Information extraction

8. Consolidation

Simon Razniewski
Winter semester 2019/20

Announcements

- Results assignment 7 online

Lab 07 Ranking	
2576611	0.816
2576861	0.815
2576748	0.797
2571663	0.78
2561347	0.773
2572758	0.721
2570975	0.659
2576796	0.646
2576572	0.62
2565094	0.611
2550309	0.541
2581455	0.487
2576770	0.44
2581370	0.436
2549786	0.393
2576612	0.326
2550421	0.237
2553344	0.171
2572706	0.162
2568101	0.108

The ideas behind this code :

1. For sentences that have S-P-O format:
   - Take the subject by dependency of "nsubj" or "nsubjpass" that have "head" as "verb".
   - Use the token.head as the verb
   - Put the remaining words on the object
2. Passive sentences are identified by the structure of '-object- verb -agent- -subject-': (line 78-105)
   i.e. Annualized interest rates on certain investments as reported by the Federal Reserve Board
   - object : Annualized interest rates on certain investments
   - verb : reported
   - agent : by
   - subject: the Federal Reserve Board
3. For sentences that have "verb" conjunction: (line 107-109)
   i.e. Large cross - border deals numbered 51 and totaled $ 17.1 billion in the second quarter , the firm added .
   - subject of "totaled" is taken from the subject of conjuction "verb", in this case "numbered"
4. For sentences that have subject and verb at last: (line 111-119)
   i.e. Large cross - border deals numbered 51 and totaled $ 17.1 billion in the second quarter , the firm added .
   - Find the whole family of "verb's child". In this case, "numbered" is the child of "added", so we extract anything that connected to "numbered".

   The similar case also used to extract the subject (line 120-122).
5. What does "find_full_subj(word)" do? (line 149-160)
   This function give all the family of the given word and returning the ordered words by the index.
   This is used in finding "complete subject" and "complete object" in 4th case.

Interesting findings :

1. By only changing the "object" of baseline to the whole sentence after the "verb" can give us 0.69 F1 score with: 0.92 precision and 0.54 recall
   i.e. Large cross - border deals numbered 51 and totaled $ 17.1 billion in the second quarter , the firm added .
   subject : deals
   predicate : numbered
   object : 51 and totaled $ 17.1 billion in the second quarter , the firm added .

Outline

1. Motivation
2. Consolidation
   - MaxSAT
   - Probabilistic Soft Logic
   - Google Knowledge Vault
3. Pattern Learning
   - Association rule mining
   - Matrix completion
   - Knowledge graph embeddings

Consolidation: Context

- Ambiguity at all stages of the pipeline
  - Entity recognition, coreference, entity disambiguation, relation extraction, …

  → All extractions are only probabilistically correct
- Solution: Redundancy
  - Extract from large corpora like web
  - Try to spot information multiple times
    - Multiple sentences, documents, patterns
    - Unlike e.g. a redundancy-free WP biography in isolation
- Solution or curse?
  - Redundancy enables competitive and contradictory information

Contradictory information

- 30x R(a, b)
- 25x R(a, c)
- 17x S(d, e)
- 17x S(e, f)
- 3x S(f, d)

- Contradictory?
  - R = place of birth
  - S = childOf

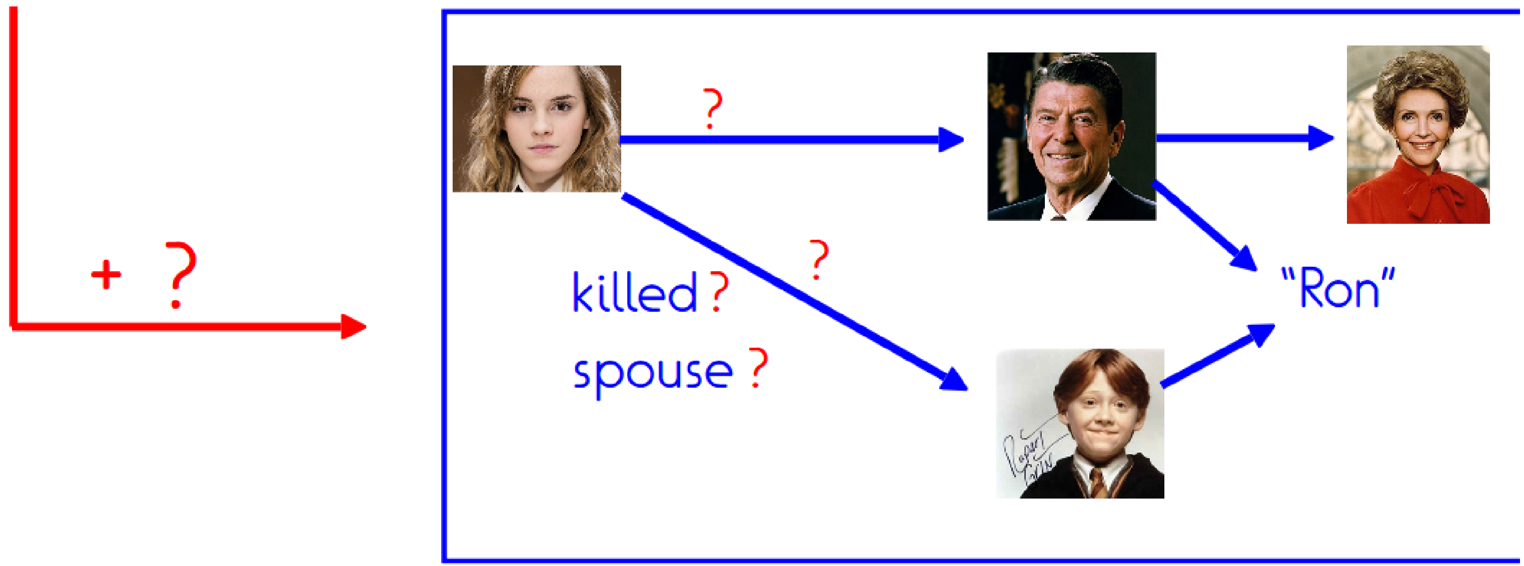
→ Contradictions surface only with world knowledge
  → Expert input or constraint/pattern mining needed

Outline

1. Motivation
2. Consolidation
    - MaxSAT
    - Probabilistic Soft Logic
    - Google Knowledge Vault
3. Pattern Learning
    - Association rule mining
    - Matrix completion
    - Knowledge graph embeddings

MaxSAT for IE

"Hermione is married to Ron"

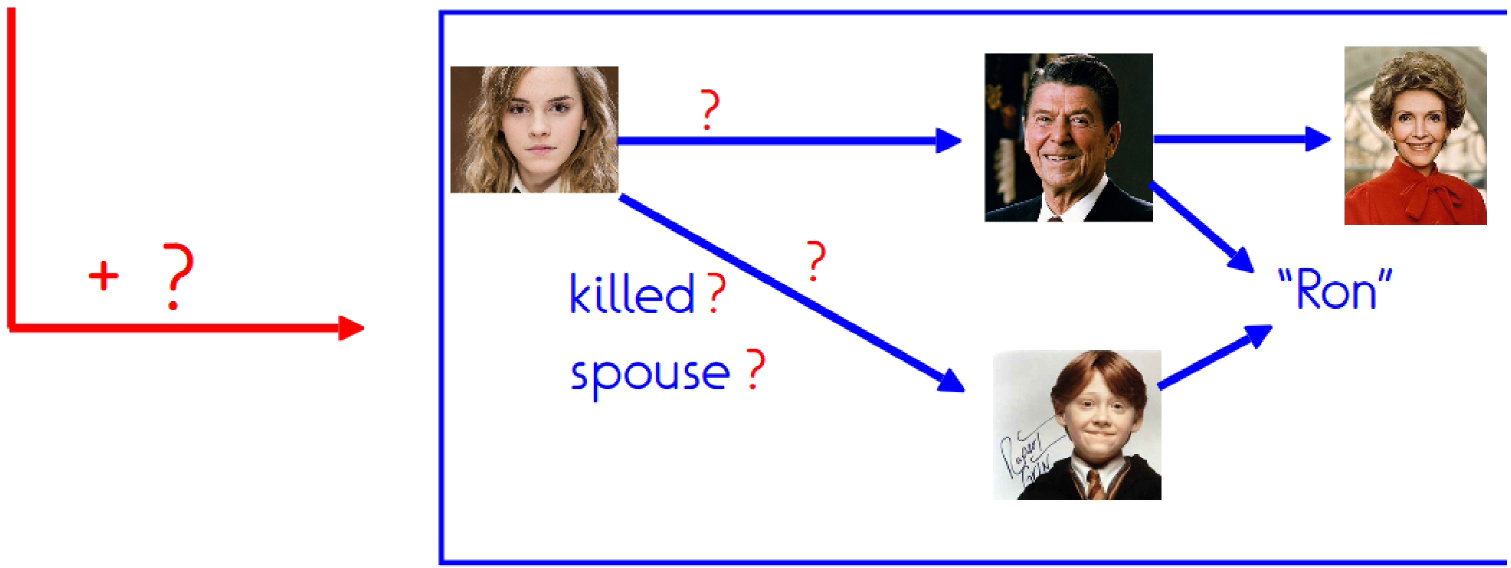


IE faces at least 3 problems:

- Understand patterns ("X is married to Y" = killed(X,Y)?)
- Disambiguate entities ("Ron" = Ronald Reagan?)
- Resolve inconsistencies (Reagan married to 2 women?)

MaxSAT for IE

“Hermione is married to Ron”



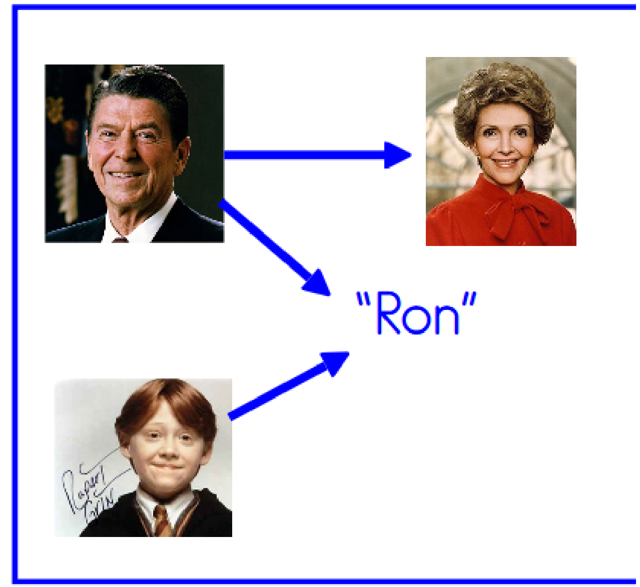
- Disambiguation avoids inconsistency
- Pattern helps disambiguation
- Consistency helps finding patterns

=> Solve all 3 problems together!

Idea: Solve all problems together

"Hermione is married to Ron"

transform everything to logical formulas



$$A \wedge B \Rightarrow C$$

Find best conclusion

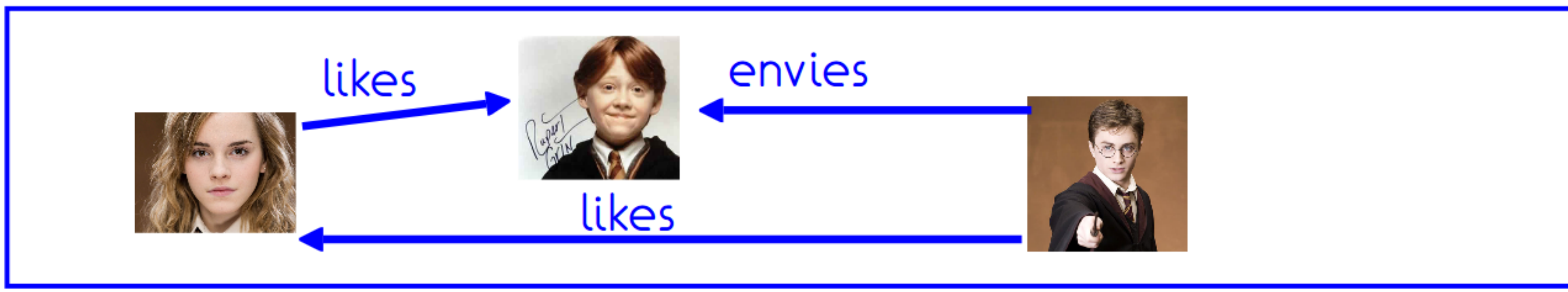
$$hasSpouse(Hermione, RonWeasley)$$

Refresh: Atoms and KBs

An positive literal holds ("is true") in a KB, if it appears in the KB.

