
neg(relevant(p2)).
end(model(train1)).
```

Example 5

Movie recommendation

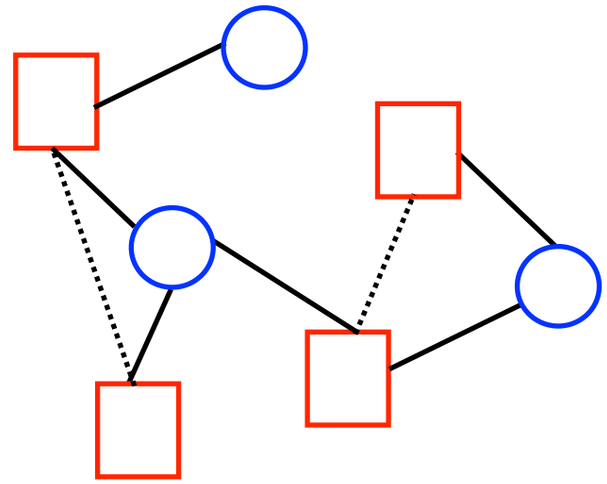
Will person X like movie M?



```
0.3::comedy(X) :- movie(X).
0.4::drama(X) :- movie(X).
0.2::friends(X,Y) :- person(X), person(Y).
0.1::likes(X,M) :- person(X), movie(M).

0.3::likes(X,M) :- comedy(M).
0.2::likes(X,M) :- drama(M).
0.3::likes(X,M) :- friend(X,Y), likes(Y,M).
```

Example 5

```
person(alice). person(bob).
person(carl). person(david).
movie(bladerunner). movie(thematrix).

friend(alice,bob). friend(bob,alice).
friend(bob,david). friend(david,bob).

likes(alice,bladerunner). likes(bob,bladerunner).
likes(carl,thematrix). likes(david,thematrix).
likes(david,bladerunner).

query(likes(alice,thematrix)).
query(likes(carl,bladerunner)).
```



Remarks on Complexity

These SRL frameworks are highly expressive and powerful, but unfortunately they can easily become **memory-intensive** and **time-consuming**

- Try to model the problem differently
- If possible, partition the data
- Reduce the expressivity of the model

Open Challenges

- Efficient inference algorithms
- Scalability
- Structure learning
- Continuous variables/features

Continuous Features

- Hybrid Markov Logic Networks
- Ground-Specific Markov Logic Networks
- DeepProbLog
- Relational Neural Networks
- Learning from Constraints
- TensorLog
- cplint (for inference)
- …

Hybrid MLNs

Introduced in [Wang & Domingos, 2008]

Continuous properties/functions usable as **features**

Extending MC-SAT and MaxWalkSAT algorithms

```
(SomeEvidence(x) < 2.3) => SomeQuery(x)
SomeQuery(x) * (SomeEvidence(x) = 1.2)
```

Ground-Specific MLNs

Introduced in [Lippi & Frasconi, 2009]

Use **neural networks** to predict the weight of rules

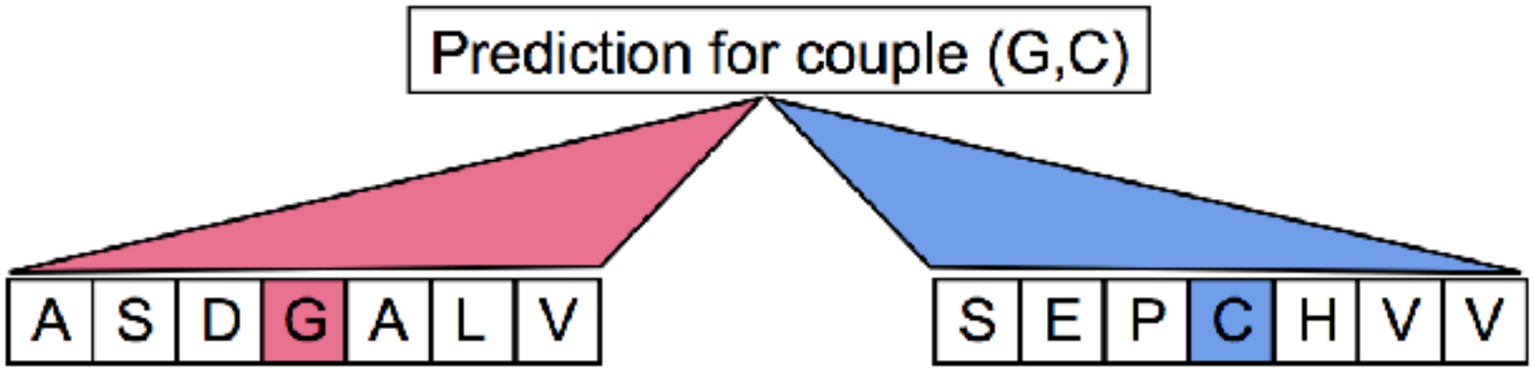
No single weight for each first-order logic formula, but a different weight for each ground formula

Trained by standard back-propagation!

