kLog

A Language for Logical and Relational Learning with Kernels

Paolo Frasconi\textsuperscript{1}  Fabrizio Costa\textsuperscript{2}  Luc De Raedt\textsuperscript{3}  Kurt De Grave\textsuperscript{3}

\textsuperscript{1}Università di Firenze, Italy \textsuperscript{2}Universität Freiburg, Germany \textsuperscript{3}K.U. Leuven, Belgium

CoLISD.ECML/PKDD — Athens, 09.09.2011
A language/framework for kernel-based relational learning
- A language/framework for kernel-based relational learning
- Currently embedded in Prolog
A language/framework for kernel-based relational learning

Currently embedded in Prolog

Four simple concepts:

- Learning from interpretations
- Entity/relationship data modeling
- Deductive databases
- Graph kernels
Goals

- Make it possible to design and maintain complex features in a declarative fashion.
Goals

- Make it possible to design and maintain complex features in a declarative fashion
- Make it possible to define in the same framework several kinds of learning problems ranging from plain classification and regression to entity classification to (hyper)link prediction and even unsupervised learning
Goals

- Make it possible to design and maintain complex features in a **declarative** fashion
- Make it possible to define in the same framework several kinds of **learning problems** ranging from plain classification and regression to entity classification to (hyper)link prediction and even unsupervised learning
- Modularity and **separation of concerns**:  
  - Plug-in different graph kernels to create actual features  
  - Plug-in different statistical, inference, optimization techniques  
  - Simple semantics: the meaning of a kLog script only defines the learning problem and the associated features
Supervised learning: A quite general formulation

- Fit a linear potential function on some feature space:

\[ F(x, y) = w' \phi(x, y) \]

where \( x \) and \( y \) are input and output ground atoms
Supervised learning: A quite general formulation

- Fit a linear potential function on some feature space:

\[ F(x, y) = w' \phi(x, y) \]

where \( x \) and \( y \) are input and output ground atoms

- \( F(x, y) \) measures the compatibility between \( x \) and \( y \)
Supervised learning: A quite general formulation

- Fit a linear potential function on some feature space:

  \[ F(x, y) = w' \phi(x, y) \]

  where \( x \) and \( y \) are input and output ground atoms

- \( F(x, y) \) measures the compatibility between \( x \) and \( y \)

- Predictions can be obtained as

  \[ f(x) = \arg \max_y F(x, y) \]

  This “inference” step is intractable in general (depending on the structure of the interdependencies between variables)
A classic trilogy

Propositional

Naïve Bayes
A classic trilogy

Propositional

Naïve Bayes

Logistic regression
A classic trilogy

Propositional

Naïve Bayes

Logistic regression

SVM
A classic trilogy

**Propositional**

- Naïve Bayes
- Logistic regression
- SVM

**Similar features**

- Relational generative models, e.g. MLN, PRM
- Relational discriminative models, e.g. MLN, CRF
- SVM-HMM
- Max-margin relational machines, e.g. M3N
- Similar loss
### Propositional vs. Sequences

<table>
<thead>
<tr>
<th>Propositional</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>HMM</td>
</tr>
<tr>
<td>Logistic</td>
<td>Linear-chain</td>
</tr>
<tr>
<td>regression</td>
<td>CRF</td>
</tr>
<tr>
<td>SVM</td>
<td>SVM-HMM</td>
</tr>
<tr>
<td>Propositional</td>
<td>Sequences</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>HMM</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>Linear-chain CRF</td>
</tr>
<tr>
<td>SVM</td>
<td>SVM-HMM</td>
</tr>
</tbody>
</table>
A classic trilogy

<table>
<thead>
<tr>
<th>Propositional</th>
<th>Sequences</th>
<th>General relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>HMM</td>
<td>Relational generative models, e.g. MLN, PRM</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>Linear-chain CRF</td>
<td>Relational discriminative models, e.g. MLN, CRF</td>
</tr>
<tr>
<td>SVM</td>
<td>SVM-HMM</td>
<td>Max-margin relational machines, e.g. M³N</td>
</tr>
</tbody>
</table>