A negative literal $\neg A$ holds in a KB if A does not hold.

A conjunction $A \wedge B \wedge \ldots \wedge Z$ holds in a KB, if all of its elements hold.



$likes(Hermione, Ron)$?

$\neg\, envies(Ron, Harry)$?

$\neg\, envies(Ron, Harry) \wedge likes(Harry, Hermione)$

$envies(Harry, Ron) \wedge likes(Hermione, Ron) \wedge likes(Harry, Elvis)$

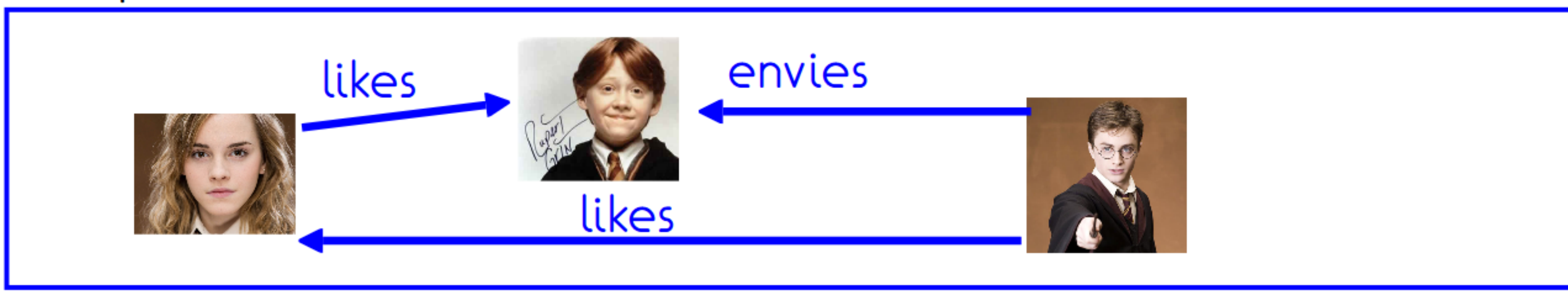
Refresh: Implications

An positive literal holds ("is true") in a KB, if it appears in the KB.

A negative literal $\neg A$ holds in a KB if A does not hold.

A conjunction $A \wedge B \wedge ... \wedge Z$ holds in a KB, if all of its elements hold.

An implication $\vec{B} \Rightarrow H$ holds in a KB if $\vec{B}$ does not hold or H holds.



$$likes(Hermione, Ron) \Rightarrow likes(Harry,Ron)$$

$$likes(Harry, Ron) \Rightarrow hasSpouse(Harry,Hermione)$$

$$\neg\, likes(Hermione, Harry) \Rightarrow envies(Harry,Ron)$$

$$\neg\, likes(Hermione, Harry) \Rightarrow hasSpouse(Harry,Ron)$$

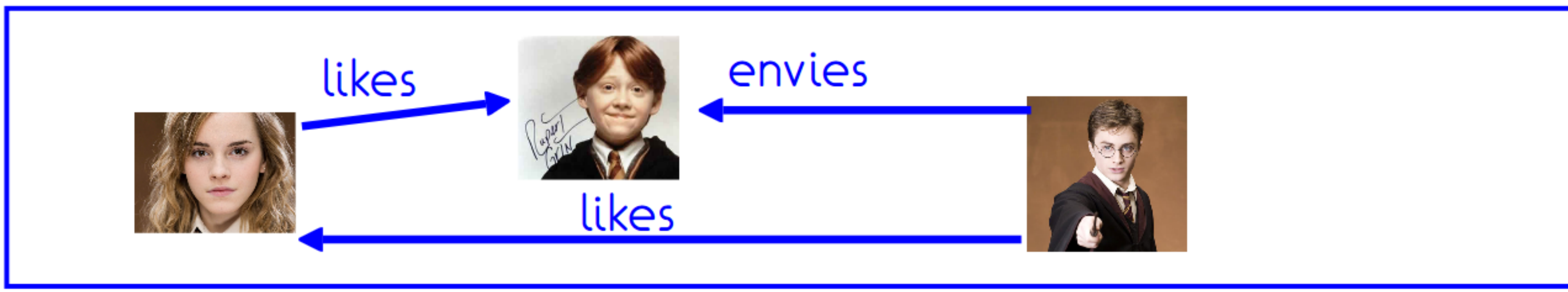
$$likes(Herm., Ron) \wedge \neg\, envies(Harry,Ron) \Rightarrow ffe(Harry,Ron)$$

Def: Rules, Disjunctions, Clauses

An implication (also: rule) $B_1 \wedge \ldots \wedge B_n \Rightarrow H$

is equivalent to a disjunction $\neg B_1 \vee \ldots \vee \neg B_n \vee H$

which we also write as a clause $\{\neg B_1, \ldots, \neg B_n, H\}$.



$$likes(Hermione, Ron) \Rightarrow likes(Harry, Ron)$$

is equivalent to

$$\neg\, likes(Hermione, Ron) \vee likes(Harry, Ron)$$

is equivalent to

$$\{\neg\, likes(Hermione, Ron), likes(Harry, Ron)\}$$

"at least one of these has to hold"

Refresh:Universally quantified formula

A universally quantified formula holds in a KB, if all of its instantiations hold.

$$hasSpouse(x, y)$$

$$hasSpouse(x, y) \Rightarrow hasSpouse(y, x)$$

$$hasSpouse(Elvis, y) \Rightarrow hasSpouse(y, Elvis)$$

$$hasSpouse(Ron, y) \Rightarrow hasSpouse(y, Ron)$$

Def: Weighted Rule

A weighted rule is a rule with an associated real-valued weight.

$$hasSpouse(Elvis,Priscilla) \Rightarrow hasSpouse(Priscilla, Elvis)[3.14]$$

A weighted rule can also be seen as a weighted disjunction...

$$\neg\, hasSpouse(Elvis,Priscilla) \vee hasSpouse(Priscilla, Elvis)[3.14]$$

...or a weighted clause.

$$\{\neg\, hasSpouse(Elvis,Priscilla), hasSpouse(Priscilla, Elvis)\}\ [3.14]$$

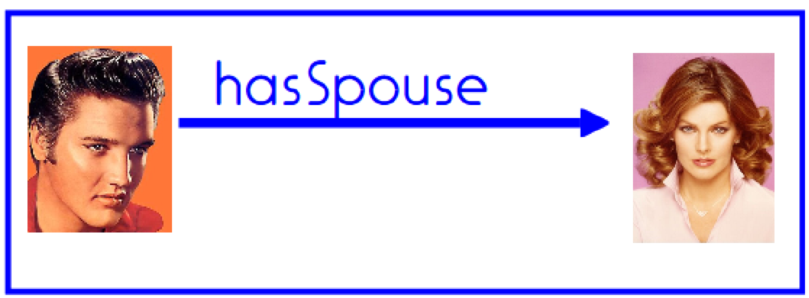
Def: Weight of a KB

Given a set of atoms (= possible world, KB) and a set of instantiated rules with weights, the weight of the KB is the sum of the weights of all rules that hold in the KB.

$$hasSpouse(Elvis, Priscilla) \Rightarrow hasSpouse(Priscilla, Elvis) [3]$$

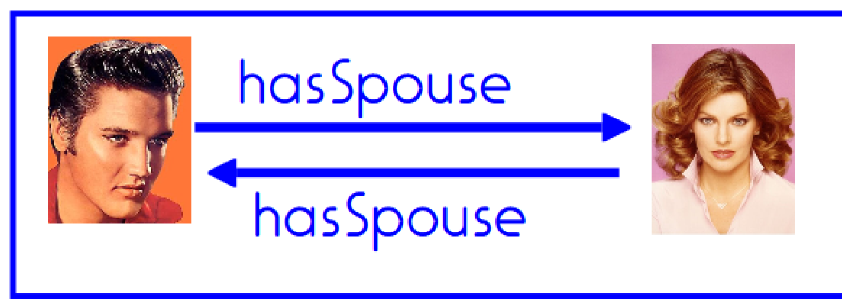
$$hasSpouse(cat, dog) \Rightarrow hasSpouse(dog, cat) [2]$$

KB1



Weight: 2

KB2



Weight: 5

Def: Weighted MAX SAT

Given a set of instantiated rules with weights, weighted MAX SAT is the problem of finding the KB with the highest weight. If there are several, find the one with the least number of atoms.

$$is(Ron, immature)\ [10]$$

$$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)\ [3]$$

$$type(Hermione, sorceress)\ [4]$$

Best world:

$$is(Ron, immature)$$
$$type(Hermione, sorceress)$$
$$likes(Hermione, Ron)$$

weight: 17

Def: Exhaustive search

Exhaustive search is an algorithm for the Weighted MAX SAT problem that tries out all possible worlds with the atoms that appear in the rules in order to find the possible world with the maximal weight.

$is(Ron,immature)$ [10]

$is(Ron,immature) \wedge type(H.,sorceress) \Rightarrow likes(H.,Ron)$ [3]

$type(Hermione,sorceress)$ [4]

Atoms:

$is(Ron,immature), type(H., sorceress), likes(H. Ron)$

Exhaustive search is a correct and complete algorithm for the Weighted Max Sat problem. However, it has to analyze 2^n possible worlds, where n is the number of atoms.

$\{is(Ron,immature), type\ (H.,sorceress)\}$: weight 14, etc.

Task: Weighted MAX SAT

Find the KB with the highest weight:

$is(Hermione, smart)[1]$

$is(Herm., smart) \wedge is(Harry, smart) \Rightarrow likes(Herm., Harry)[3]$

$likes(Hermione, Ron) \Rightarrow \neg\, likes(Hermione, Harry)[100]$

$is(Harry, smart)[10]$

$likes(Hermione, Ron)[20]$

Solving Weighted MAX SAT

To always find the optimal solution, one has to do an exhaustive search. Since SAT is NP-complete, so is MAX SAT and Weighted MAX SAT.

To find an approximate solution, possible strategies are:

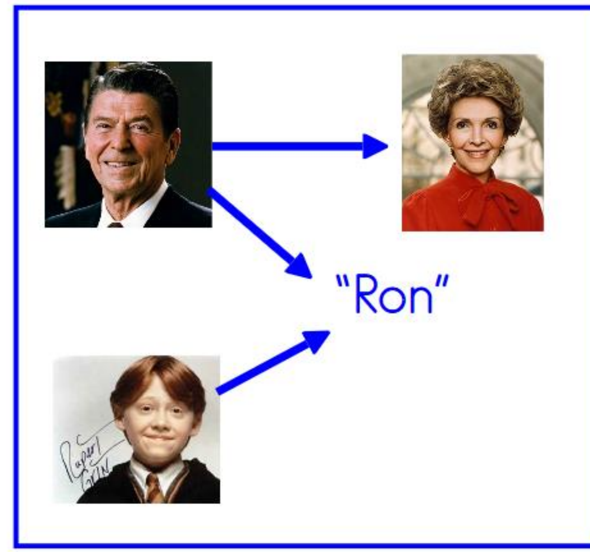
- do an exhaustive search if there are few atoms
- try out several random KBs
- apply unit propagation wherever possible
- give preference to rules/unit clauses with higher weights
- remove atoms that appear only negative, add atoms to the KB that appear only positive.