```
w1: Node(X,$Features_X) ^ Node(Y,$Features_Y)
                    => Link(X,Y)

w2: Node(P,$Features_P) ^ Node(Q,$Features_Q)
                    => Link(P,Q)
```

Ground-Specific MLNs

Predict whether two residues in a protein are linked…

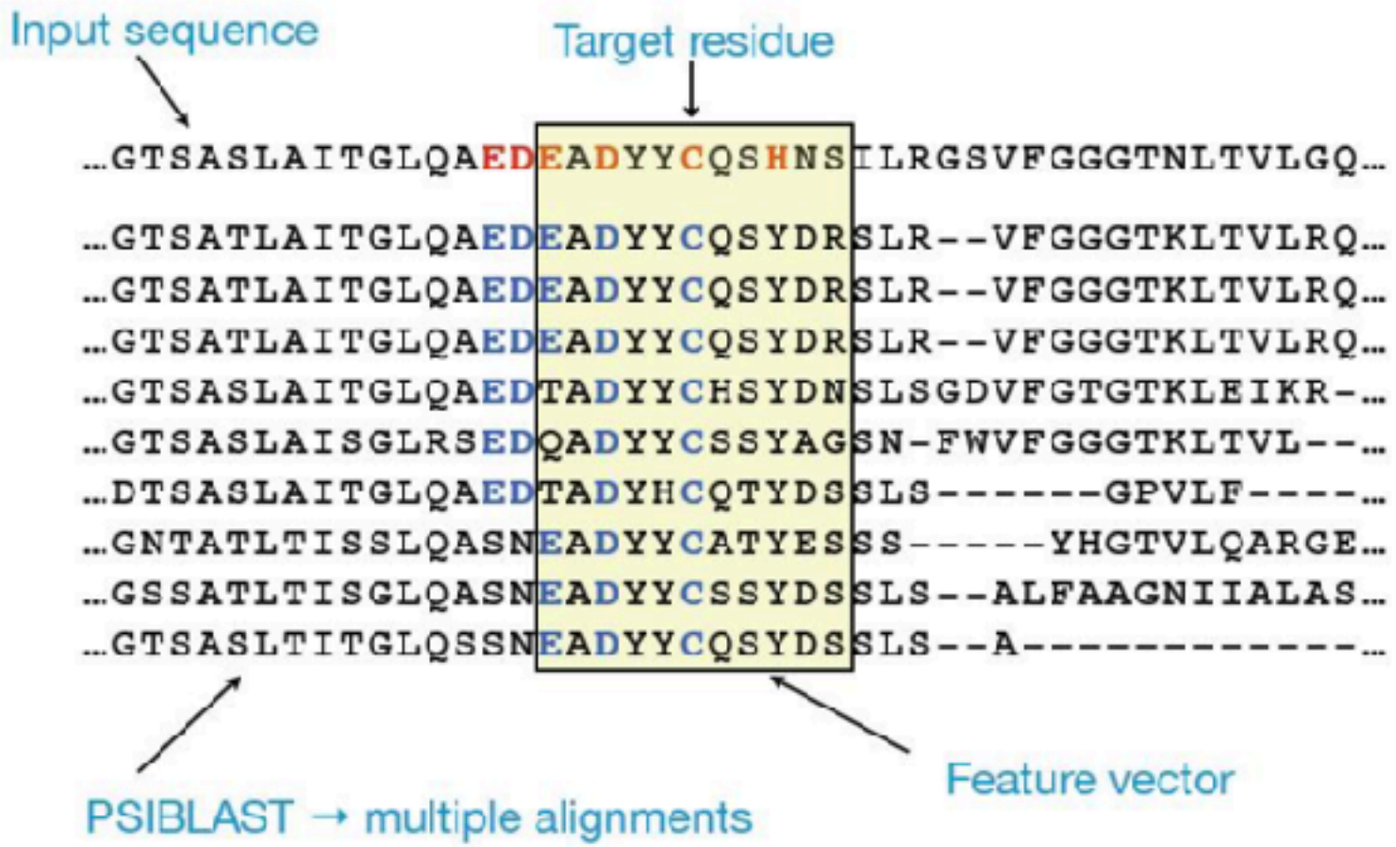


Example

```
Residue(i-1,D) ^ Residue(i,G) ^ Residue(i+1,A) ^
Residue(j-1,P) ^ Residue(j,C) ^ Residue(j+1,H)
                    => Partners(i,j)
```

Ground-Specific MLNs

- Perform multiple alignment
- Use the profile of the window as input features



Ground-Specific MLNs

- Re-parametrization of MLN
- Compute each weight as a function of the grounding

Standard MLN

$$P(Y = y|X = x) = \frac{exp\left(\sum_{F_i \in F_y} w_i n_i(x,y)\right)}{Z_x}$$

MLNs with grounding-specific weights

$$P(Y = y|X = x) = \frac{exp\left(\sum_{F_i \in F_y} \sum_j w_i(c_{ij},\theta_i) n_{ij}(x,y)\right)}{Z_x}$$

Ground-Specific MLNs

- Inference algorithms **do not change**
- Learning algorithms implement **gradient descent**

$$\frac{\partial P(y|x)}{\partial \theta_k} = \frac{\partial P(y|x)}{\partial w_i} \frac{\partial w_i}{\partial \theta_k}$$

where the **first** term is computed by MLN inference and the **second** term is computed by backprop

DeepProbLog

Introduced in [Manhaeve et al., 2018]

Integrating logical reasoning with neural networks

Symbolic and sub-symbolic representation/inference