### Similar features

### Similar loss
kLog by example: UW-CSE
kLog by example: UW-CSE

- Classic E/R diagram
- Boxes are entities
- Diamonds are relationships
- Ovals are properties
- Underlined properties are entity identifiers (not directly used to create features)
kLog by example: UW-CSE

signature student(
    student_id::self
)::*extensional.

signature in_phase(
    student_id::student,
    phase::property
)::*extensional.

signature years_in_program(
    student_id::student,
    years::property
)::*extensional.

signature professor(
    prof_id::self
)::*extensional.

signature has_position(
    prof_id::professor,
    position::property
)::*extensional.
signature \texttt{student}(\texttt{student_id}::\texttt{self})::\texttt{extensional}.
signature \texttt{in\_phase}(\texttt{student_id}::\texttt{student}, \texttt{phase}::\texttt{property})::\texttt{extensional}.
signature \texttt{years\_in\_program}(\texttt{student_id}::\texttt{student}, \texttt{years}::\texttt{property})::\texttt{extensional}.
signature \texttt{professor}(\texttt{prof_id}::\texttt{self})::\texttt{extensional}.
signature \texttt{has\_position}(\texttt{prof_id}::\texttt{professor}, \texttt{position}::\texttt{property})::\texttt{extensional}.
signature \texttt{advised\_by}(\texttt{student_id}::\texttt{student}, \texttt{prof_id}::\texttt{professor})::\texttt{extensional}.
Data

- Data is a set of interpretations (in the pure logical sense)
- One interpretation is a set of ground facts
- Data is a set of interpretations (in the pure logical sense)
- One interpretation is a set of ground facts
- Interpretations are independent
- In UW-CSE there is one interpretation for every research group (AI, Graphics, etc.)
Data

- Data is a set of interpretations (in the pure logical sense)
- One interpretation is a set of ground facts
- Interpretations are independent
- In UW-CSE there is one interpretation for every research group (AI, Graphics, etc.)
- Interpretations are *invisible* at the level of kLog scripts (since they are independent, you are not allowed to create interactions)
Data

- Data is a set of interpretations (in the pure logical sense)
- One interpretation is a set of ground facts
- Interpretations are independent
- In UW-CSE there is one interpretation for every research group (AI, Graphics, etc.)
- Interpretations are *invisible* at the level of kLog scripts (since they are independent, you are not allowed to create interactions)
- The keyword *extensional* declares that all true ground facts for the given predicate are actually given as data (under the usual CWA).
Example of interpretation

interpretation(ai, student(person311)).
interpretation(ai, student(person14)).
...
interpretation(ai, professor(person7)).
interpretation(ai, professor(person185)).
...
interpretation(ai, has_position(person292, faculty_affiliate)).
interpretation(ai, has_position(person79, faculty)).
...
interpretation(ai, in_phase(person139, post_quals)).
interpretation(ai, in_phase(person333, pre_quals)).
...
interpretation(ai, years_in_program(person382, year_3)).
interpretation(ai, years_in_program(person333, year_2)).
...
interpretation(ai, advised_by(person265, person168)).
interpretation(ai, advised_by(person352, person415)).
...
Interpretations may contain more relations

\begin{verbatim}
interpretation(ai,publication(title25,person284)).
interpretation(ai,publication(title284,person14)).
interpretation(ai,publication(title110,person14)).
...
interpretation(ai,taught_by(course12,person211,autumn_0001)).
interpretation(ai,taught_by(course123,person150,autumn_0001)).
interpretation(ai,taught_by(course44,person293,winter_0001)).
...
interpretation(ai,ta(course44,person193,winter_0304)).
interpretation(ai,ta(course128,person271,winter_0304)).
interpretation(ai,ta(course128,person392,winter_0304)).
\end{verbatim}
signature on_same_paper(
    student_id::student,
    prof_id::professor
)::intensional.
signature **on_same_paper**(
    student_id::student,
    prof_id::professor
)::intensional.

on_same_paper(S,P) :-
    student(S), professor(P),
    publication(Pub, S),
    publication(Pub,P).
Intensional signatures

signature on_same_paper(  
    student_id::student,  
    prof_id::professor  
)::intensional.