Back to our problem

"Hermione is married to Ron"

transform everything to logical formulas

$$A \wedge B \Rightarrow C$$



Find best conclusion

$hasSpouse(Hermione, RonWeasley)$

Consistency

Consistency constraints can be expressed by rules:

$$hasSpouse(X,Y) \wedge different(Y,Z) \Rightarrow \neg\, hasSpouse(X,Z)\ [10]$$

$$hasSpouse(X,Y) \Rightarrow type(X,person)\ [20]$$

$\ldots$

Rules and weights can be designed manually. Such rules will guide our information extraction process.

A KB can be expressed as rules

Every fact from the KB can be expressed as a weighted rule:

$$type(Hermione, Person)\ [100]$$

High weight

This is corresponds to the rule

$$\Rightarrow type(Hermione, Person)\ [100]$$

Expressing the corpus as rules

D42: Hermione married Ron.

$occurs$(Hermione@D42, "married", Ron@D42)

Does not talk about Ronald Reagan or Ron Weasley, but about the word "Ron" in document D42.

Ron@D42 is a "word in context" (wic).

Expressing the corpus as rules

D42: Hermione married Ron.

$occurs$(Hermione@D42, "married", Ron@D42)

The word "Ron" in document D42 can mean different entities (from KB):

$means$(Ron@D42, $RonaldReagan$)

$means$(RonD42, $RonWeasley$)

But only one entity in practice:

$$means(X,Y) \wedge different(Y,Z) \Rightarrow \neg\, means(X,Z)$$

Weights for corpus rules

occurrences

$occurs$(Hermione@D42, "married", Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)

$means$(Ron@D42, $RonWeasley$)

$$means(X,Y) \wedge different(Y,Z) \Rightarrow \neg\, means(X,Z)$$

Weights for corpus rules

$occurs$(Hermione@D42, "married", Ron@D42)[3]

From disambiguation by context/prior

$means$(Ron@D42, $RonaldReagan$)[5]

$means$(Ron@D42, $RonWeasley$)[7]

$$means(X,Y) \wedge different(Y,Z) \Rightarrow \neg\, means(X,Z)$$

Weights for corpus rules

$occurs$(Hermione@D42, "married", Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)[5]

$means$(Ron@D42, $RonWeasley$)[7]

"Hard" rule with very high weight

$means(X,Y) \wedge different(Y,Z) \Rightarrow \neg\, means(X,Z)$ [100]

Weights for corpus rules

$occurs$(Hermione@D42, "married", Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)

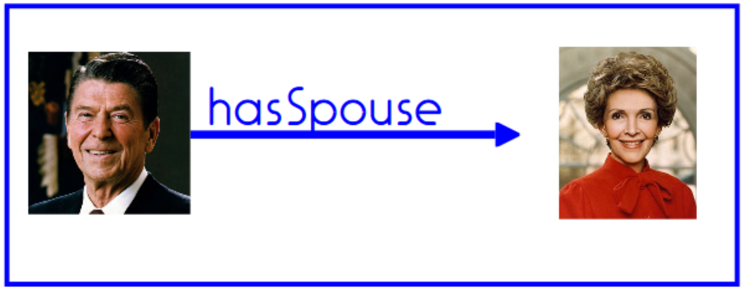
$means$(Ron@D42, $RonWeasley$)

$means(X,Y) \wedge different(Y,Z) \Rightarrow \neg\, means(X,Z)$ [100]

Let us ignore the weights for a moment.

Deducing patterns

KB



+

Reagan married Davis.

=

"X married Y"
is pattern for
hasSpouse(X,Y)

$hasSpouse(Reagan,Davis)$

occurs(R@1,"married", D@1)

means(R@1, *Reagan*)

means(D@1, *Davis*)

$occurs(X,P,Y)$
$\wedge\ means(X,X')$
$\wedge\ means(Y,Y')$
$\wedge\ R(X',Y')$
$\Rightarrow\ isPatternFor(P,R)$

isPatternFor("married",*hasSpouse*)

Applying patterns

"X married Y"
is pattern for
$hasSpouse$(X,Y)

$isPatternFor$("married", $hasSpouse$)

+

$occurs$(E@1, "married", P@1)

$means$(E@1, $Elvis$)

$means$(P@1, $Priscilla$)

Elvis married Priscilla.

=

$occurs(X,P,Y)$

$\wedge\ means(X,X')$

$\wedge\ means(Y,Y')$

$\wedge\ isPatternFor(P,R)$

$\Rightarrow R(X',Y')$

$hasSpouse(Elvis,Priscilla)$

Task: Pattern deduction by rules

Pattern deduction:

$$occurs(X,P,Y) \wedge means(X,X') \wedge means(Y,Y') \wedge R(X',Y') \Rightarrow isPatternFor(P,R)$$

Pattern application:

$$occurs(X,P,Y) \wedge means(X,X') \wedge means(Y,Y') \wedge isPatternFor(P,R) \Rightarrow R(X',Y')$$

1: $occurs$(P@1, "adores", E@1)
2: $means$(E@1, $Elvis$)
3: $means$(P@1, $Priscilla$)
4: $hasSpouse(Priscilla,Elvis)$
5: $occurs$(M@1, "adores", E@1)
6: $means$(M@1, $Madonna$)

All rules have weight 1.

Compute facts that will be in the best world.

Life is not easy

- words are ambiguous

  "Ron"

- corpora may err

  "Madonna is married to Elvis"

- contradictions may occur

  Reagan was married twice.

=> we will compute the most plausible world

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married", $spouse$)[1]

$means$(H@1, $Hermione$)[5]

$means$(R@1, $RonWeasley$)[2]

$means$(R@1, $Reagan$)[3]

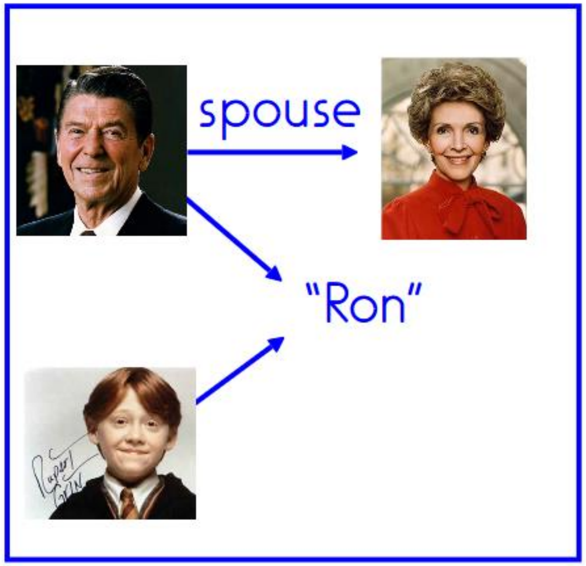
$means(X, Y) \wedge Y \neq Z \Rightarrow \neg\, means(X, Z)$[10]

$spouse(X, Y) \wedge Y \neq Z \Rightarrow \neg\, spouse(Z, X)$[6]

+ Symmetry of marriage

+ Pattern Application Rule [10]

+ Facts from the KB [100]



World1:

World2:

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married", $spouse$)[1]

$means$(H@1, $Hermione$)[5]

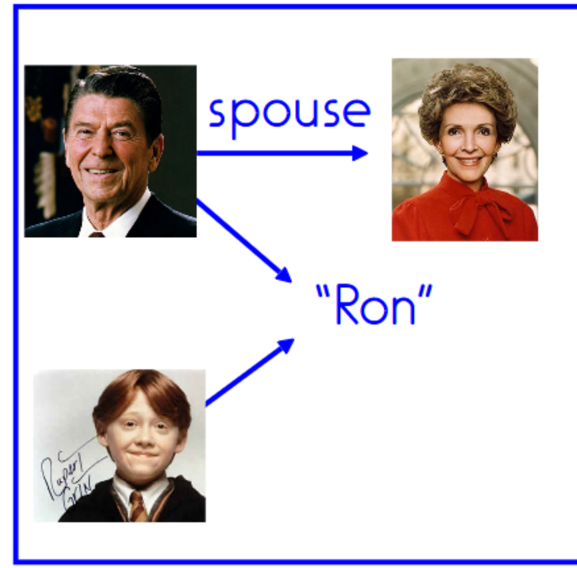
$means$(R@1, $RonWeasley$)[2]

$means$(R@1, $Reagan$)[3]

$means(X, Y) \wedge Y \neq Z \Rightarrow \neg\, means(X, Z)$[10]

$spouse(X, Y) \wedge Y \neq Z \Rightarrow \neg\, spouse(Z, X)$[6]

+ Symmetry of marriage
+ Pattern Application Rule [10]
+ Facts from the KB [100]



World1:

loses 2

wins 3

loses 6

World2:

loses 3

wins 6

wins 2

Outline

1. Motivation
2. Consolidation
    - MaxSAT
    - Probabilistic Soft Logic
    - Google Knowledge Vault
3. Pattern Learning
    - Association rule mining
    - Matrix completion
    - Knowledge graph embeddings

Probabilistic Soft Logic

- MaxSAT computes THE most likely world
- Probability of individual statements to be true?
  →Via sum of probability of all worlds in which they are true

→ Markov logic networks/Probabilistic soft logic
- Efficient approximations via Gibbs sampling
  - Compute random worlds
  - Update individual variables based on priors and state of other variables
  - Repeat updating until convergence
  - Sum up probabilities from samples with positive variable value

→ Prominent system: DeepDive

Google Knowledge Vault

- Four text-based extractors
  - Text, HTML (DOM) trees, tables, manual RDF annotations
- Two predictive models
  - ~Random walk-based, supervised matrix completion using MLP

→ 6 features per triple

→ Supervised classification
  - linear regression/boosted decision stumps

- Ignores interaction between tuples

- Prototypical for many IE systems
  - E.g., Aristo TupleKB, Quasimodo

Outline

1. Motivation
2. Consolidation
    - MaxSAT
    - Probabilistic Soft Logic
    - Google Knowledge Vault
3. Pattern Learning
    - Association rule mining
    - Matrix completion
    - Knowledge graph embeddings

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married', $spouse$)[1]

$means$(H@1, $Hermione$)[5]

$means$(R@1, $RonWeasley$)[2]

$means$(R@1, $Reagan$)[3]

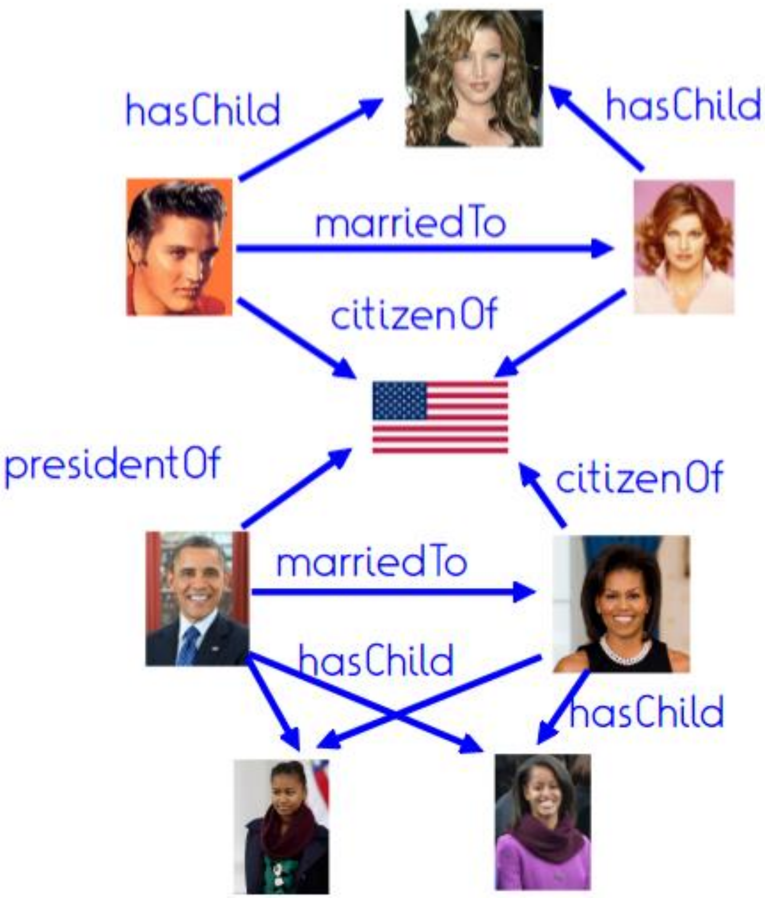
$means(X, Y) \wedge Y \neq Z \Rightarrow \neg\, means(X, Z)$[10]

$spouse(X, Y) \wedge Y \neq Z \Rightarrow \neg\, spouse(Z, X)$[6]

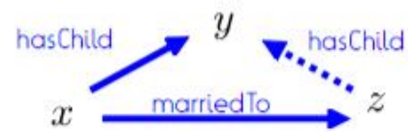
+ Pattern Application Rule [10]

Where do we get
these constraints from?

