A Markov Logic Semantic Role Labeler using Phrase Structure Grammars: Final Report

Connor Mayer

CPSC 503

UBC

Why semantic role labelling

- How to efficiently and accurately tag the predicate in a sentence, as well as its arguments, and what their role is
  - Who did what to whom, where and how?

"Yesterday, the dingo ate the wallaby"

Temporal **Agent** **Predicate** **Goods (theme)**

- Many NLP applications
- **Markov logic networks** offer a unique approach to this task
  - Allow for simultaneous determination of a predicate, the arguments to the predicate, and the sense of the predicate (Meza-Ruiz and Riedel 2009)
  - Highly desirable in SRL, as these decisions are not independent
  - Standard approaches cannot perform all tasks simultaneously.

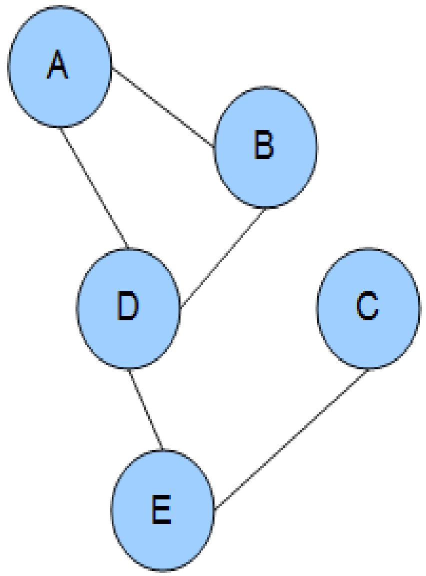
Markov Logic Networks

- Undirected graphical model representing joint probability distribution
- Define potential function ϕ over each clique (complete subset):

- Can represent in log-linear form:

$$P(X=x) = \frac{1}{Z} \exp\left\{ \sum_j w_j f_j(x) \right\}$$

where *j* ranges over all features *f*, *wj* is a real-valued weight associated with the *j*th feature and *Z* is a normalization constant.



Markov Logic Networks

- Markov Logic Networks consist of one binary node for **each possible grounding** of every predicate.

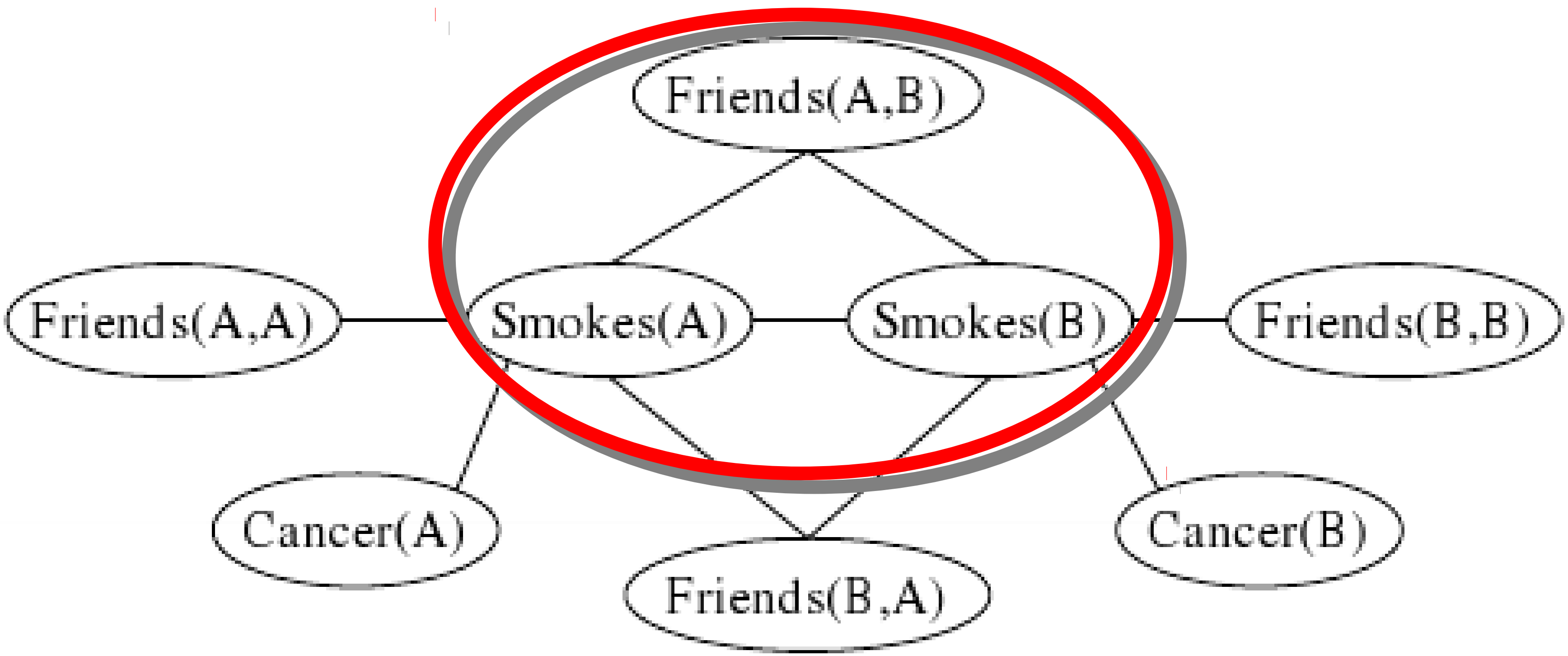
- Predicates that occur in the same formula form cliques in the graph.