Ground **neural annotated disjunctions**

Output of NNs translated into **probabilities** (softmax)

End-to-end training with back-propagation

ProbLog and cplint

Remember that also ProbLog and cplint can handle continuous variables for inference…

More Frameworks

- Knowledge-based artificial neural networks [Towell & Shavlik, 1994]
- Learning from Constraints [Diligenti et al. 2012]
- Lifted Relational Neural Networks [Sourek et al., 2015]
- Logic Tensor Networks [Serafini & d'Avila Garcez, 2016]
- Relational Neural Networks [Kazemi & Poole, 2017]
- TensorLog [Cohen et al., 2017]
- Relational recurrent neural networks [Santoro et al., 2018]
- ???

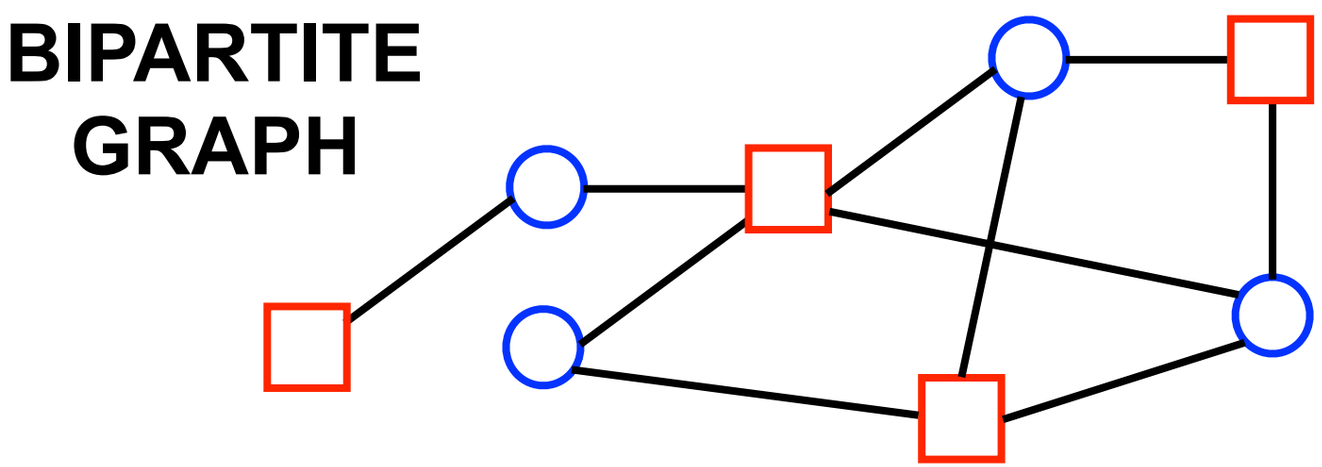
Data Resources

- https://alchemy.cs.washington.edu/data
- https://linqs.soe.ucsc.edu/data
- http://netkit-srl.sourceforge.net/data.html
- http://cplint.ml.unife.it/
- https://snap.stanford.edu/data/

MovieLens Dataset

A dataset of **movie ratings**

- User information (age, sex, occupation, zipcode)
- Movie information (genres, release date)
- Rating information (user, item, rating, timestamp)



Exercise 1

Rating prediction (recommendation)

The aim is to predict the rating a user gives to an item

- Consider past ratings information only
- Consider also user information
- Consider also item information
- How to consider timestamps?