on_same_paper(S,P) :-  
    student(S), professor(P),  
    publication(Pub, S),  
    publication(Pub,P).

signature on_same_course(  
    student_id::student,  
    prof_id::professor  
)::intensional.

on_same_course(S,P) :-  
    professor(P), student(S),  
    ta(Course,S,Term),  
    taught_by(Course,P,Term).
Graphicalization

- Second semantic layer: map each interpretation into a simple graph (not a hypergraph).
Graphicalization

- Second semantic layer: map each interpretation into a simple graph (not a hypergraph).
- The mapping is lossless:
  - There is one vertex for every ground fact, labeled by the fact itself
Graphicalization

- Second semantic layer: map each interpretation into a simple graph (not a hypergraph).
- The mapping is lossless:
  - There is one vertex for every ground fact, labeled by the fact itself
  - One (undirected) edge between $u$ and $v$ iff:
    1) $u$ is an entity-fact
    2) $v$ is a relationship-fact
    3) $v$ refers to the identifier in $u$
  (so the graph is bipartite)
Second semantic layer: map each interpretation into a simple graph (not a hypergraph).

The mapping is lossless:

- There is one vertex for every ground fact, labeled by the fact itself.
- One (undirected) edge between $u$ and $v$ iff:
  1) $u$ is an entity-fact
  2) $v$ is a relationship-fact
  3) $v$ refers to the identifier in $u$

(so the graph is bipartite)

Graphicalization is essentially grounding the E/R diagram.
Graphicalization in UW-CSE
Graphicalization in UW-CSE
A supervised learning job is defined by marking some signature(s) as target (aka query, aka output)
Supervised learning jobs

- A **supervised learning job** is defined by marking some signature(s) as target (aka query, aka output)
- This means to kLog: **learn a statistical model capable of predicting tuples for the corresponding relation(s)**
Supervised learning jobs

- A **supervised learning job** is defined by marking some signature(s) as target (aka query, aka output).
- This means to kLog: learn a statistical model capable of predicting tuples for the corresponding relation(s).
- The focus of a job is on **what**, not on **how** (different statistical models can solve the same job – maybe with different performances).
Single task binary classification

- Job type obtained when we specify a single target signature with **no properties**.

- Example:
  
  \[
  \text{signature} \ \text{advised\_by} (\\ \\
  s::\text{student,}\\
  p::\text{professor} \\
  )::\text{extensional}.
  \]

- In this example \( y \) consists of all ground atoms of the relation \( \text{advised\_by} \)

- Each tuple of identifiers in the target relation, e.g. a \( (\text{student,professor}) \) pair, is called a **case** (or **instance**)

- If the target signature has entity sets \( E_1, \cdots, E_k \), the set of cases is their the Cartesian product
Other jobs types

- Single-task multiclass classification: if the target signature contains a categorical property

  signature pageclass(
    url::webpage,
    category::property
  )::extensional.
Other jobs types

- Single-task multiclass classification: if the target signature contains a categorical property
  
  signature pageclass(
    url::webpage,
    category::property
  )::extensional.

- Single-task regression: if the target signature contains a numerical property
  
  signature intelligence(
    student_it::student,
    qi::property
  )::extensional.
Other jobs types

- Single-task multiclass classification: if the target signature contains a categorical property

  signature `pageclass`
  ```
  url::webpage,
  category::property
  ```
  ::extensional.

- Single-task regression: if the target signature contains a numerical property

  signature `intelligence`
  ```
  student_it::student,
  qi::property
  ```
  ::extensional.

- Multi-task: if there are several target signatures or a single target signature with several properties.
### Overview of job types

<table>
<thead>
<tr>
<th># of properties</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Binary classification of interpretations</td>
<td>Binary classification of entities</td>
<td>Link prediction</td>
</tr>
<tr>
<td>0</td>
<td>Multiclass / regression on interpretations</td>
<td>Multiclass / regression on entities</td>
<td>Attributed link prediction</td>
</tr>
<tr>
<td>1</td>
<td>Multitask on interpretations</td>
<td>Multitask predictions on entities</td>
<td>Multitask attributed link prediction</td>
</tr>
<tr>
<td>&gt;1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The next semantic layer of kLog concerns feature vectors. Here we use graph kernels.
The next semantic layer of kLog concerns feature vectors. Here we use graph kernels. In principle any graph kernel can be used and the architecture of kLog is open enough to allow different feature generators to be plugged in.
The next semantic layer of kLog concerns feature vectors.