- Weights are essentially potential functions that represent the truth of that particular grounding.

$$P(X = x) = \frac{1}{Z} \exp\left\{ \sum_j w_j f_j(x) \right\}$$

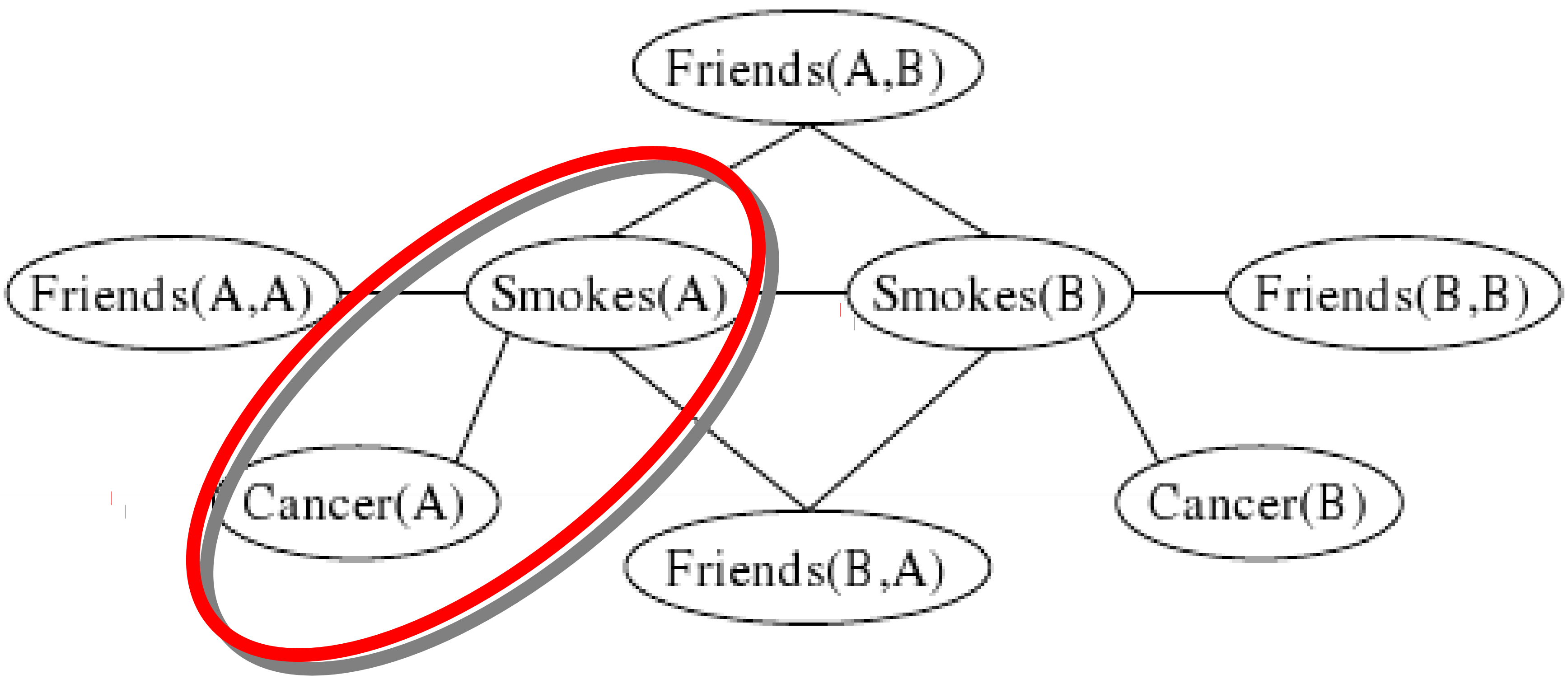
Example

- ∀xSmokes(x) ⇒ Cancer (x)
- ∀x∀yFriends(x, y) ⇒ (Smokes(x) ⇔ Smokes(y))
- Ground using Anna and Bob:



