Tuffy

Scaling up Statistical Inference in Markov Logic using an RDBMS

Feng Niu, Chris Ré, AnHai Doan, and Jude Shavlik

University of Wisconsin-Madison

One Slide Summary

Machine Reading is a DARPA program to capture knowledge expressed in free-form text

Similar challenges in enterprise applications

We use **Markov Logic**, a language that allows rules that are likely – but not certain – to be correct

Markov Logic yields high quality, but current **implementations** are confined to small scales

Tuffy scales up Markov Logic by orders of magnitude using an RDBMS

Outline

- Markov Logic
  - Data model
  - Query language
  - Inference = grounding then search
- Tuffy the System
  - Scaling up grounding with RDBMS
  - Scaling up search with partitioning

A Familiar Data Model

Datalog?

Markov Logic program

Datalog + Weights
≈ Markov Logic

Relations with known facts

EDB

Relations to be predicted

IDB

Markov Logic*

- Syntax: a set of ***weighted*** logical rules

  3 wrote(s,t) ∧ advisedBy(s,p) → wrote(p,t)

  // students' papers ***tend*** to be co-authored by advisors

  - Weights: cost for rule violation

- Semantics: a distribution over ***possible worlds***

  - Each possible world I incurs total cost $cost(I)$
  - $\Pr[I] \propto \exp(-cost(I))$ exponential models
  - Thus most likely world has lowest cost

* [Richardson & Domingos 2006]

Markov Logic by Example

Rules

3 **wrote(s,t) ∧ advisedBy(s,p) → wrote(p,t)**
// students' papers ***tend*** to be co-authored by advisors

5 **advisedBy(s,p) ∧ advisedBy(s,q) → p = q**
// students ***tend*** to have at most one advisor

∞ **advisedBy(s,p) → professor(p)**
// advisors ***must*** be professors

EDB

Evidence

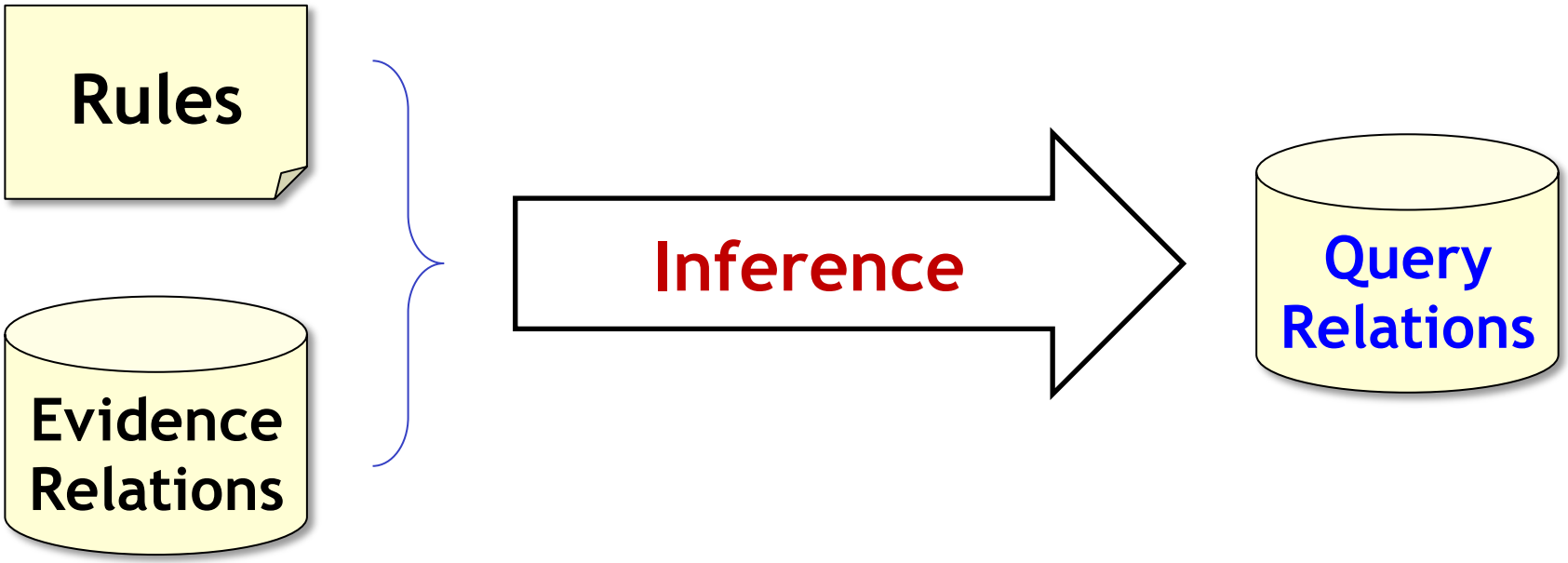
wrote(Tom, Paper1)
wrote(Tom, Paper2)
wrote(Jerry, Paper1)
professor(John)
…

