Extracting Complex Relations

From (Biomedical) Texts

Based on:

McDonald et al. (2005): Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE

What Are Complex Relations?

UNIVERSITÄT DES SAARLANDES

- *n*-ary relations among *n* typed entities, (*n > 2)*
- *"John Smith is the CEO at Inc. Corp."*
  - Entities: John Smith CEO Inc. Corp.
  - Types: e.g. *person* *job* *company*
  - Relation: *(person, job, company)*
  - Instance: *(John Smith, CEO, Inc. Corp.)*
- Relation definition: tuple of types $(t_1, \ldots, t_n)$
- Relation instance: tuple of entities $(e_1, \ldots, e_n)$

Extracting Binary Relations

UNIVERSITÄT DES SAARLANDES

- A lot of research, works reasonably well
- Standard algorithm
  1. Train a binary classifier
     - Decide whether or not two entities are an instance of the relation.
  2. Enumerate all entity pairs
  3. Classify the pairs
- In principle also applicable to complex relations.

Why This Won't Work

UNIVERSITÄT DES SAARLANDES

- Enumeration of tuples
  - For binary relations: $O(|t_1|\ |t_2|)$ pairs of entities, where $|t|$ is the number of entities of type t.
    - => Polynomial complexity
  - For n-ary relations: $O(|t_1|\ |t_2|\ \ldots\ |t_n|) = O(m^n)$ tuples (potential instances), m being the smallest t_i.
    - => Exponential complexity

Another Issue

- Incomplete relation instances are sometimes desirable
  - *"Everyday John Smith goes to his office at Inc. Corp."*
  - Should yield *(John Smith,⊥, Inc. Corp.)*
  - Frequent occurrences in the training data.
- How to treat such instances during training?
  - Positive instance? Negative? Simply ignore?

General Idea Of McDonald et al.

UNIVERSITÄT DES SAARLANDES

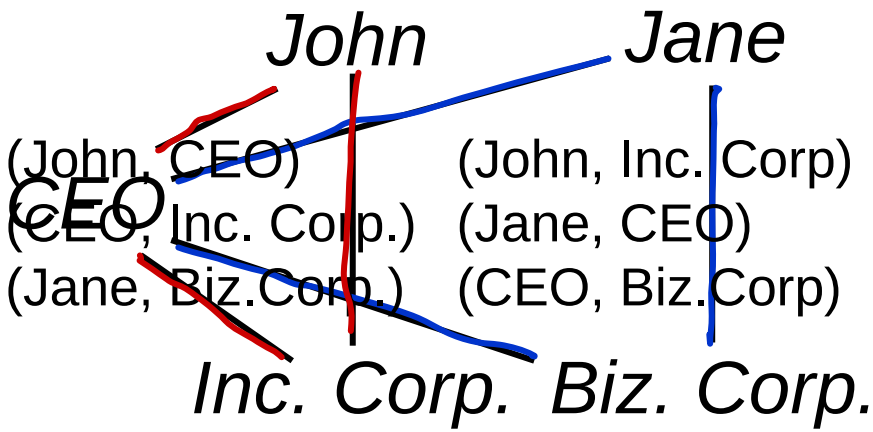
- Binary relation extraction is good, so why not use it?
  - Factor complex relations into sets of binary relations.

(person, job, company) => *(person, job)*, *(person, company)*, *(job, company)*

  - Also factorize training instances.
  - Train binary classifier for relatedness.

General Idea (2)

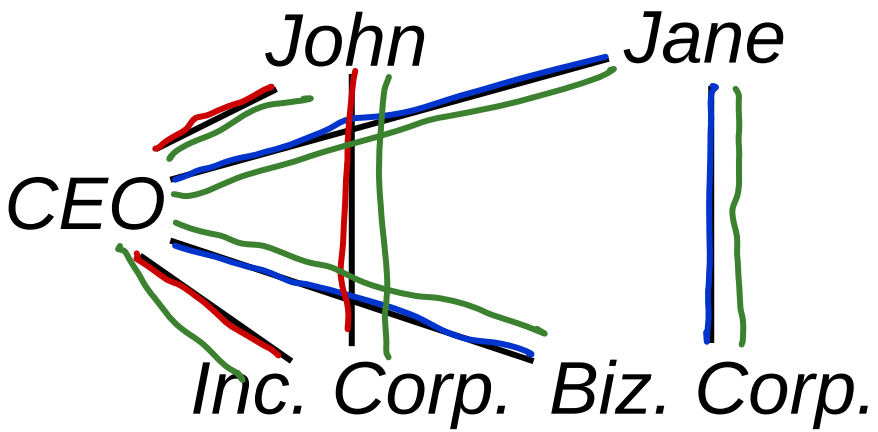
*"**John** and **Jane** are **CEOs** at **Inc. Corp.** and **Biz. Corp.** respectively"*



- Enumerate all pairs of entities (number is $O(n^2)$) and classify as related or not.
- Construct graph with entities as nodes and edges between related entities.
- Reconstruct complex relations from that graph.