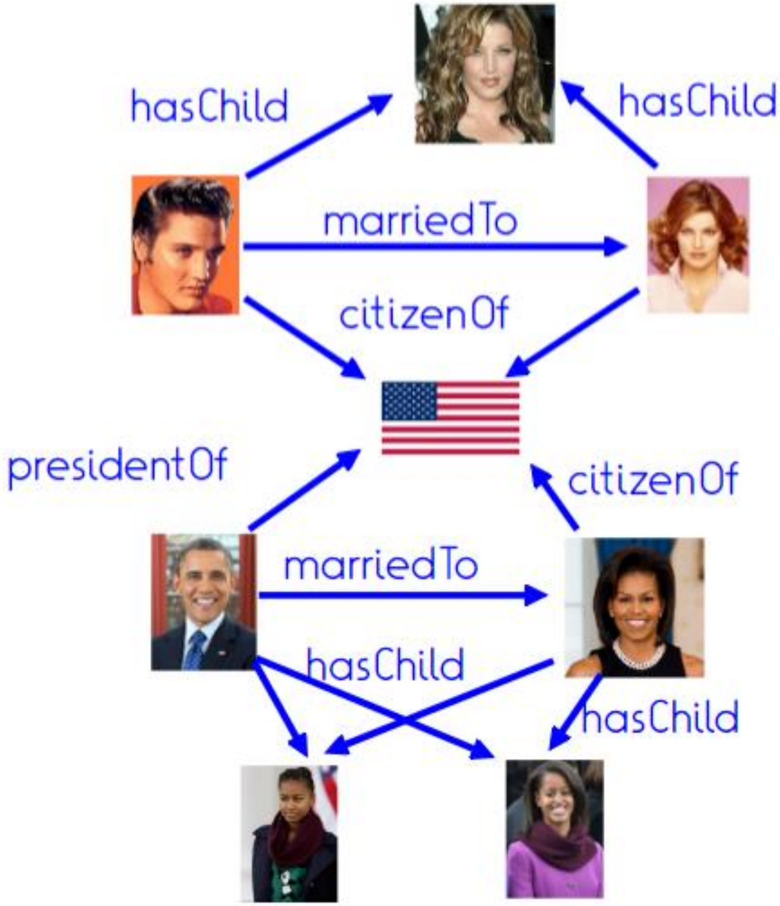
Patterns



"Whenever
someone has
a child,
this is also
the child of
the spouse"



Association rule mining



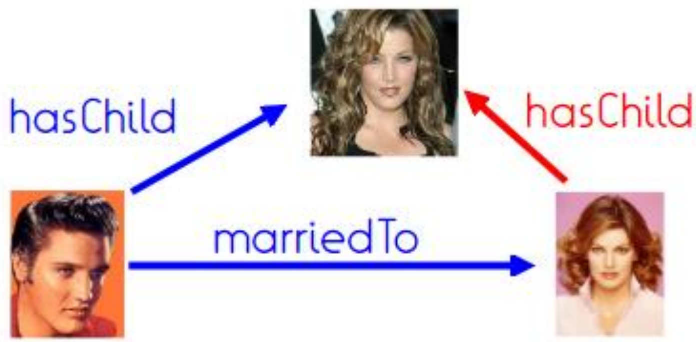
$hasChild(x,y)$

$\wedge\ marriedTo(x,z)$

$\Rightarrow hasChild(z,y)$

(rules don't have to be always correct)

Rule Mining for Completion

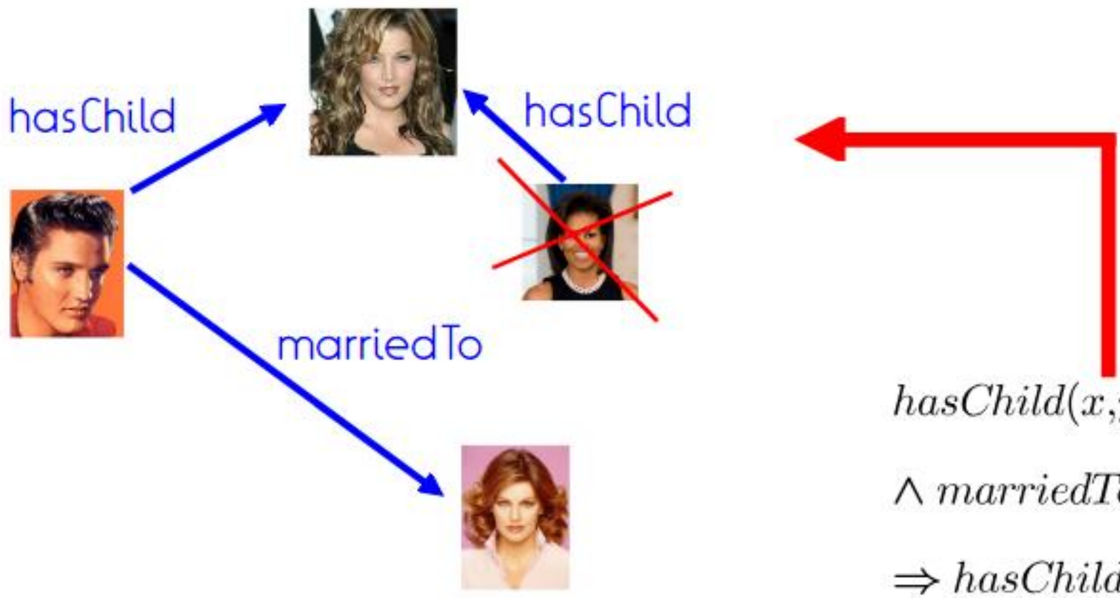


$$hasChild(x,y)$$

$$\wedge\ marriedTo(x,z)$$

$$\Rightarrow\ hasChild(z,y)$$

Rule Mining for Correction



$$hasChild(x,y)$$
$$\wedge\ marriedTo(x,z)$$
$$\Rightarrow\ hasChild(z,y)$$

Rule Mining for Insights

$$hasChild(x,y) \wedge marriedTo(x,z) \Rightarrow hasChild(z,y)$$

... in 4% of the cases

$$in(x, Europe) \wedge president(x, y) \Rightarrow male(x)$$

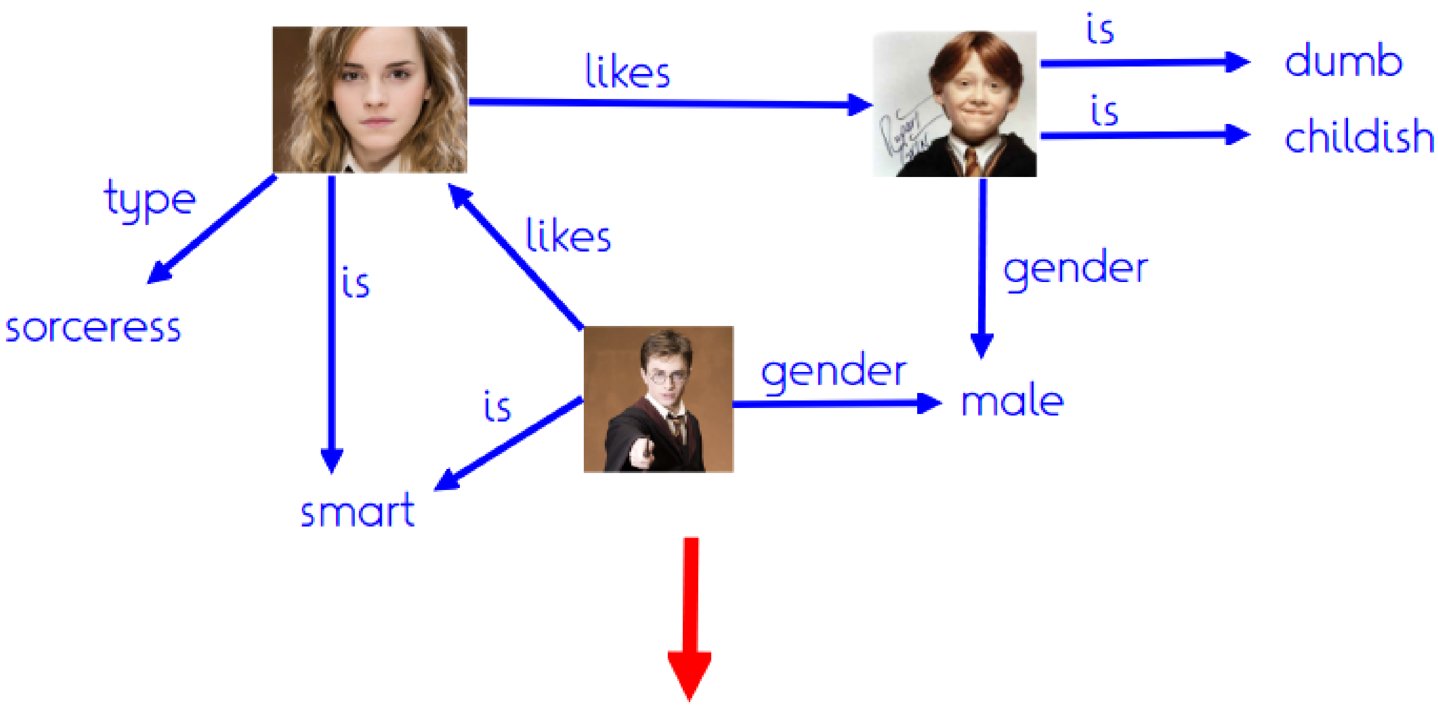
$$married(x, y) \Rightarrow married(y, x)$$

... obvious for us, but non-trivial for a computer.

Applications in

- artificial intelligence / question answering
- information extraction (as we saw)
- science
- engineering (fault finding)
- medicine
- social network mining

Running example



$$type(x, sorceress) \wedge dumb(y) \Rightarrow likes(x, y) \quad ?$$

Def: Safe Rule

The head of a rule is its positive literal, the other literals are the body of the rule. A rule is safe, if all variables of the head appear in the body.

Body Head

Example:

- safe: $child(x, y) \wedge spouse(x, z) \Rightarrow child(z, y)$
- not safe: $smart(x) \Rightarrow loves(y, x)$

Def: Connected Rule & Closed Rule

Two atoms are connected if they share a variable or constant.
A rule is connected if every atom is transitively connected to every other atom.