IDB

Query

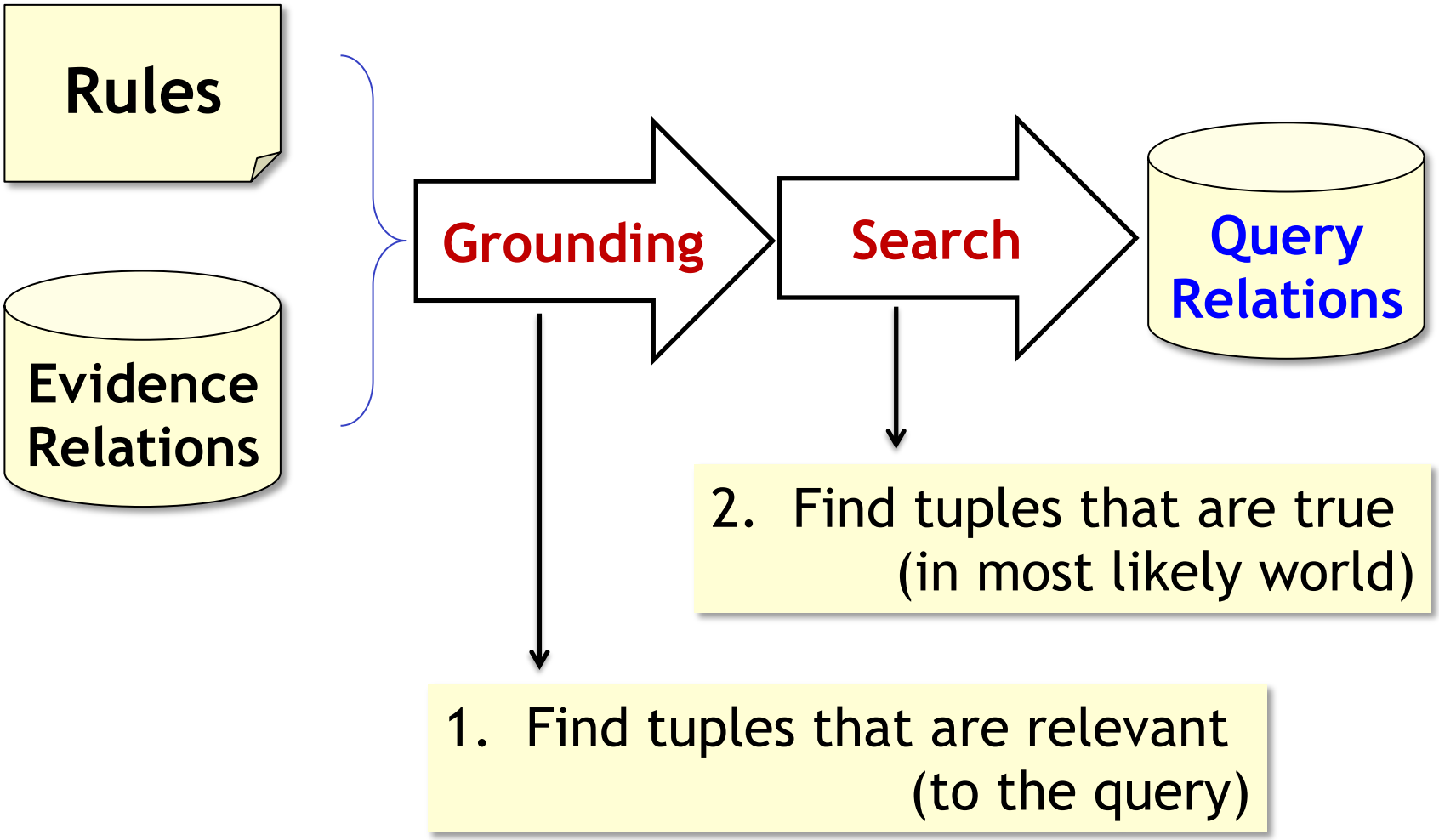
advisedBy(?, ?)
// who advises whom

Inference



MAP	regular tuples
Marginal	tuple probabilities

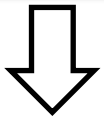
Inference


Rules
Evidence Relations
Grounding
Search
Query Relations
2. Find tuples that are true (in most likely world)
1. Find tuples that are relevant (to the query)

How to Perform Inference

- Step 1: Grounding
  - Instantiate the rules

3 wrote(*s*, *t*) ∧ advisedBy(*s*, *p*) → wrote(*p*, *t*)



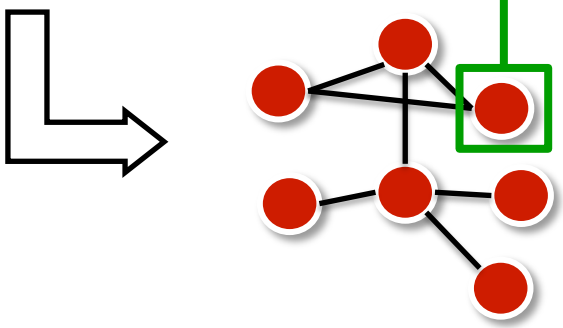
Grounding

3 wrote(Tom, P1) ∧ **advisedBy**(Tom, Jerry) → wrote (Jerry, P1)
3 wrote(Tom, P1) ∧ **advisedBy**(Tom, Chuck) → wrote (Chuck, P1)
3 wrote(Chuck, P1) ∧ **advisedBy**(Chuck, Jerry) → wrote (Jerry, P1)
3 wrote(Chuck, P2) ∧ **advisedBy**(Chuck, Jerry) → wrote (Jerry, P2)