Example

- $\forall x Smokes(x) \Rightarrow Cancer(x)$
- $\forall x \forall y Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$
- Ground using Anna and Bob:



The System

1) Extract predicates from NLTK Propbank subset into databases
   - Certain features had to be calculated (e.g. finding constituent heads, finding paths between constituents)
2) Learn weights for predicates by giving Alchemy the databases and a FOL knowledge base
3) Do inference using Alchemy on test sentences
4) Evaluate results using script to calculate precision, recall and F_1.

Sample Predicates for SRL

Based on Meza-Ruiz and Riedel (2009) and Xue and Palmer (2004)

Hidden Predicates

- **isPredicate(*p*)**: *p* is the sentence's predicate
- **PredicateSense(*p, s*)**: predicate *p* has sense *s*
- **Role(*p, a, r*)**: constituent *a* has role *r* for predicate *p*

Observable Predicates

- **type*(a, t*)**: constituent *a* has type *t* (NP, VP, etc.)
- **path(*a1, a1, p*)**: path *p* leads from *a1* to *a2*
- **subcat(*a, e*)**: parent of constituent *a* uses expansion rule *e*
- **headword(a, o):** word at head of constituent *a* has POS *o*

Sample formulae

- Coarse “Part of speech” tag:
  - Cpos(p,+p_pos) ^ Cpos(a,+a_pos) => role(p, a, +r)
- Relative position of constituents:
  - Word(p, +p_w) ^ Word(a, +a_w) ^ position(p, a, “Left”) => role(p, a, +r)
- Lemma:
  - Lemma(p, +l) ^ Lemma(a, +l) => role(p, a, +r)

Many, many more…

Learning and Inference in Alchemy

<u>Learning</u>

- Need to learn formula weights
- Uses *Discriminative Weight Learning*
  - Set weights to maximize conditional probability of training set given the examples
- Can also do *Generative Weight Learning*
  - Maximizes pseudo-log likelihood
  - Generally less efficient than discriminative learning

<u>Inference</u>

- Get most probable values of hidden predicates given evidence
- Alchemy supports a variety of inference algorithms
- Default is *Lifted Belief Propagation*
  - Lifted inference exploits FOL properties for more efficient inference

Memory Issues in Learning

All computations done on quad-core 3.4 Ghz Intel Core i7-2600 CPU with 8 GB of RAM

- Running learning on 1000 training sentences caused the system to run out of memory while converting the formulae to Conjunctive Normal Form (prior to any actual learning)
- Ditto for 250
- 50 made it to MC-SAT phase before running out of memory
- Lazy inference and memory limiting flags made no appreciable difference

Memory Issues in Learning

- Running with 10 sentences did not cause the system to run out of memory
  - Did cause segmentation fault with no error message in the middle of the learning process.
- Segmentation fault did not occur when training on a single sentence.
  - Not a very effective classifier...
- Realized this weekend that using the less efficient generative learning avoids segfault.
- Ran on 10 sentences. It's still going...

Why?

- Markov Logic Networks grow exponentially:
  - Number of ground predicates in $O(d^c)$
    - d = maximum predicate arity
    - C = number of constants

- Problem compounded by Phrase Structure Grammar:
  - Dependency Grammar tree has exactly n nodes for n words
  - PSG tree has $O(2n - 1)$

- Constants also include number of unique paths in tree $(O(n^2))$, parts of speech, etc...

- Running on 10 sentences generated almost 200,000 ground clauses

Lessons learned

Alchemy may be unsuitable for SRL

- Claims “more than 10 million ground clauses on machines with 4GB of memory”

- Model problems?

- Meza-Ruiz and Riedel (2009) pulled it off
  - Didn't use Alchemy
  - Used dependecy grammar
  - No mention of training corpus size or learning time

- The amount of information being processed is not unreasonable for SRL

Future Directions

- Talk to Alchemy creators and ask where we went wrong
- Try a different Markov Logic engine like *Tuffy*
  - Better with lots of data?
- Examine original idea of using coarse-to-fine inference to reduce the model size by means of an existing labeler
  - *markov thebeast*

Questions?