Example:

- not connected: $dumb(y) \wedge smart(x) \Rightarrow rich(x)$

?

A rule is closed, if every variable appears at least twice.

Example:

- closed: $married(x, y) \wedge male(x) \Rightarrow married(y, x)$
- not closed: $married(x, y) \wedge likes(x, z) \Rightarrow married(y, x)$

A variable is dangling, if it appears only once.

Query

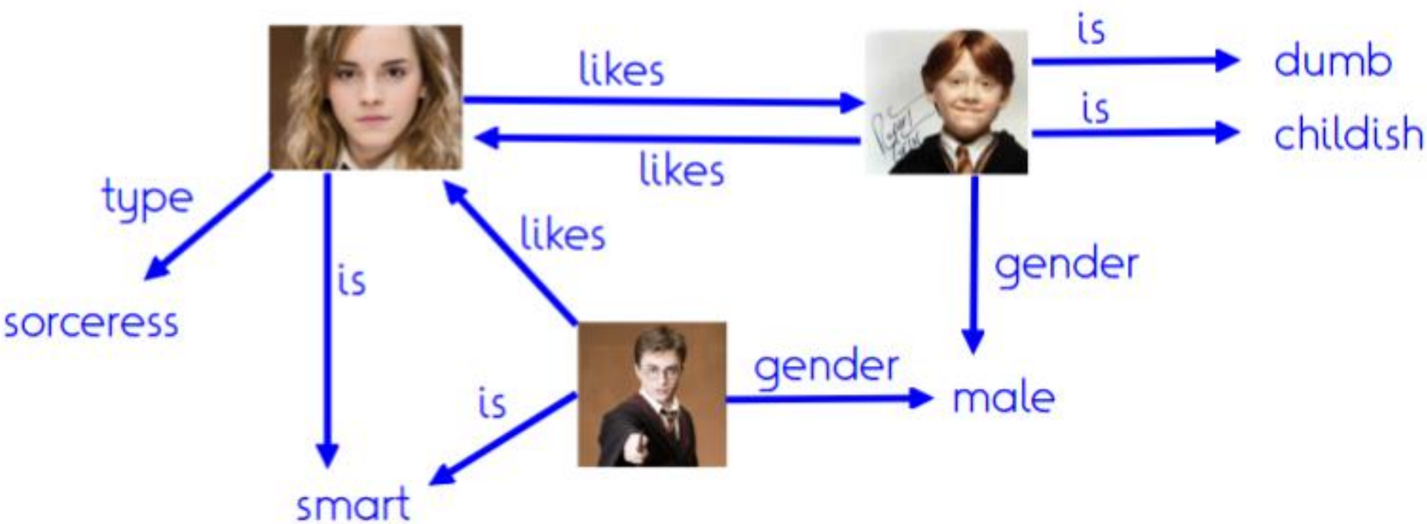
A (conjunctive) query is a conjunction of literals

$$loves(Hermione, x) \wedge dumb(x) \wedge smart(y) \ \ ?$$

An answer to a query for a given KB is a binding that makes the query true.

$$\{\, x \longrightarrow Ron, y \longrightarrow Harry \,\}$$

Example

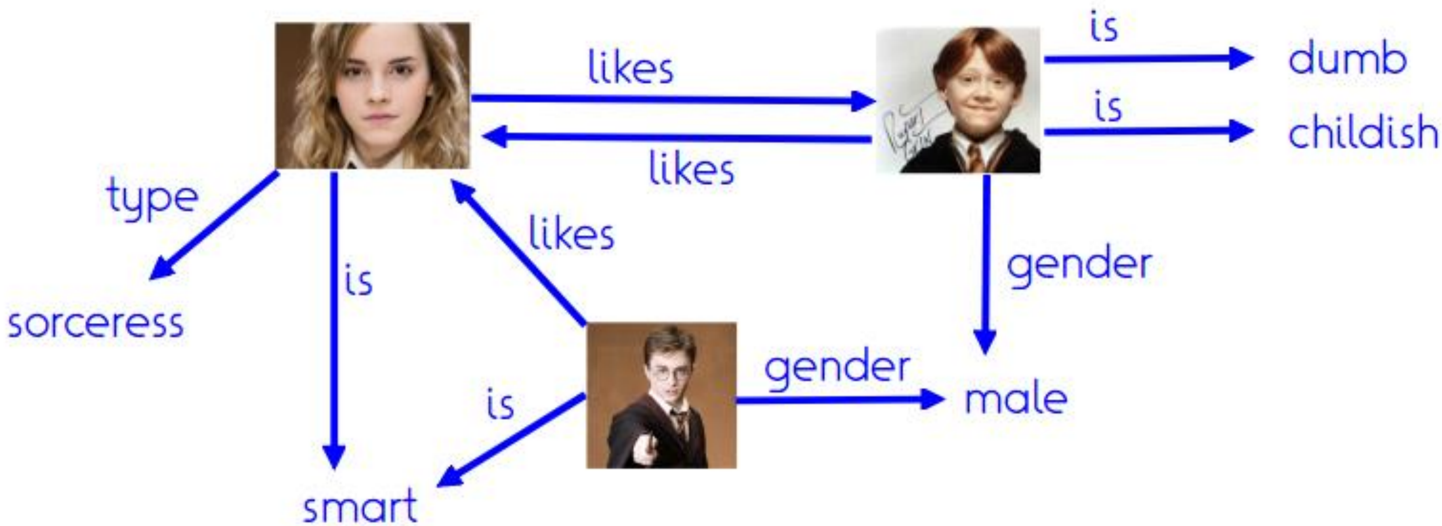


$likes(x, y) \wedge gender(x, male)$?

$gender(x, male)$?

$likes(x, y) \wedge is(y, smart) \wedge gender(y, male)$?

Queries and Rules



A query can be written as a rule.

$$likes(x, y) \land gender(x, male) \Rightarrow answer(x, y)$$

The body of a rule is basically a query.

$$\underbrace{gender(x, male) \land is(x, dumb)}_{\text{query}} \Rightarrow likes(Hermione, x)$$

Types of Reasoning

Deductive reasoning:

- Given α and $\alpha \Rightarrow \beta$
- Deduce β

$$smart(Bill).\quad smart(x) \Rightarrow rich(x)$$
$$\rightarrow \quad rich(Bill)$$

Abductive reasoning:

- Given β and $\alpha \Rightarrow \beta$
- Deduce α

$$rich(Bill).\quad smart(x) \Rightarrow rich(x)$$
$$\rightarrow \quad smart(Bill)$$

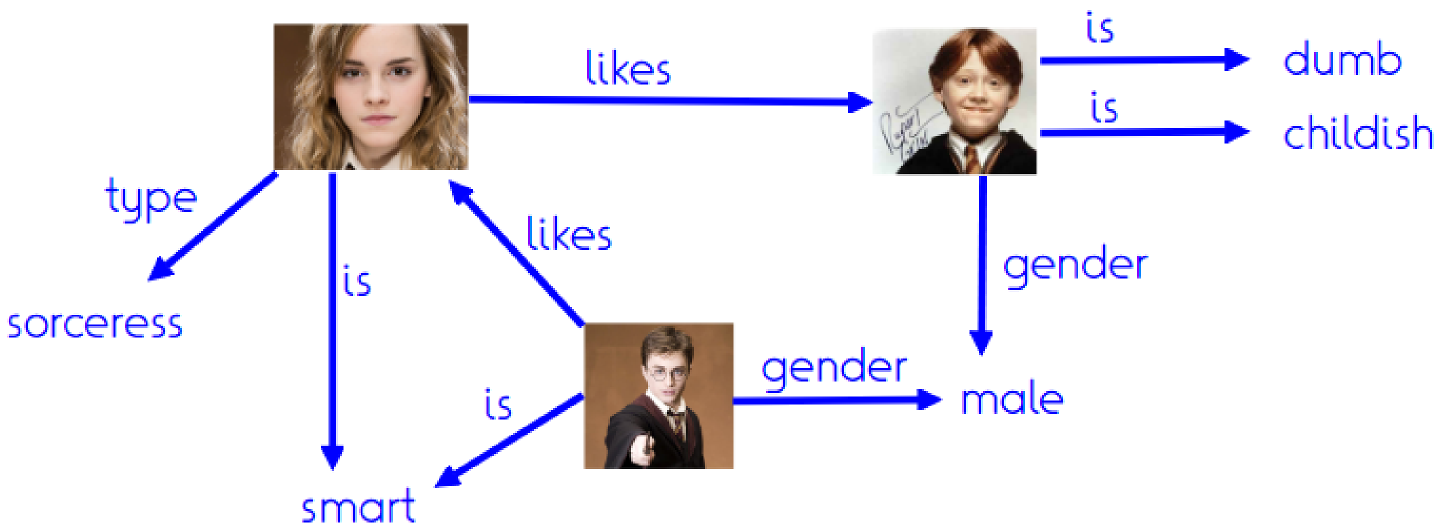
Inductive reasoning:

- Given α and β
- Deduce $\alpha \Rightarrow \beta$

$$rich(Bill).\quad smart(Bill)$$
$$\rightarrow \quad smart(x) \Rightarrow rich(x)$$

This is what
we want to do

Example



$$type(x, sorceress) \wedge is(y, dumb) \Rightarrow likes(x, y) \quad ?$$

$$type(x, sorceress) \wedge is(y, childish) \Rightarrow likes(x, y) \quad ?$$

$$is(x, smart) \wedge is(y, dumb) \Rightarrow likes(x, y) \quad ?$$

$$type(x, s.) \wedge is(y, dumb) \wedge gender(y, male) \Rightarrow likes(x, y) \quad ?$$

Def: Inductive Logic Programming

Given

- background knowledge B

  B is a set of logical formulae.

  In most cases, B is simply a set of atoms, i.e., a KB.
- positive examples $E+$ and negative examples $E-$

  $E+$ and $E-$ are commonly just sets of atoms, usually of the same relation, which is called the target relation.

inductive logic programming (ILP) is the task of finding

- a hypothesis h

  Usually, the hypothesis is a set of rules that have the target relation in the head. Often, h is a single rule.

such that we have

- completeness: $B \wedge h \models E+$ (all positive examples are predicted)
- correctness: $B \wedge h \not\models E-$ (no negative example is predicted)
- minimality: There is no shorter rule with the same properties

Example: ILP

Given

- background knowledge B
  - dumb(Ron), smart(Harry), male(Ron), male(Harry)
  - sorceress(Hermione)
- positive examples $E+$ and negative examples $E-$
  - $E+$: likes(Hermione, Ron)
  - $E-$: likes(Hermione, Harry)

inductive logic programming (ILP) is the task of finding

- a hypothesis h
  - $sorceress(x) \wedge dumb(y) \Rightarrow likes(x, y)$

such that we have

- completeness: $B \wedge h \vDash E+$ (all positive examples are predicted)
- correctness: $B \wedge h \nvDash E-$ (no negative example is predicted)
- minimality: There is no shorter rule with the same properties

Language Bias

The language bias of an ILP problem is the type of hypotheses that we consider.

- rules or arbitrary formulae?

  $loves(x, y)\ \ xor\ \ hates(x, y)$
- with or without negation?

  $\neg\ smart(x) \Rightarrow loves(Hermione, x)$
- with or without quantifiers?

  $smart(x) \Rightarrow \exists\ y\text{: } loves(y, x)$
- how many atoms per rule?

  $loves(Herm., x) \Leftarrow smart(x) \wedge cute(x) \wedge rich(x) \wedge \ldots$
- with or without constants?