…

How to Perform Inference

- Step 1: Grounding
  - Instantiated rules → Markov Random Field (MRF)
    - A graphical structure of correlations

3 wrote(Tom, P1) ∧ **advisedBy**(Tom, Jerry) → wrote (Jerry, P1)
3 wrote(Tom, P1) ∧ **advisedBy**(Tom, Chuck) → wrote (Chuck, P1)
3 wrote(Chuck, P1) ∧ **advisedBy**(Chuck, Jerry) → wrote (Jerry, P1)
3 wrote(Chuck, P2) ∧ **advisedBy**(Chuck, Jerry) → wrote (Jerry, P2)
…

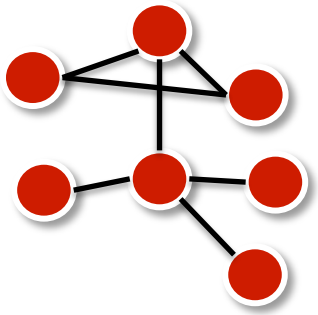


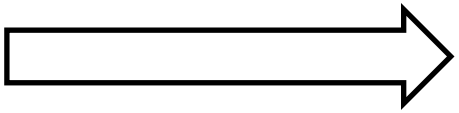
Nodes: Truth values of tuples

Edges: Instantiated rules

How to Perform Inference

- Step 2: Search
  - Problem: Find most likely state of the MRF (NP-hard)
  - Algorithm: WalkSAT*, random walk with heuristics
  - Remember lowest-cost world ever seen