Finding Maximal Cliques

- Clique: part of a graph, where each node has edges to every other node in the graph.
  - Graph terms: *complete subgraph*



  - *Maximal* clique: you know this by now

Finding Maximal Cliques (2)

UNIVERSITÄT DES SAARLANDES

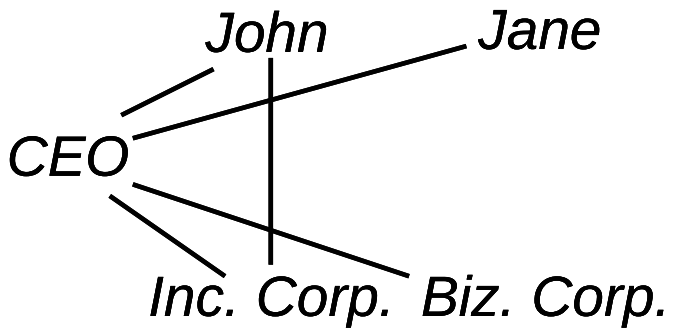
- Cliques in this graph:
  - e.g. {*John*}, {*Jane*}, …
    {*John, CEO*}, {*Jane, Biz. Corp.*}, …
    {*John, CEO, Inc. Corp.*}, …
  - Only cliques consistent with the relation schema are considered. (Not: {John, Jane, CEO})
- E.g.:
  - {*John, CEO, Inc. Corp.*} => *(John, CEO, Inc. Corp.)*
  - {*John, CEO*} => *(John, CEO,* ⊥ *)*

Problems

UNIVERSITÄT DES SAARLANDES

- Number of cliques in an arbitrary graph may be exponential.
  - Would negate the runtime complexity gain over simple enumeration of possible relation instances.
  - Solution: Bron & Kerbosch => *Michaela's talk*
- Errors of the binary classifier in borderline cases result in relation instances not being found.

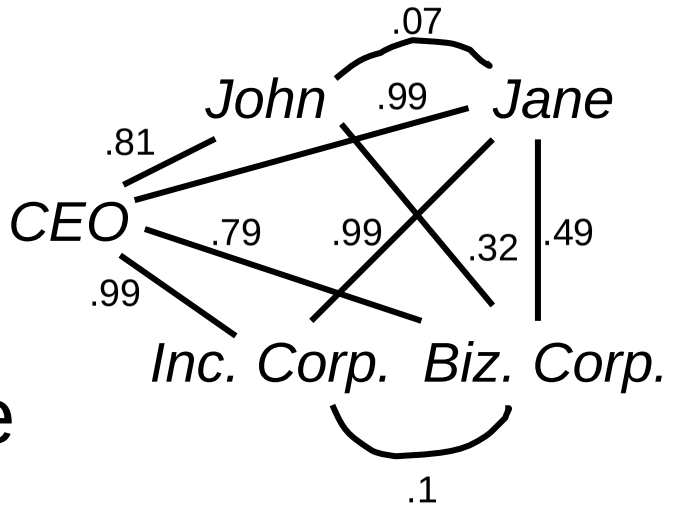
Probabilistic Cliques



*"**John** and **Jane** are **CEOs** at **Inc. Corp.** and **Biz. Corp.** respectively"*

- Classifier output might be:
  - P(Jane, Biz. Corp.) = 0.49
    P(Jane, CEO) = 0.99
    P(CEO, Biz. Corp.) = 0.99
- Idea: Cliques should *on average* consist of valid binary relations.

Probabilistic Cliques (2)



- Construct a *complete* graph with the output of the classifier as edge weights.
- Define the weight of a clique *w(C)* as the mean of its edges' weights.
- *w(C)* ≥ 0.5 => C is a valid relation instance
- Unfortunately they don't tell us exactly how it is done…

Evaluation

- Task: Extraction of genomic variation events from biomedical text.
  - Schema:
    (var-type, location, initial-state, altered-state)
- ~ 4700 sentences from MEDLINE abstracts
- 4773 entities and 1218 relation instances
- Relatively small dataset => 10-fold cross-validation
- 4% of instances cross sentence boundaries => upper bound for accuracy: 96%

Systems Overview

UNIVERSITÄT DES SAARLANDES

- **MC**: Binary relations classification followed by maximum clique finding to construct complex relations.
- **PC**: As MC, but using the probabilistic clique method.
- **NE**: Naïve enumeration and classification of all possible relation instances.
  - For NE, incomplete but correct instances were marked as positive during training.

Results: Binary Classifier

UNIVERSITÄT DES SAARLANDES

- Maximum entropy binary classifier with features like entity types, POS, distance in the text, words in between the entities, …
- Quite good, but could be improved.
- Authors hope that using the position of entities in parse trees will improve result.

ACT	PRD	COR
2577	2722	2101
Prec	**Rec**	**F-Meas**
0.7719	0.8153	0.7930

Results: Actual Extraction Task

System	Prec	Rec	F-Meas
NE	0.4588	0.6995	0.5541
MC	0.5812	0.7315	0.6480
PC	0.6303	0.7726	0.6942

- No partial credit: all entities must be correct
- Time taken:
  - MC and PC: under 5 min., NE: ~ 26 min.
- Gains especially in precision.
  - Error rate reduction by 21% for MC and 31% for PC compared to NE

Results: Probabilistic Cliques

- PC improves recall on larger relation instances.

System	2-ary	3-ary	4-ary
NE	760:1097:600	283:619:192	175:141:60
MC	760:1025:601	283:412:206	175:95:84
PC	760:870:590	283:429:223	175:194:128

Actual:Predicted:Correct

Summary

- Factoring complex relations into sets of binary relations has shown to facilitate the process of extracting relation instances.
- Constructing complex relation instances through maximum clique finding significantly improves performance of naïve enumeration.
- Using probabilistic cliques helps finding larger relation instances.

That's all folks…

UNIVERSITÄT DES SAARLANDES

Thank you!

Weight Of A Clique

$$w(C) = \left(\prod_{e \in E_C} w(e) \right)^{1/|E_C|}$$

C:	clique
e:	edge
E_C:	set of edges in C
$w(e)$:	edge weight

10-fold Cross-validation

- Divide your data set into 10 parts.
- In each of 10 steps, choose one of the parts to be used as the test set, the other parts form the training set.
- When finished average the results.