Exercise 2

User classification (profiling)

The aim is to predict some property of a user

- Consider past ratings information only (?)
- Consider also information about other users
- Consider also item information
- How to consider timestamps?

Argumentation Mining

Fig. 1. Example of argument extraction from plain text.

CLAIM 1 While those on the far-right think that immigration threatens national identity, as well as cheapening labor and increasing dependence on welfare.

[...]

Proponents of immigration maintain that, according to Article 13 of the Universal Declaration of Human Rights, everyone has the right to leave or enter a country, along with movement within it. [...] **EVIDENCE 2**

[...]

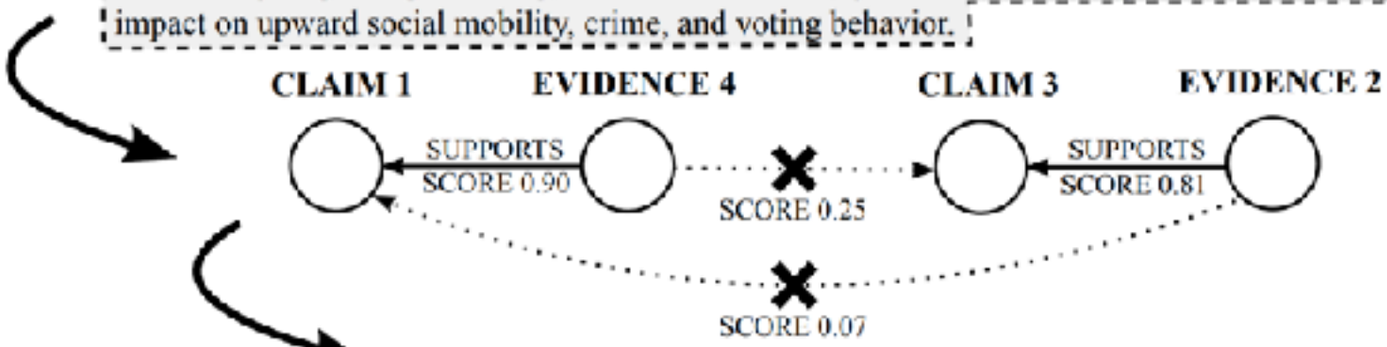
CLAIM 3 Some argue that the freedom of movement both within and between countries is a basic human right, and that the restrictive immigration policies, typical of nation-states, violate this human right of freedom of movement.

(a)

[...]