Search

advisee	advisor
Tom	Jerry
Tom	Chuck

False

True

* [Kautz et al. 2006]

Outline

- Markov Logic
  - Data model
  - Query language
  - Inference = grounding then search
- Tuffy the System
  - Scaling up grounding with RDBMS
  - Scaling up search with partitioning

Challenge 1: Scaling Grounding

- Previous approaches
  - Store all data in RAM
  - Top-down evaluation

[Singla and Domingos 2006]
[Shavlik and Natarajan 2009]

RAM size quickly becomes bottleneck

Even when runnable, grounding takes long time

Grounding in Alchemy*

- Prolog-style top-down grounding with C++ loops
  - Hand-coded pruning, reordering strategies

3 wrote(s, t) ∧ advisedBy(s, p) → wrote(p, t)

```
For each person s:
    For each paper t:
        If !wrote(s, t) then continue
        For each person p:
            If wrote(p, t) then continue
            Emit grounding using <s, t, p>
```

Grounding sometimes accounts for over 90% of Alchemy's run time

[*] reference system from UWash

Grounding in Tuffy

Encode grounding as SQL queries

Executed and optimized by RDBMS

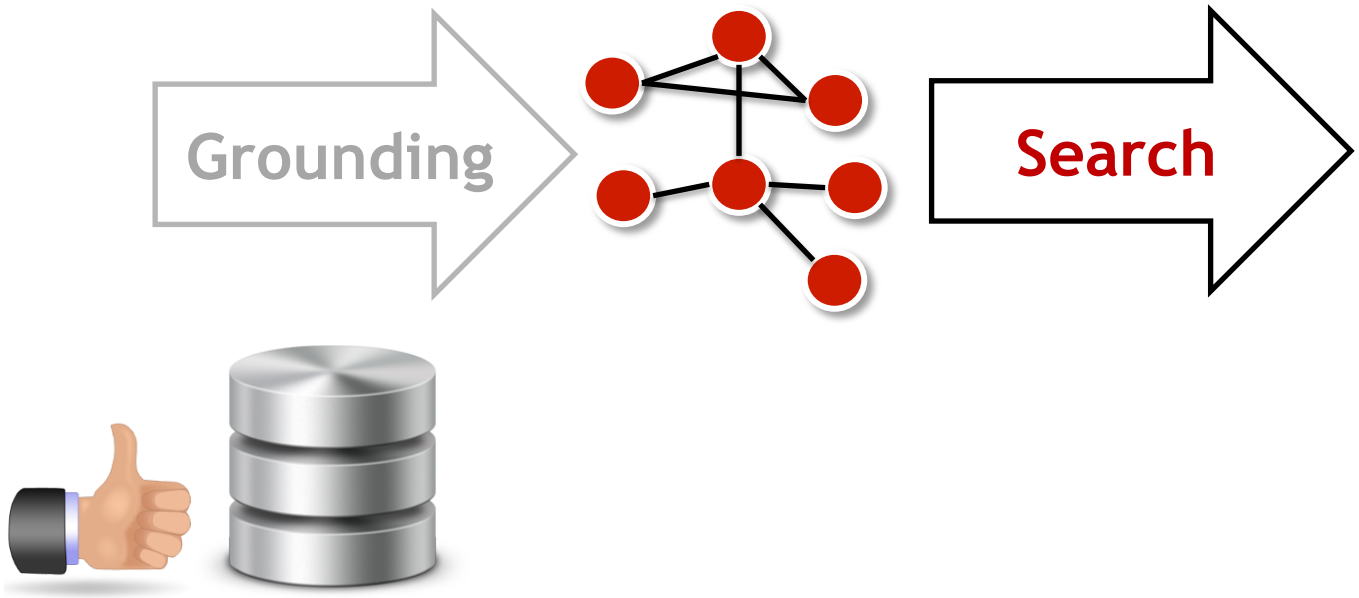
Grounding Performance

Tuffy achieves orders of magnitude speed-up

	Relational Classification	Entity Resolution
Alchemy [C++]	68 min	420 min
Tuffy [Java + PostgreSQL]	1 min	3 min
Evidence tuples	430K	676
Query tuples	10K	16K
Rules	15	3.8K