Here we use graph kernels.

In principle any graph kernel can be used and the architecture of kLog is open enough to allow different feature generators to be plugged in.

In practice we have implemented a generalization of the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK, Costa & De Grave, ICML 2010).
NSPDK: Core ideas

- Decompose graphs into subgraphs rooted at certain designated vertices called kernel-points (KP)
- Consider all pairs of such subgraphs
- Count the number of common pairs between two graphs
- Use hashing to approximate subgraph isomorphism
The NP-relation

Given graph $G = (V, E)$ and $u, v \in V$, let

$$
\delta_{u,v} = \begin{cases} 
  \text{SPD}(u, v) & \text{if } u, v \in \text{KP} \\
  \infty & \text{otherwise}
\end{cases}
$$

where SPD is for shortest-path-distance.

Let $N_{r}(G)$ denote the subgraph of $G$ induced by all $x \in V$ s.t. $\text{SPD}(x, v) \leq r$.

The neighborhood-pair (NP) relation is the set of triplets $R_{r,d} = \{(A, B, G) : A \sim = N_{v}(G), B \sim = N_{u}(G), \delta_{u,v} = d\}$ where $\sim = \text{is graph isomorphism}$.
The NP-relation

- Given graph $G = (V, E)$ and $u, v \in V$, let

$$
\delta_{u, v} = \begin{cases} 
\text{SPD}(u, v) & \text{if } u, v \in KP \\
\infty & \text{otherwise}
\end{cases}
$$

where SPD is for shortest-path-distance

- Let $\mathcal{N}_r^v(G)$ denote the subgraph of $G$ induced by all $x \in V$ s.t. $\text{SPD}(x, v) \leq r$
The NP-relation

- Given graph $G = (V, E)$ and $u, v \in V$, let

\[
\delta_{u,v} = \begin{cases} 
\text{SPD}(u, v) & \text{if } u, v \in KP \\
\infty & \text{otherwise}
\end{cases}
\]

where SPD is for shortest-path-distance

- Let $\mathcal{N}_r^v(G)$ denote the subgraph of $G$ induced by all $x \in V$ s.t. $\text{SPD}(x, v) \leq r$

- The neighborhood-pair (NP) relation is the set of triplets

\[
R_{r,d} = \{(A, B, G) : A \cong \mathcal{N}_r^v(G), B \cong \mathcal{N}_r^u(G), \delta_{u,v} = d\}
\]

where $\cong$ is graph isomorphism
NP’s for fixed $d = 6$
NP’s for fixed $d = 6$
NP’s for fixed $d = 6$

$r = 0$
NP’s for fixed $d = 6$
NP’s for fixed $d = 6$

$r = 0$

$r = 1$

$r = 2$
NP’s for fixed $d = 6$

$r = 0$

$r = 1$

$r = 2$

$r = 3$
NP's for fixed $r = 2$
NP’s for fixed $r = 2$
NP’s for fixed $r = 2$
NP's for fixed $r = 2$
NP’s for fixed $r = 2$

$d = 0$

$d = 1$

$d = 2$

$d = 3$
Definition of the NSPDK

- $\kappa_{r,d}$ counts common NP’s between two graphs:

$$
\kappa_{r,d}(G, G') = \sum \delta(A, A')\delta(B, B')
$$

$$(A, B) \in R_{r,d}^{-1}(G)$$

$$(A', B') \in R_{r,d}^{-1}(G')$$

where the “inverse” of a relation $R \subset A \times B \times C$ is the multiset $R^{-1}(c) = \{(a, b) : R(a, b, c)\}$
Definition of the NSPDK

\( \kappa_{r,d} \) counts common NP’s between two graphs:

\[
\kappa_{r,d}(G, G') = \sum_{(A, B) \in R^{-1}_{r,d}(G)} \delta(A, A') \delta(B, B')
\]

where the “inverse” of a relation \( R \subset A \times B \times C \) is the multiset \( R^{-1}(c) = \{(a, b) : R(a, b, c)\} \)