Immigration has been a major source of population growth and cultural change throughout much of the history of Sweden. The economic, social, and political aspects of immigration have caused controversy regarding ethnicity, economic benefits, jobs for non-immigrants, settlement patterns, impact on upward social mobility, crime, and voting behavior. **EVIDENCE 4**

(b)



(c)

Figure from
[Lippi & Torroni 2016]

Argumentation Mining

The standard pipeline

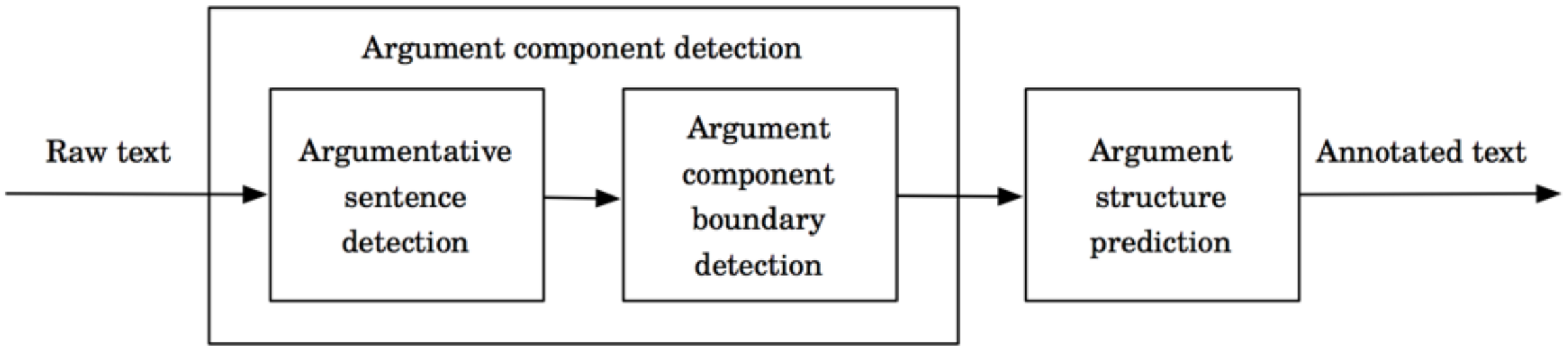


Figure from
[Lippi & Torroni 2016]

Argumentation Mining

Persuasive Essays corpus labeled with

- Claims, MajorClaims, Premises (components)
- Support/Attack (relations)
- Stance (against/for)

Admittedly, [cloning could be misused for military purposes]$_{Claim5}$*. For example, [it could be used to manipulate human genes in order to create obedient soldiers with extraordinary abilities]*$_{Premise9}$*. However, because [moral and ethical values are internationally shared]*$_{Premise10}$*, [it is very unlikely that cloning will be misused for militant objectives]*$_{Premise11}$*.*