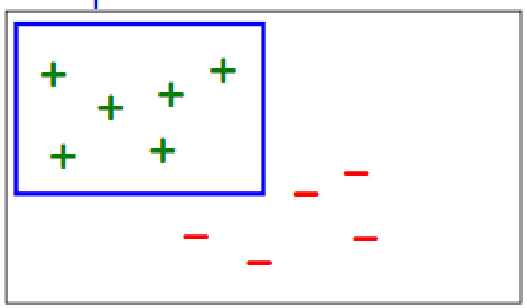
  $hates(Harry, x) \Rightarrow loves(Hermione, x)$
- how many rules?

Def: Properties of hypotheses

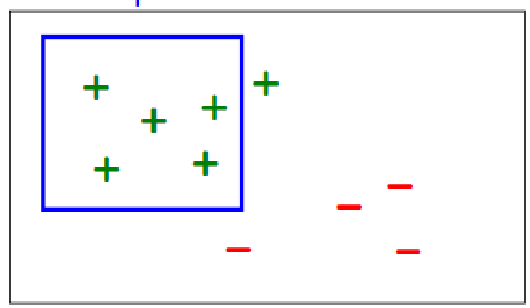
h is complete, if it predicts all positive examples.

h is consistent, if it does not predict any negative examples.

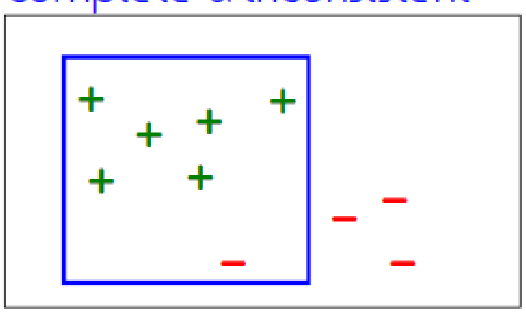
complete & consistent



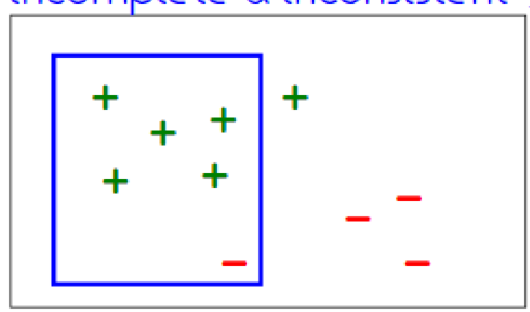
incomplete & consistent



complete & inconsistent

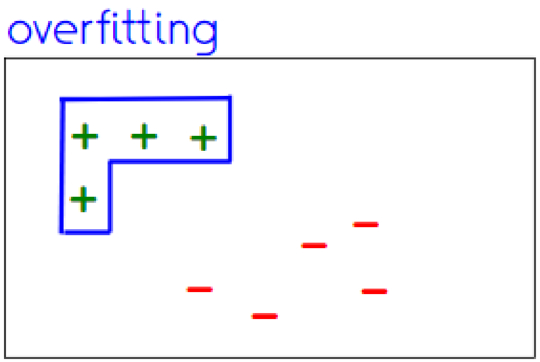


incomplete & inconsistent



Overfitting

A hypothesis overfits, if it does not generalize to new positive examples.



B: dumb(Ron), smart(Harry), male(Ron), male(Harry)
sorceress(Hermione)

E+: likes(Hermione, Ron)
E-: likes(Hermione, Harry)

$$x{=}Ron \wedge y{=}Herm. \wedge dumb(x) \wedge male(x) \Rightarrow likes(x, y)$$

Overgeneralization

A hypothesis overgeneralizes, if it does not generalize to new negative examples.

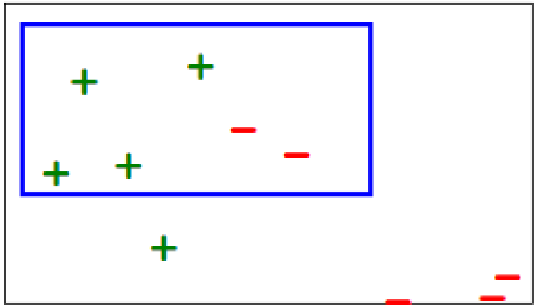
B: dumb(Ron), smart(Harry), male(Ron), male(Harry)
sorceress(Hermione)

E+: likes(Hermione, Ron)
E-: likes(Hermione, Harry)

$$y \neq Harry \Rightarrow likes(x, y)$$

Def: Support

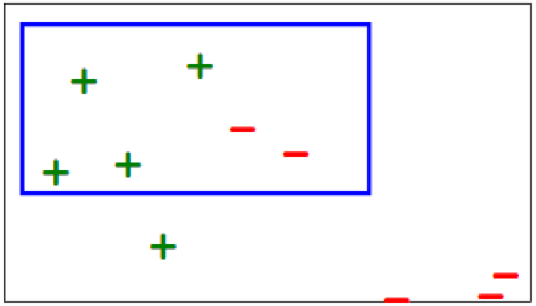
The support of a hypothesis is the number of predicted positive examples.



$support = 4$

Task: Support

The support of a hypothesis is the number of predicted positive examples.



B = {person(Ron), person(Harry), person(Hermione)}

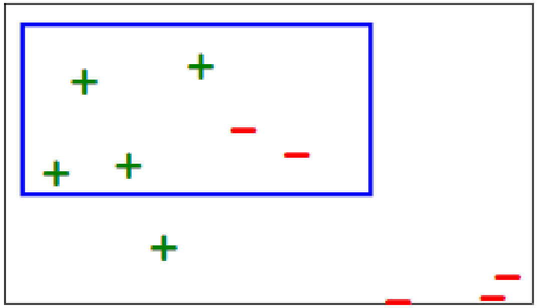
E+ = {likes(Hermione, Ron), likes(Harry, Hermione), likes(Hermione, Harry)}

E- = {likes(Hermione, Hermione)}

$$person(x) \Rightarrow likes(Hermione, x)$$

Def: Confidence

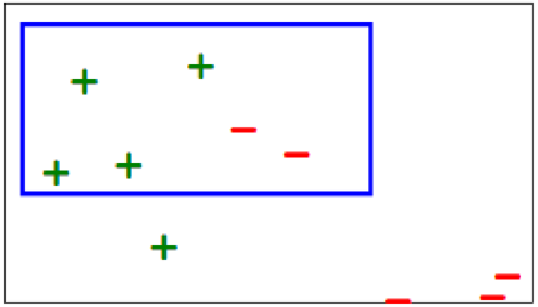
The confidence of a hypothesis is the ratio
of predicted positive examples, out of all predicted examples.



$$confidence = \frac{4}{6}$$

Task: Confidence

The confidence of a hypothesis is the ratio
of predicted positive examples, out of all predicted examples.



B = {male(Ron), male(Harry)}

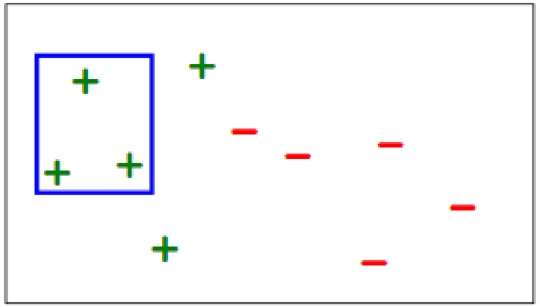
E+ = {likes(Hermione, Ron), likes(Harry, Hermione)}

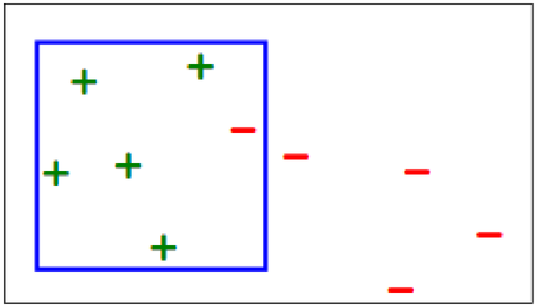
E- = {likes(Hermione, Harry)}

$$male(x) \Rightarrow likes(Hermione, x)$$

Confidence / Support trade-off

Usually, confidence and support are in a trade-off.





- high confidence
- low support

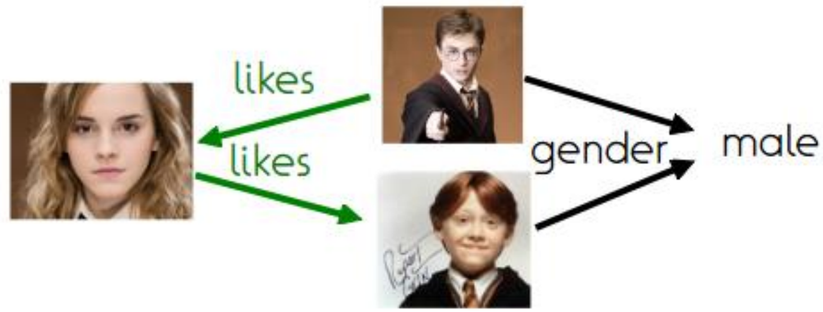
=> conservative hypothesis

- high support
- low confidence

=> general hypothesis

Confidence in a KB

KBs usually do not store negative information:



B = {gender(Harry, male), gender(Ron, male)}

E+ = {likes(Hermione, Ron), likes(Harry, Hermione)}

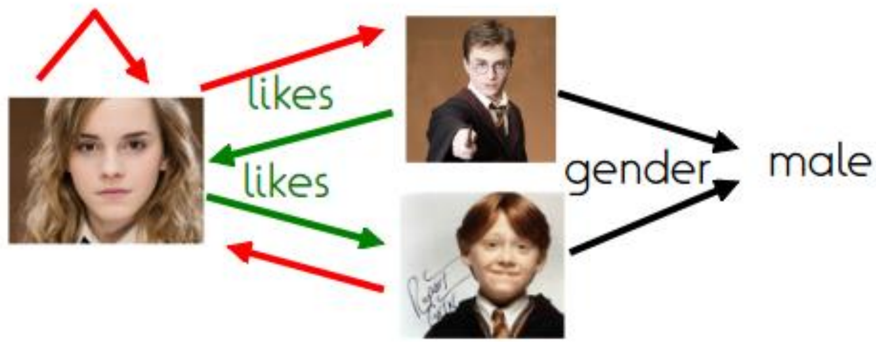
E- = {}

$$gender(x, male) \Rightarrow likes(Hermione, x)$$

Def: Closed World Assumption

The Closed World Assumption (CWA) assumes that all atoms that are not in the KB are wrong (i.e., negative examples).

Confidence with CWA

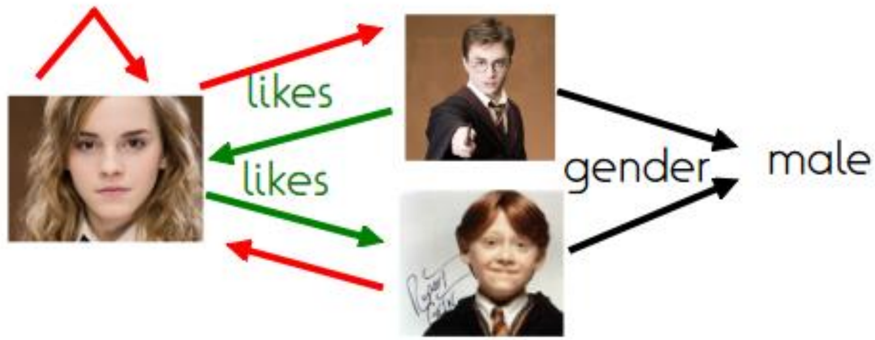


B = {gender(Harry, male), gender(Ron, male)}

E+ = {likes(Hermione, Ron), likes(Harry, Hermione)}

E- = {likes(Hermione, Hermione), likes(Ron, Hermione), likes(Ron, Ron), ... }

Task: Confidence with CWA



B = {gender(Harry, male), gender(Ron, male)}

E+ = {likes(Hermione, Ron), likes(Harry, Hermione)}

E- = {likes(Hermione, Hermione), likes(Ron, Hermione), likes(Ron, Ron), ... }

$male(x) \Rightarrow likes(x, Hermione)$

Problem with the CWA

We have a great rule that makes great predictions,
but since these predictions were not yet known,
they act as counter-examples!