Overall kernel:

\[
K(G, G') = \sum_{r=0}^{R} \sum_{d=0}^{D} \kappa_{r,d}(G, G').
\]
Soft matches

- The hard-match might produce very “rare” features depending on the structure of the graph
Soft matches

- The hard-match might produce very “rare” features depending on the structure of the graph
- Bad if vertices have high degree
Soft matches

- The hard-match might produce very “rare” features depending on the structure of the graph
- Bad if vertices have high degree
- Soft match kernel:

\[
k_{r,d}(G, G') = \sum_{(A, B) \in R_{r,d}^{-1}(G)} \sum_{v \in V(A) \cup V(B)} \sum_{(A', B') \in R_{r,d}^{-1}(G')} \sum_{v' \in V(A') \cup V(B')} \delta(L(v), L(v'))
\]

where \(L(v)\) is the label of vertex \(v\)
signature `atm`
(`atom_id::self,
  element::property)::extensional.

signature `bnd`
(`atom_1@b::atm,
  atom_1@b::atm,
  type::property)::extensional.

signature `fgroup`
(`fgroup_id::self,
  group_type::property)::intensional.

signature `fgmember`
(`fg::fgroup,
  atom::atm)::intensional.

signature `fg_linked`
(`fg::fgroup,
  alichain::fgroup,
  saturation::property)::intensional.

signature `mutagenic`::extensional.
:- use_module('klog').

begin_domain.

signature \texttt{atm}(atom\_id::self,element::property)::extensional.

signature \texttt{activity}(act::property)::extensional.

\texttt{kernel\_points}([\texttt{atm}, \texttt{fgroup}]).

end_domain.

experiment :-

new\_feature\_generator(\texttt{my\_fg}, \texttt{nspdk}),
set\_klog\_flag(\texttt{my\_fg}, \texttt{radius}, 4),
set\_klog\_flag(\texttt{my\_fg}, \texttt{distance}, 8),
attach(\texttt{bursi\_ext}),
neglect\_model(\texttt{my\_model}, \texttt{libsvm\_c\_svc}),
set\_klog\_flag(\texttt{my\_model}, \texttt{c}, 0.5),
stratified\_kfold(\texttt{mutagenic}, 10, \texttt{my\_model}, \texttt{my\_fg}, \texttt{muta\_stratum}).
Small molecules (regression/classification)

<table>
<thead>
<tr>
<th>Biodegradability</th>
<th>Setting</th>
<th>RMSE</th>
<th>SCC</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Functional groups</td>
<td>1.07 ± 0.01</td>
<td>0.54 ± 0.01</td>
<td>14.0 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>Atom bonds</td>
<td>1.13 ± 0.01</td>
<td>0.48 ± 0.01</td>
<td>14.5 ± 0.1</td>
</tr>
</tbody>
</table>
Small molecules (regression/classification)

<table>
<thead>
<tr>
<th>Biodegradability</th>
<th>RMSE</th>
<th>SCC</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional groups</td>
<td>1.07 ± 0.01</td>
<td>0.54 ± 0.01</td>
<td>14.0 ± 0.1</td>
</tr>
<tr>
<td>Atom bonds</td>
<td>1.13 ± 0.01</td>
<td>0.48 ± 0.01</td>
<td>14.5 ± 0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bursi</th>
<th>AUROC</th>
<th>F1</th>
<th>Error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional groups</td>
<td>0.91 ± 0.01</td>
<td>86.78 ± 1.05</td>
<td>14.7 ± 1.5</td>
</tr>
<tr>
<td>Atom bonds</td>
<td>0.90 ± 0.01</td>
<td>85.21 ± 1.37</td>
<td>16.9 ± 1.5</td>
</tr>
</tbody>
</table>
Small molecules (regression/classification)

### Biodegradability

<table>
<thead>
<tr>
<th>Setting</th>
<th>RMSE</th>
<th>SCC</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional groups</td>
<td>1.07 ± 0.01</td>
<td>0.54 ± 0.01</td>
<td>14.0 ± 0.1</td>
</tr>
<tr>
<td>Atom bonds</td>
<td>1.13 ± 0.01</td>
<td>0.48 ± 0.01</td>
<td>14.5 ± 0.1</td>
</tr>
</tbody>
</table>