To sum up, although [permitting cloning might bear some risks like misuse for military purposes]$_{Claim6}$*, I strongly believe that [**this technology is beneficial to humanity**]*$_{MajorClaim2}$*. It is likely that [this technology bears some important cures which will significantly improve life conditions]*$_{Claim7}$*.*

Figure from [Stab & Gurevych 2016]

Argumentation Mining

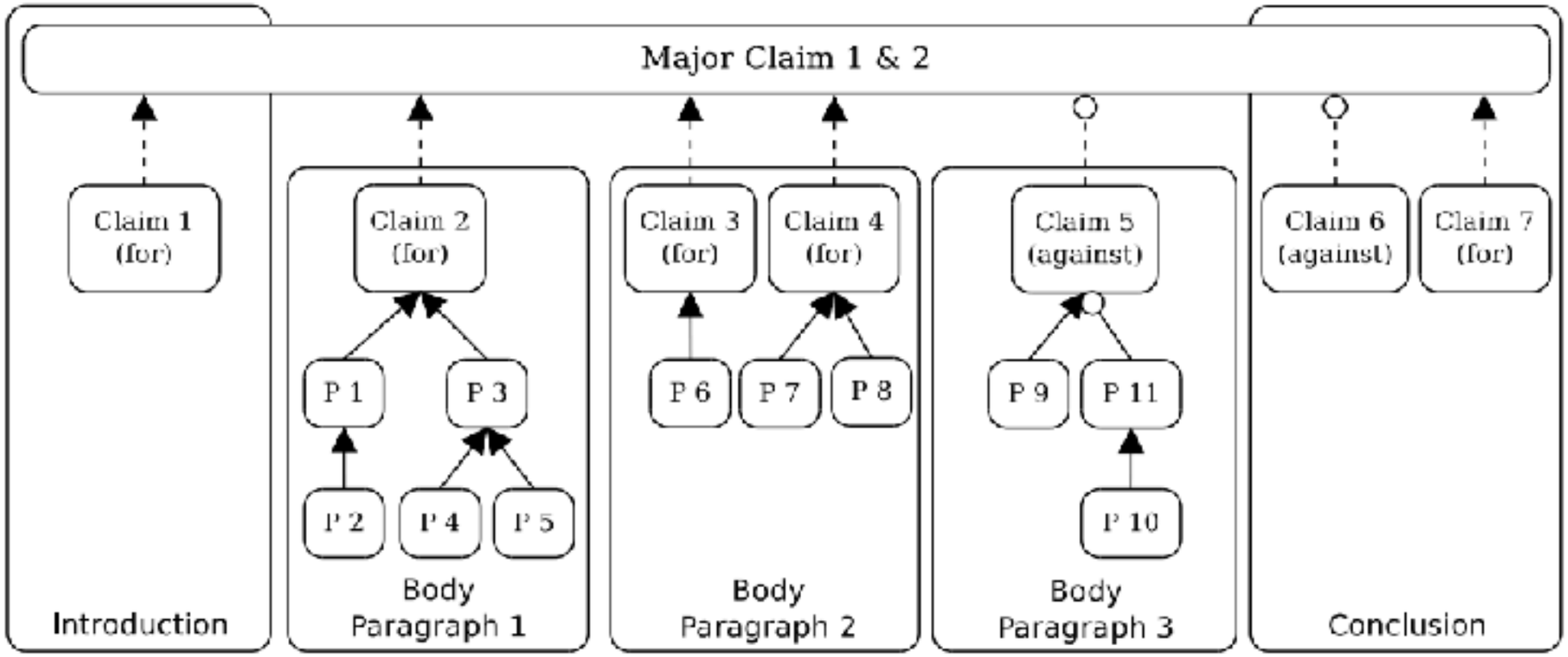


Figure 2
Argumentation structure of the example essay. Arrows indicate argumentative relations. Arrowheads denote argumentative support relations and circleheads attack relations. Dashed lines indicate relations that are encoded in the stance attributes of claims. "P" denotes premises.

Figure from [Stab & Gurevych 2016]

Exercise 3

Argument component classification

The aim is to predict the type of argument component

- Consider **words** in a sentence
- Consider **sequence** of argument components
- Consider **order/position** of sentences

Exercise 4

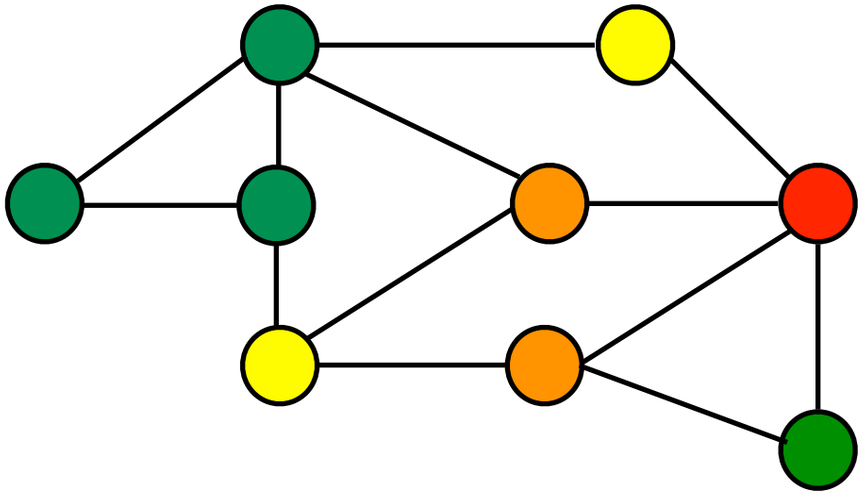
Structure prediction

The aim is to predict the relations between argument components (i.e., the links in the argument graph)

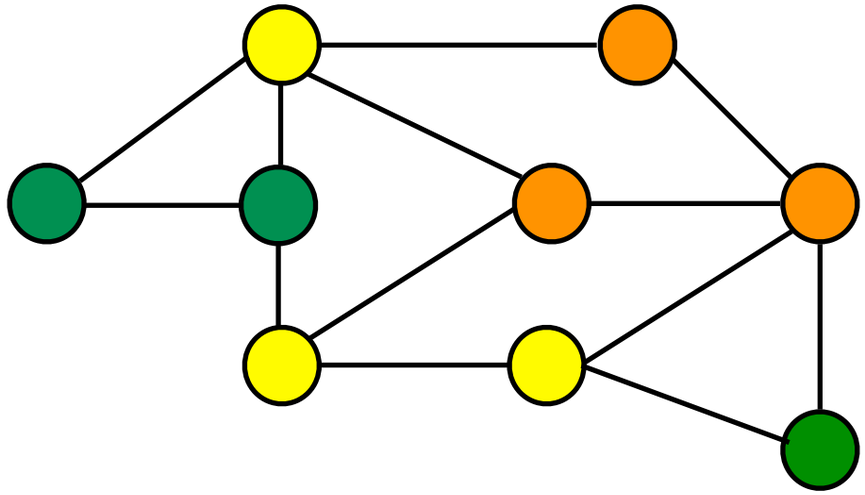
- Consider **words** in sentences
- Consider **order/position** of sentences
- Consider **distances** between components
- **Jointly** predict argument components?