B = {gender(Harry, male), gender(Ron, male)}
E+ = {likes(Hermione, Ron), likes(Harry, Hermione)}
E- = {likes(Hermione, Hermione), likes(Ron, Hermione), likes(Ron, Ron), ... }

$male(x) \Rightarrow likes(x, Hermione)$

likes(Harry, Hermione)
likes(Ron, Hermione)

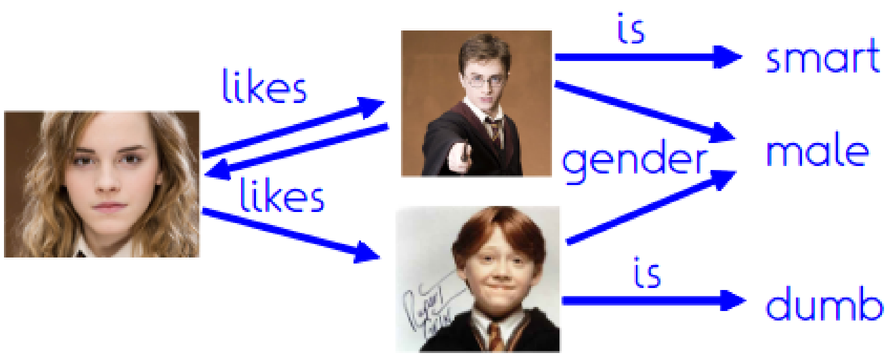
$confidence$
$=50$ %

Def: Open World Assumption

The Open World Assumption (OWA) assumes that not all absent atoms are wrong (i.e., not all are neg. examples).

Problem with the OWA

The Open World Assumption (OWA) assumes that not all absent atoms are wrong (i.e., not all are neg. examples).



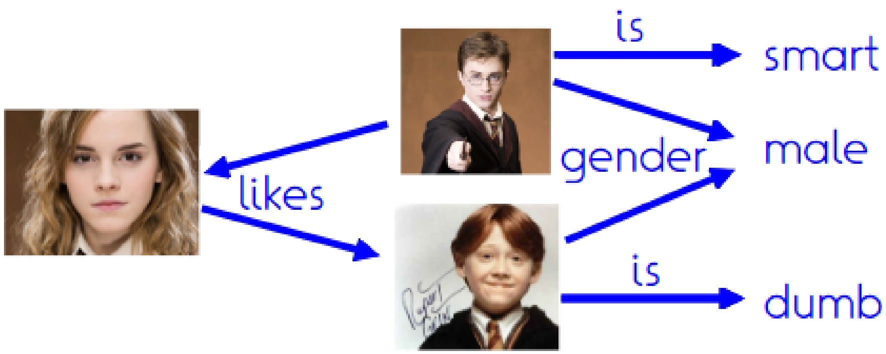
$gender(x, male) \Rightarrow is(x, smart)$?

E- = {}

The OWA does not yield counter-examples!

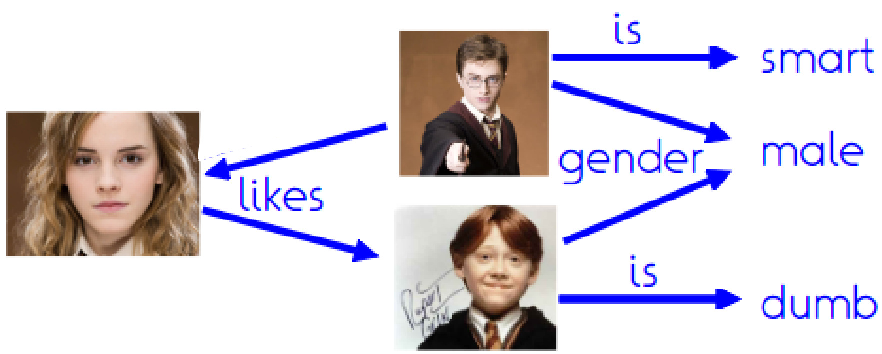
Def: Partial Completeness Assumption

The partial completeness assumption assumes that
if the KB contains $r(x, y)$ then it contains all correct $r(x, y')$.



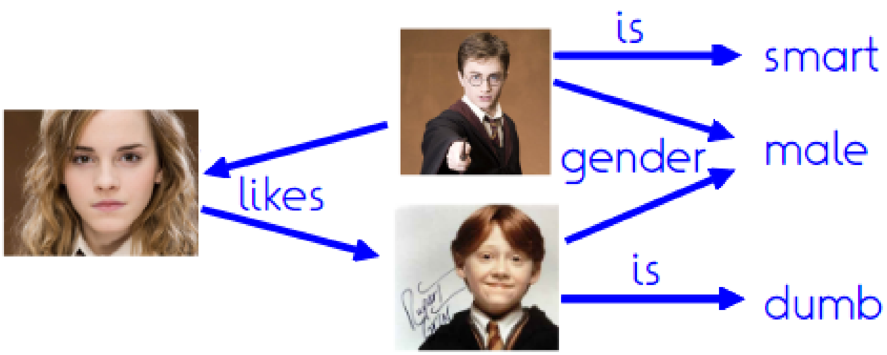
Task: Partial Completeness Assumption

The partial completeness assumption assumes that
if the KB contains $r(x, y)$ then it contains all correct $r(x, y')$.



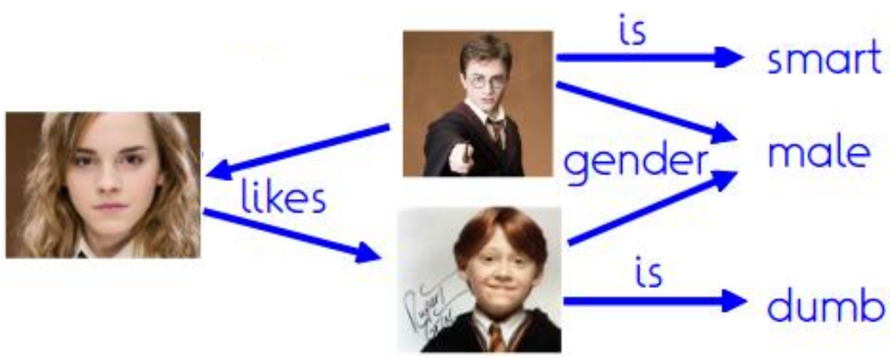
Is Harry dumb?
Does Hermione like herself?
Is Hermione male? female? none? both?
Does Ron like Hermione?

Confidence with the PCA



$gender(x, male) \Rightarrow likes(x, Hermione)$

Task: Confidence with the PCA



$$gender(x, male) \Rightarrow likes(Hermione, x)$$

Computing the PCA confidence

Let's compute the PCA confidence for

$$p(x, y) \Rightarrow r(x, y)$$

i.e., the ratio of predictions that are true out of the predictions that are true or supposed to be false.

Computing the PCA confidence

Let's compute the PCA confidence for

$$p(x, y) \Rightarrow r(x, y)$$

Predictions that are true:

$\# \, (x, y)\colon \quad p(x, y) \in KB \quad \wedge \quad r(x, y) \in KB$

Predictions that are supposed to be false:

$\# \, (x, y)\colon (p(x, y) \in KB) \wedge (\exists \, y'\colon r(x, y') \in KB) \wedge (r(x, y) \notin KB)$

KB knows r about x

but not the one that we predict

Predictions that are true + supposed to be false:

$\# \, (x, y)\colon \quad p(x, y) \in KB \quad \wedge \quad \exists \, y'\colon r(x, y') \in KB$

Computing the PCA confidence

The PCA confidence for

$$p(x, y) \Rightarrow r(x, y)$$

is

$$\frac{\#\ (x, y):\quad p(x, y) \in KB \quad \wedge \quad r(x, y) \in KB}{\#\ (x, y):\quad p(x, y) \in KB \quad \wedge \quad \exists\, y'\!: r(x, y') \in KB}$$

The numerator is just the support of the rule.

Computing the PCA confidence

The PCA confidence for

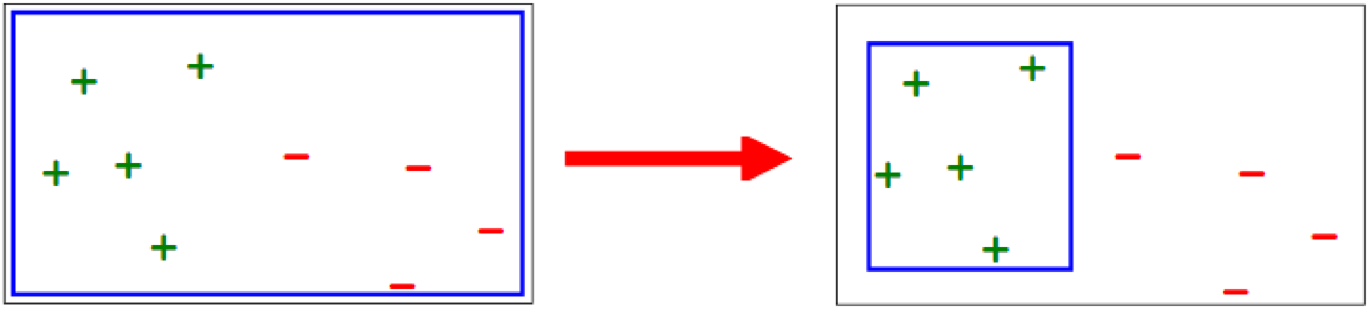
$$\beta \Rightarrow r(x, y)$$

is

$$\frac{support(\beta \Rightarrow r(x, y))}{\#\,(x, y):\quad p(x, y) \in KB \quad \wedge \quad \exists\, y': r(x, y') \in KB}$$

Top-down ILP

Top-down ILP starts with one all-embracing hypothesis
and specializes it until no negative examples are covered.



Specialization operator

A specialization operator is a function that takes a rule, and returns a set of more special rules.

Some specialization operators for binary atoms are:

$$\Rightarrow hasChild(x, y)$$

- Add dangling atom: Adds a connected atom that has a fresh variable.

$$married(x, z) \Rightarrow hasChild(x, y)$$

- Add instantiated atom: Adds a connected atom that has a constant.

$$likes(x, cheese) \Rightarrow hasChild(x, y)$$

- Add closing atom: Adds an atom that is connected in both arguments.

$$hasFather(y, x) \Rightarrow hasChild(x, y)$$

Example: Specialization operators

$$\Rightarrow hasChild(x, y)$$

add dangling atom $\downarrow$

$$married(x, z) \Rightarrow hasChild(x, y)$$

add closing atom $\downarrow$

$$hasChild(z, y) \wedge married(x, z) \Rightarrow hasChild(x, y)$$

add instantiated atom $\downarrow$

$$rel(z, catholic) \wedge hasChild(z, y) \wedge married(x, z) \Rightarrow hasChild(x, y)$$

Example: Add dangling atom

$$\Rightarrow hasChild(x, y)$$

add dangling atom $\downarrow$

All combinations of
a connected variable (x, y)
with a fresh variable (z).

$$married(x, z) \Rightarrow hasChild(x, y)$$
$$married(z, x) \Rightarrow hasChild(x, y)$$
$$married(z, y) \Rightarrow hasChild(x, y)$$
$$married(y, z) \Rightarrow hasChild(x, y)$$

With all relations

$$loves(x, z) \Rightarrow hasChild(x, y)$$
$$loves(z, x) \Rightarrow hasChild(x, y)$$
$$loves(z, y) \Rightarrow hasChild(x, y)$$
$$loves(y, z) \Rightarrow hasChild(x, y)$$

Pruning

Pruning a rule means abandoning it together with all specializations.
Examples for pruning strategies are:

- Prune rules that are too long

$$rel(z, catholic) \wedge hasChild(z, y) \wedge married(x, z) \Rightarrow hasChild(x, y)$$

$$\downarrow$$

$$\times$$

Pruning

Pruning a rule means abandoning it together with all specializations.
Examples for pruning strategies are:

- Prune rules that are too long

$$rel(z, catholic) \wedge hasChild(z, y) \wedge married(x, z) \Rightarrow hasChild(x, y)$$

$$\downarrow$$

$$\times$$

- Prune rules that have too small support

$$type(x, pope) \wedge married(x, y) \Rightarrow hasChild(x, z)$$

$$\downarrow$$

$$\times$$

Support decreases monotonically with specialization!