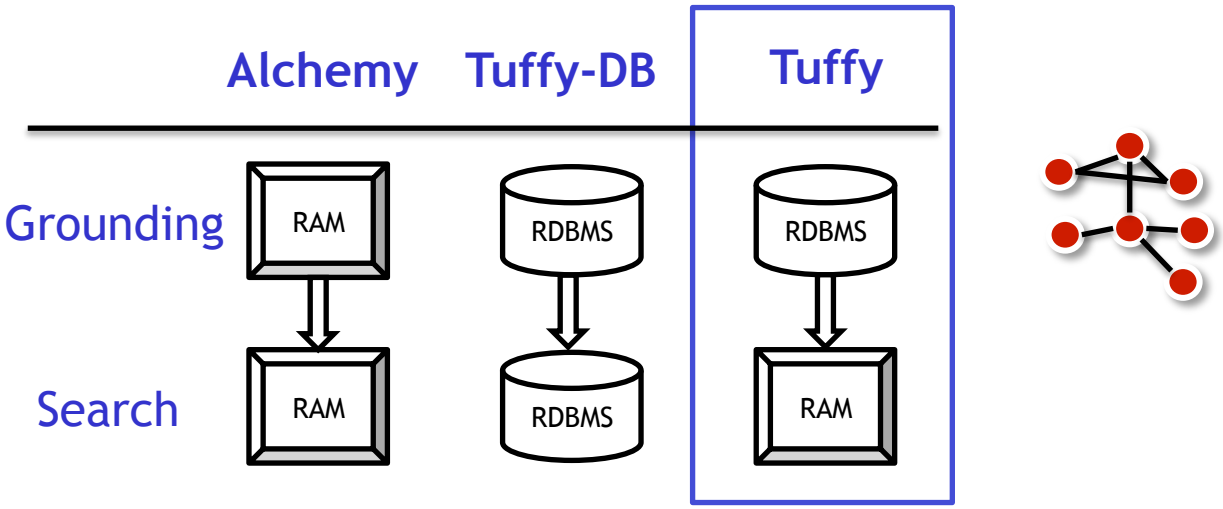
Yes, join algorithms & optimizer are the key!

Challenge 2: Scaling Search



Challenge 2: Scaling Search

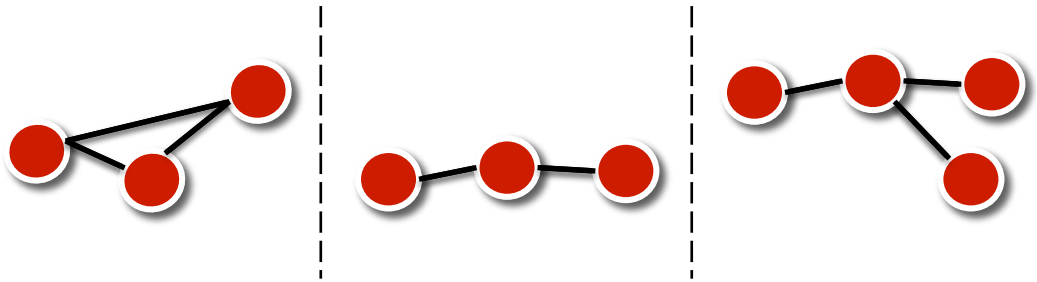
- First attempt: pure RDBMS, search also in SQL
  - **No-go:** millions of random accesses
- Obvious fix: hybrid architecture



Problem: stuck if |MRF| > |RAM|!