### Bursi

<table>
<thead>
<tr>
<th>Setting</th>
<th>AUROC</th>
<th>F1</th>
<th>Error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional groups</td>
<td>0.91 ± 0.01</td>
<td>86.78 ± 1.05</td>
<td>14.7 ± 1.5</td>
</tr>
<tr>
<td>Atom bonds</td>
<td>0.90 ± 0.01</td>
<td>85.21 ± 1.37</td>
<td>16.9 ± 1.5</td>
</tr>
</tbody>
</table>

With functional groups

Atom bond
Is the NPDK kernel general enough?

What about multiple interdependent predictions within the same interpretation?
Is the NPDK kernel general enough?

- What about multiple interdependent predictions within the same interpretation?
- Two possible answers:
Is the NPDK kernel general enough?

- What about multiple interdependent predictions within the same interpretation?
- Two possible answers:
  - The graph kernel creates joint features $\phi(x,y)$ so go for collective (structured-output) prediction i.e. $\text{argmax } w'\phi(x,y)$ over an exponential number of assignments to $y$. This is especially challenging because intensional predicates need to be re-evaluated for different assignments.
Is the NPDK kernel general enough?

- What about multiple interdependent predictions within the same interpretation?
- Two possible answers:
  - The graph kernel creates joint features $\phi(x, y)$ so go for collective (structured-output) prediction i.e. $\operatorname{argmax} w'\phi(x, y)$ over an exponential number of assignments to $y$. This is especially challenging because intensional predicates need to be re-evaluated for different assignments.
  - Project the collective problem into several i.i.d. views
Let $c \in y$ be a case
Let $c \in y$ be a case

The viewpoint of $c$, $W_c$, is the set of vertices that touch $c$ in the graph.
Let $c \in y$ be a case

The viewpoint of $c$, $W_c$, is the set of vertices that touch $c$ in the graph

Consider the mutilated graph $G_c$ where all vertices in $y$ except $c$ are removed
Let $c \in y$ be a case.

The viewpoint of $c$, $W_c$, is the set of vertices that touch $c$ in the graph.

Consider the mutilated graph $G_c$ where all vertices in $y$ except $c$ are removed.

Define a kernel $\hat{\kappa}$ on mutilated graphs: like NSPDK but with the restriction that the first endpoint must be in $W_c$.

$$\hat{R}_{r,d} = \{(A, B, G_c) : A \cong \mathcal{N}^v_r, B \cong \mathcal{N}^u_r, v \in W_c, \delta_{u,v} = d\}$$
We get in this way a kernel “centered” around case $c$:

$$
\hat{K}(G_c, G'_{c'}) = \sum_{r,d} \sum_{A, B \in \hat{R}_{r,d}^{-1}(G_c)} \delta(A, A') \delta(B, B')
$$

Finally let

$$
K(G, G') = \sum_{c \in y, c' \in y'} \hat{K}(G_c, G'_{c'})
$$

This kernel corresponds to the potential

$$
F(x,y) = w' \sum c \hat{\varphi}(x,c)
$$

which is clearly maximized by maximizing, independently, all sub-potentials $w' \hat{\varphi}(x,c)$ with respect to $c$. 
We get in this way a kernel “centered” around case $c$:

$$\hat{K}(G_c, G'_c) = \sum_{r, d} \sum_{A, B \in \hat{R}_{r,d}^{-1}(G_c)} \delta(A, A') \delta(B, B')$$

Finally let

$$K(G, G') = \sum_{c \in y, c' \in y'} \hat{K}(G_c, G'_c)$$
View points and i.i.d. views (non-collective)

- We get in this way a kernel “centered” around case $c$:

$$
\hat{K}(G_c, G'_{c'}) = \sum_{r,d} \sum_{A, B \in \hat{R}_{r,d}(G_c)} \delta(A, A') \delta(B, B')
$$