Other pruning strategies

- Prune rules that are redundant

$$married(x, z) \wedge married(x, z) \Rightarrow hasChild(x, y)$$

$$\downarrow$$

$$\times$$

- Prune rules that have an equivalent elsewhere

$$married(x, y) \wedge hasChild(x, y) \Rightarrow hasChild(z, y)$$

$$hasChild(z, y) \wedge married(z, y) \Rightarrow hasChild(x, y)$$

$$\downarrow$$

$$\times$$

- Prune specializations of rules with perfect confidence

$$married(x, y) \Rightarrow married(y, x)$$

100% confidence, do not specialize further,
because confidence will stay and support will decrease.

AMIE for Rule Mining

AMIE is the following top-down rule mining algorithm

- start with the queue of rules for each relation:

$$Q := [\Rightarrow r_1(x_1,y_1), \ldots]$$

- while Q is not empty
  - h := Q.dequeue()
  - If h is a good rule, output h
  - for each specialization h' of h
    - if h' is not pruned, Q.enqueue(h')

support>threshold

PCA conf>threshold

Prune

- rules > 3 atoms
- rules with support<threshold
- ...

Example: AMIE

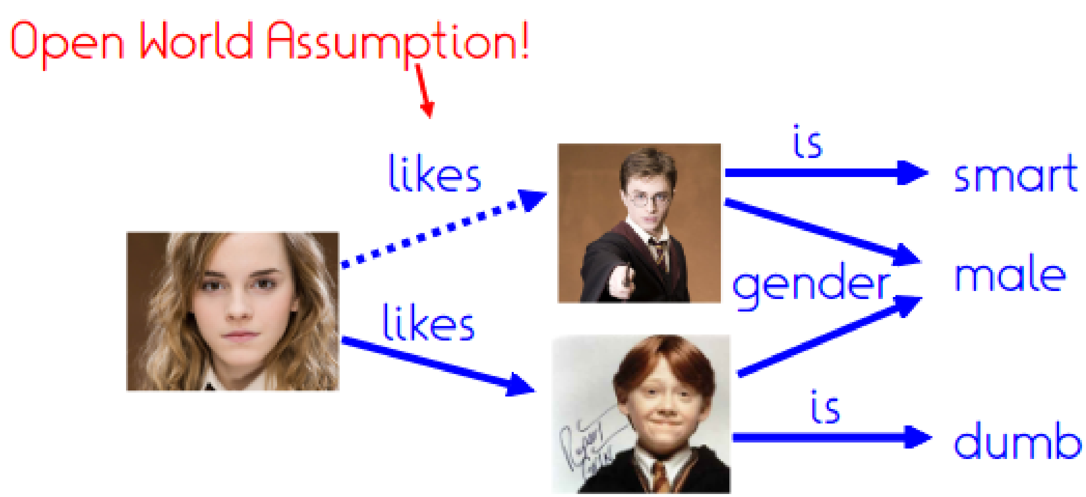
$$gender(x, male) \Rightarrow likes(Herm, x)$$

support: 2
confidence: 100%
=> output this rule

Rule cannot get better => stop

Summary

- ILP is the task of finding hypotheses that cover examples
- ILP can be performed top-down by specialization (AMIE)
- ... or bottom-up by generalization (GOLEM)
- The Open World Assumption allows that data that is not in the KB can still be true.



Outline

1. Motivation
2. Consolidation
    - MaxSAT
    - Probabilistic Soft Logic
    - Google Knowledge Vault
3. Pattern Learning
    - Association rule mining
    - Matrix completion
    - Knowledge graph embeddings

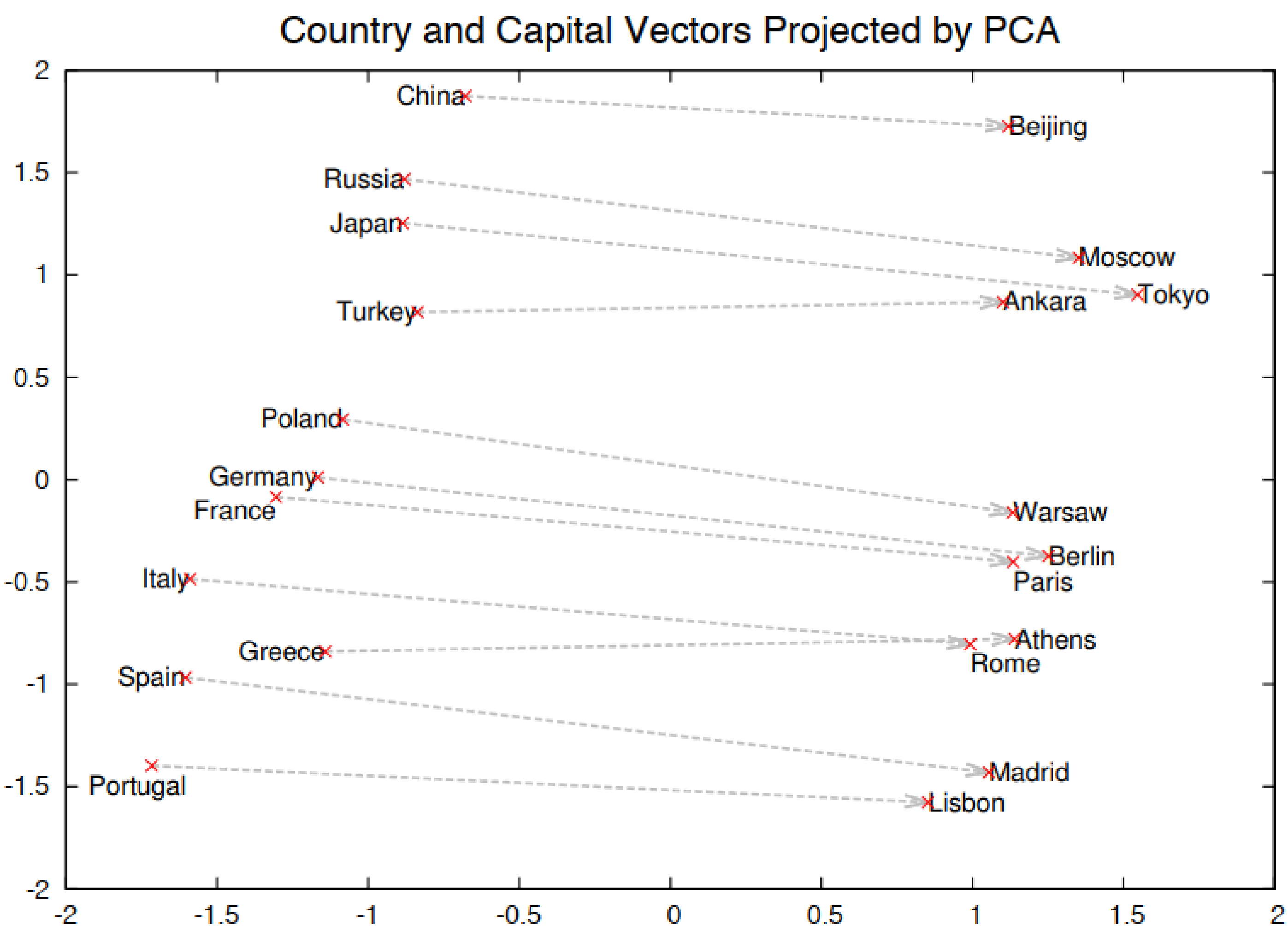
Matrix completion for IE

	president of	*prime minister of*	*chancellor of*	*chief executive*	*leader of*	*head of state*	headOf	Top Member
Obama, U.S.	Y			Y			Y	
Merkel, Germany			Y		Y	Y	Y	
S Harper Canada		Y			Y		Y	
V Putin Russia	Y				Y	Y	Y	
Larry Page Google				Y			Y	Y
V. Rometty IBM	Y			Y	Y			Y
Tim Cook Apple				Y			Y	Y
E Grimson MIT			Y				Y	

Classical problem in recommender systems
- PCA, Matrix factorization, …

See e.g., Riedel et al., "Universal schema"

Knowledge graph embeddings

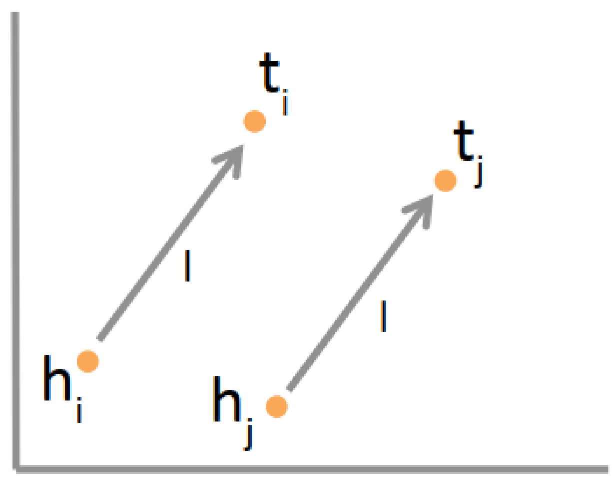


from [Mikolov 2013]

Knowledge graph embeddings (2)

- For word embeddings a by-product
- Idea: Train embeddings to intentionally exhibit such features:

"Translation intuition"
For a triple (h, l, t) : $\vec{h} + \vec{l} \approx \vec{t}$.



Plethora of ML works: TransE, TransH, TransG, TransR, …

References

- Core papers:
  - Suchanek et al., SOFIE – a self-organizing framework for IE, WWW 2009
  - Galarraga et al., AMIE: Association rule mining under incomplete evidence, WWW 2013
- Further reading
  - Dong, Xin, et al. "Knowledge vault: A web-scale approach to probabilistic knowledge fusion." KDD 2014
  - Wu et al., Incremental Knowledge Base Construction Using DeepDive, VLDB 2015
  - Wang,et al. "Knowledge graph embedding by translating on hyperplanes." AAAI 2014.
  - Riedel et al. "Relation extraction with matrix factorization and universal schemas." NAACL. 2013.
- Slides
  - Adopted from Fabian Suchanek, Laura Dietz, Andrew McCallum

Assignment 8

- Find patterns in Game of Thrones
- Data: Subset of infobox relations from https://gameofthrones.fandom.com (provided)
- Task:
  - Evaluate rules of the form P(_,_), R(_,_) → S(_,_)
    - 3 specific variable patterns
  - For each form, find top 10 rules in terms of support
  - Bonus
    - All rules with two atoms in body (safe+connected)
    - Rules w/ constants
    - Other scores, e.g., PCA confidence

Take home

- IE typically operates in settings of uncertainty and noise
  - Redundancy is solution and challenge at same time
- Consolidation frameworks translate background knowledge, logical constraints, extraction candidates into logical/probabilistic frameworks
  - Then optimize for global coherence
  - Usually require some constraints
  - MaxSAT as example approach
- Pattern discovery in extractions is useful for completion and consolidation
  - Association rule mining as sample approach