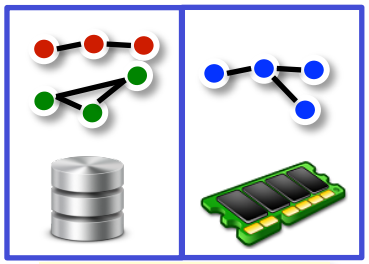
Partition to Scale up Search

- Observation
  - MRF sometimes have multiple components
- Solution
  - Partition graph into components
  - Process in turn

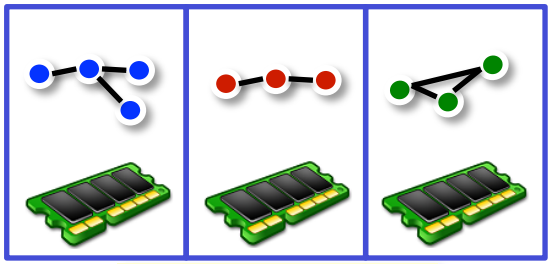


Effect of Partitioning

- Pro



Scalability

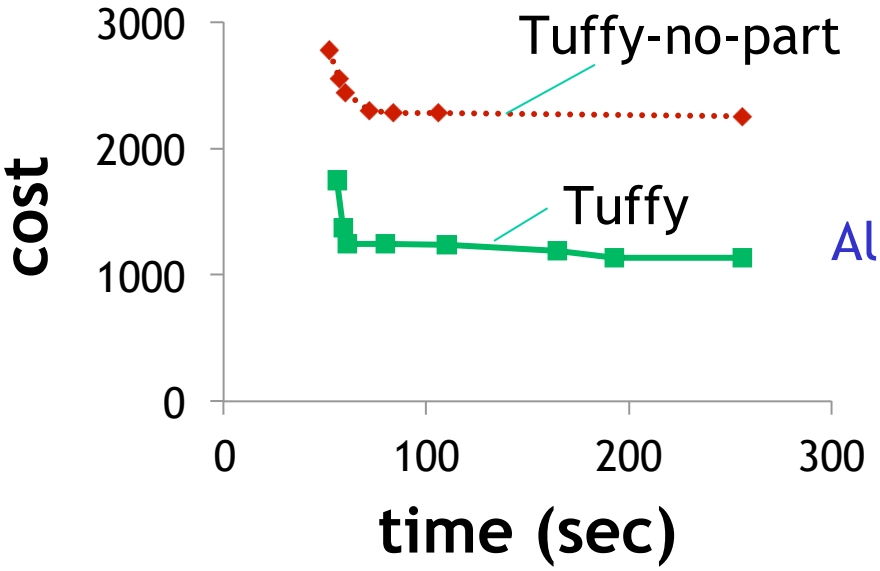


Parallelism

- Con (?)
  - Motivated by scalability
  - Willing to sacrifice quality

What's the effect on quality?

Partitioning Hurts Quality?



Relational Classification

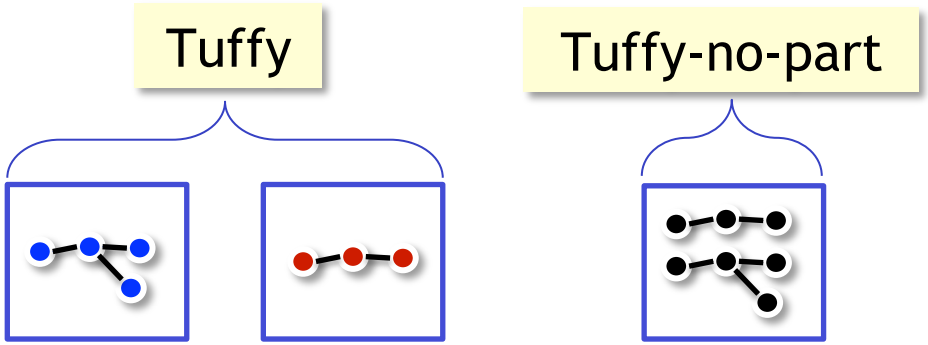
Alchemy took over 1 hr.
Quality similar to
Tuffy-no-part

Goal: lower the cost quickly

Partitioning can actually improve quality!