- Finally let

$$
K(G, G') = \sum_{c \in y, c' \in y'} \hat{K}(G_c, G'_{c'})
$$

- This kernel corresponds to the potential

$$
F(x, y) = w' \sum_c \hat{\phi}(x, c)
$$

which is clearly maximized by maximizing, independently, all sub-potentials $w' \hat{\phi}(x, c)$ with respect to $c$. 
UW-CSE: All information

![Graphs showing precision and recall for different categories: All (all), ai (all), languages (all), graphics (all), systems (all), theory (all).]
Alternative setting: we only know about persons without knowing whether they are professors or students.
Stacking

- Alternative setting: we only know about persons without knowing whether they are professors or students.
- Without collective inference we also lack the ability of joint inference on the predicates student, professor, and advised_by.
Alternative setting: we only know about persons without knowing whether they are professors or students.

Without collective inference we also lack the ability of joint inference on the predicates student, professor, and advised_by.

In kLog it is easy to define a stacked (pipelined) prediction method:
Stacking

- Alternative setting: we only know about persons without knowing whether they are professors or students.
- Without collective inference we also lack the ability of joint inference on the predicates `student`, `professor`, and `advised_by`.
- In kLog it is easy to define a stacked (pipelined) prediction method:
  - First, learn to discriminate between professors and students.
Stacking

- Alternative setting: we only know about persons without knowing whether they are professors or students
- Without collective inference we also lack the ability of joint inference on the predicates student, professor, and advised_by
- In kLog it is easy to define a stacked (pipelined) prediction method:
  - First, learn to discriminate between professors and students
  - Assert induced groundings (predicted in cross-validation mode)
Alternative setting: we only know about persons without knowing whether they are professors or students.

Without collective inference we also lack the ability of joint inference on the predicates `student`, `professor`, and `advised_by`.

In kLog it is easy to define a stacked (pipelined) prediction method:

- First, learn to discriminate between professors and students.
- Assert *induced* groundings (predicted in cross-validation mode).
- Learn the binary relation taking saved groundings as additional data.
UW-CSE: Partial information

All (partial)

Precision vs Recall

ai (partial)

languages (partial)

systems (partial)

theory (partial)
signature \texttt{csNNN\_in\_url}\(\texttt{(pageid::page)>::intensional.}\)

\texttt{csNNN\_in\_url(Url) :- page(Url),}
\texttt{atom\_codes(Url,CUrl),}
\texttt{regexp("cs(e*)[0-9]+",CUrl,\[],[\_Match]).}\n
signature \texttt{category}\(\texttt{(page\_id::page,}
\texttt{cat::property)>::extensional.}\)
# WebKB: results

## WebKB: results

- **# Cases:** 1039
- **Case error rate:** 12.42%
- **Interpretation error rate:** 100.00%
- **Contingency table:** (rows are predictions)

<table>
<thead>
<tr>
<th></th>
<th>research</th>
<th>faculty</th>
<th>course</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>research</td>
<td>59</td>
<td>11</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>faculty</td>
<td>9</td>
<td>125</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>course</td>
<td>0</td>
<td>0</td>
<td>233</td>
<td>0</td>
</tr>
<tr>
<td>student</td>
<td>16</td>
<td>17</td>
<td>5</td>
<td>493</td>
</tr>
</tbody>
</table>

- **Average p, r, f1:** 0.89 0.88 0.88
signature `blockbuster(film_id::film)::intensional.
blockbuster(M) :-
    opening_weekend(M,Receipts),
    Receipts > 2000000.
signature **blockbuster**(film_id::film)::intensional.

blockbuster(M) :-
    opening_weekend(M,Receipts),
    Receipts > 2000000.

signature **individual**(individual_id::self)::intensional.

individual(P) :-
    person(P,_Name), active_enough(P).

has_active_role(P,M) :- acted_in(P,M).

has_active_role(P,M) :- directed(P,M).

has_active_role(P,M) :- produced(P,M).

active_enough(P) :-
    setof(M,has_active_role(P,M),Ms), length(Ms,N), N>2.
signature **blockbuster**(film_id::film)::intensional.
blockbuster(M) :-
  opening_weekend(M,Receipts),
  Receipts > 2000000.