Exercise 5

Traffic congestion



Traffic at time T

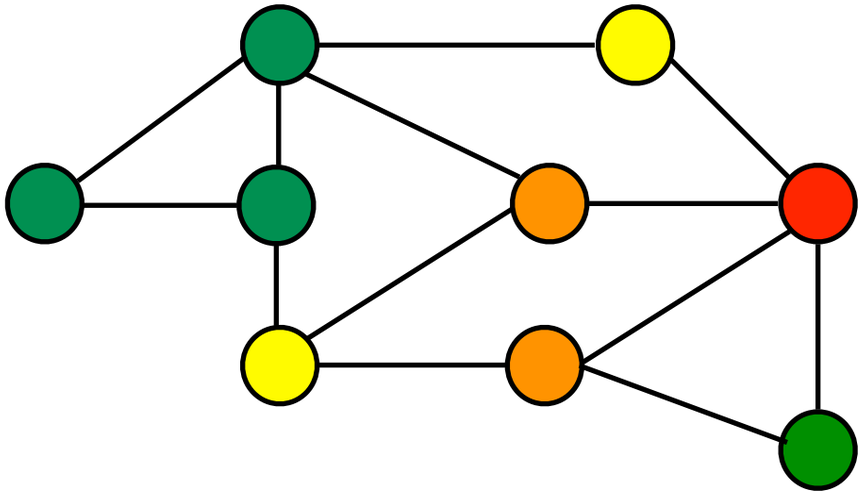


Traffic at time T+1

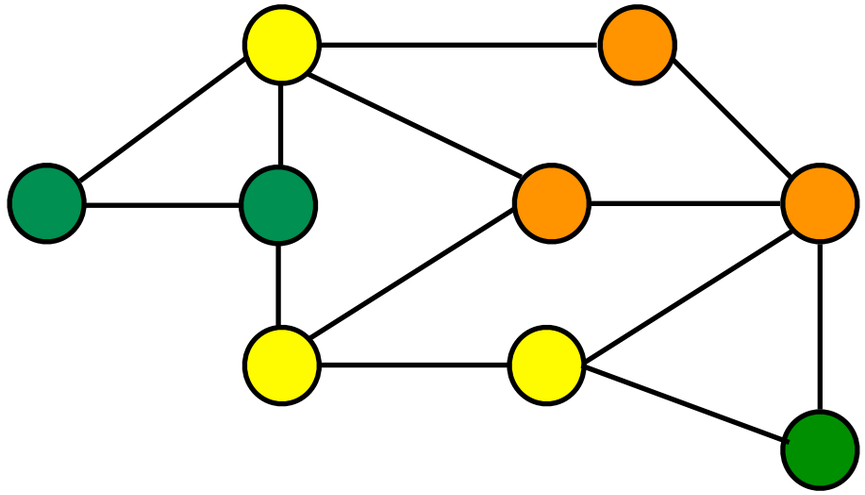
Exercise 5

Traffic congestion

REMEMBER TO
COMPUTE BASELINES!
WHICH ARE GOOD BASELINES?



Traffic at time T



Traffic at time T+1

Exercise 6

Finding communities…

IS IT A PARTITION?
OR CAN GROUPS BE OVERLAPPING?