Partitioning (Actually) Improves Quality

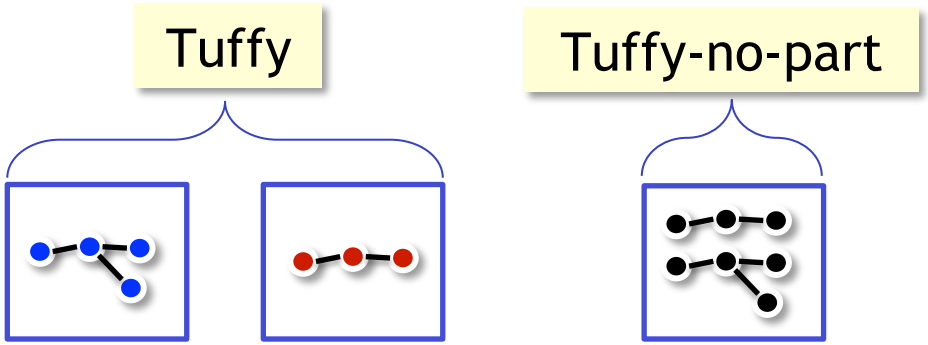
Reason:



WalkSAT iteration	cost1	cost2	cost1 + cost2
1	5	20	25
min	**5**	**20**	**25**

Partitioning (Actually) Improves Quality

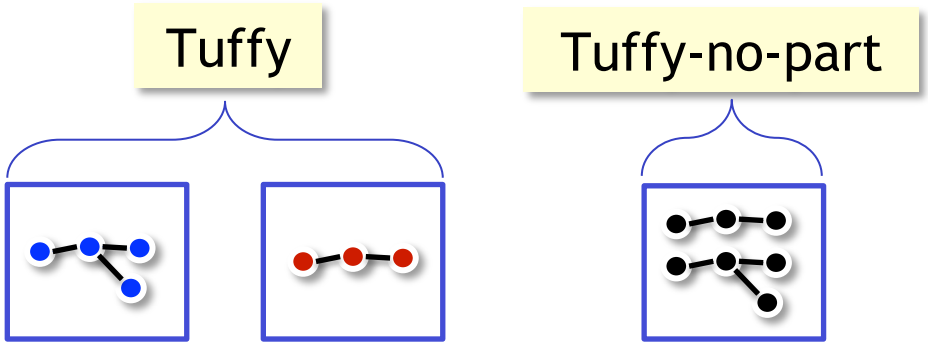
Reason:



WalkSAT iteration	cost1	cost2	cost1 + cost2
1	5	20	25
2	20	10	30
min	**5**	**10**	**25**

Partitioning (Actually) Improves Quality

Reason:



WalkSAT iteration	cost1	cost2	cost1 + cost2
1	5	20	25
2	20	10	30
3	20	5	25
min	**5**	**5**	**25**

cost[Tuffy] = 10
cost[Tuffy-no-part] = 25

Partitioning (Actually) Improves Quality

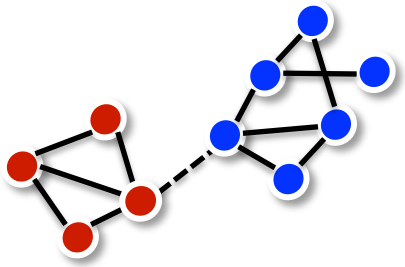
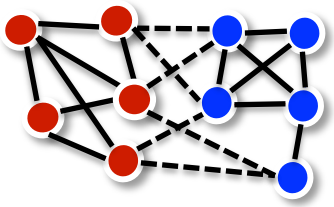
Theorem (roughly):

Under certain technical conditions, component-wise partitioning reduces expected time to hit an optimal state by (2 ^ #components) steps.

100 components → 100 years of gap!

Further Partitioning

Partition one component further into pieces

Graph	Scalability	Quality
Sparse 	☺	☺
Dense 		☺ / ☹

In the paper: cost-based trade-off model

Conclusion

- Markov Logic is a powerful framework for statistical inference
  - But existing implementations do not scale
- Tuffy scales up Markov Logic inference
  - RDBMS query processing is perfect fit for grounding
  - Partitioning improves search scalability *and quality*
- Try it out!

http://www.cs.wisc.edu/hazy/tuffy