signature **individual**(individual_id::self)::intensional.
individual(P) :-
  person(P,_Name), active_enough(P).

has_active_role(P,M) :- acted_in(P,M).
has_active_role(P,M) :- directed(P,M).
has_active_role(P,M) :- produced(P,M).
active_enough(P) :-
  setof(M,has_active_role(P,M),Ms), length(Ms,N), N>2.

signature **in_blockbuster**(individual_id::individual)::intensional.
in_blockbuster(P) :-
  has_active_role(P,M),
  blockbuster(M).

signature **bb_cast_len**(film_id::film,n::property)::intensional.
bb_cast_len(M,N) :-
  setof(Actor, (acted_in(Actor,M), in_blockbuster(Actor)), Set),
  length(Set,N).
In a data set like IMDb there is one single interpretation.

How to split training and test data?
Slicing

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>

Train on \{1992,1993\}

Test on \{1995,1996\}

= invisible = visible = targets = to be predicted
<table>
<thead>
<tr>
<th>Year</th>
<th>movies</th>
<th>facts</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>74</td>
<td>2483</td>
<td>-</td>
</tr>
<tr>
<td>1996</td>
<td>223</td>
<td>6406</td>
<td>-</td>
</tr>
<tr>
<td>1997</td>
<td>311</td>
<td>8031</td>
<td>0.85</td>
</tr>
<tr>
<td>1998</td>
<td>332</td>
<td>7822</td>
<td>0.92</td>
</tr>
<tr>
<td>1999</td>
<td>348</td>
<td>7842</td>
<td>0.88</td>
</tr>
<tr>
<td>2000</td>
<td>381</td>
<td>8531</td>
<td>0.95</td>
</tr>
<tr>
<td>2001</td>
<td>363</td>
<td>8443</td>
<td>0.94</td>
</tr>
<tr>
<td>2002</td>
<td>370</td>
<td>8691</td>
<td>0.93</td>
</tr>
<tr>
<td>2003</td>
<td>343</td>
<td>7626</td>
<td>0.94</td>
</tr>
<tr>
<td>2004</td>
<td>371</td>
<td>8850</td>
<td>0.94</td>
</tr>
<tr>
<td>2005</td>
<td>388</td>
<td>9093</td>
<td>0.92</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td>0.92 ± 0.03</td>
</tr>
</tbody>
</table>
Conclusions

- **Highlights**
  - Complex feature generation thanks to graph kernels
  - Easy but powerful declaration of jobs
  - Most statistical learners pluggable-in (even as external programs)

- To be done (or to be tried)
  - Collective classification (e.g. label propagation, MaxWalkSAT, ...)

- Constraints

- Applications (work in progress)
  - Information extraction from text, e.g. spatial role labeling (Kordjamshidi et al. 2011)
  - Hedge cue detection, e.g. recognition of weasel sentences (Verbeke et al. 2011)
  - Vision, e.g. image segmentation/labeling (Antanas et al. 2011)
Conclusions

- **Highlights**
  - Complex feature generation thanks to graph kernels
  - Easy but powerful declaration of jobs
  - Most statistical learners pluggable-in (even as external programs)

- **To be done (or to be tried)**
  - Collective classification (e.g. label propagation, MaxWalkSAT, ...)
  - Constraints
  - Multitask regularization

Applications (work in progress)
- Information extraction from text, e.g. spatial role labeling (Kordjamshidi et al. 2011)
- Hedge cue detection, e.g. recognition of weasel sentences (Verbeke et al. 2011)
- Vision, e.g. image segmentation/labeling (Antanas et al. 2011)
Conclusions

- Highlights
  - Complex feature generation thanks to graph kernels
  - Easy but powerful declaration of jobs
  - Most statistical learners pluggable-in (even as external programs)

- To be done (or to be tried)
  - Collective classification (e.g. label propagation, MaxWalkSAT, ...)
  - Constraints
  - Multitask regularization

- Applications (work in progress)
  - Information extraction from text, e.g. spatial role labeling (Kordjamshidi et al. 2011)
  - Hedge cue detection, e.g. recognition of weasel sentences (Verbeke et al. 2011)
  - Vision, e.g. image segmentation/labeling (Antanas et al. 2011)