

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
oooooo  
oooooo  
oooo

# Open Information Extraction

Dominik Both, Tonio Weidler

Proseminar *Text Mining*  
Andrea Zielinski

Institut für Computerlinguistik, Universität Heidelberg, 15.07.2016

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo

## Papers

Identifying Relations for Open Information Extraction (Fader et al., 2011)

LODifier: Generating Linked Data from Unstructured Text (Augenstein et al., 2012)

# Strukturierung

- 1 Introduction to Information Extraction
- 2 OIE - Principles
- 3 Example: LODifier
- 4 Conclusion

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo

# Introduction to Information Extraction

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo

# What is Information Extraction?

## Information Extraction

Goal of Information Extraction is automatically extracting information from unseen text

*Information:* entities, relations, events...

To make the dough for a good pizza, we start with putting 1kg of flour into the mixing bowl.  
(1kg of flour, put into, mixing bowl)

ooo  
oooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oooo

# Problems of Information Extraction

- Named Entity Recognition
- Relationship Extraction
- Coreference Resolution
- Comment Extraction
- many more..

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo  
oooo

# OIE - Principles

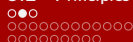
●○○  
○○○○○○○○○○○○  
○○○○○○○○○

○○○○  
○○○○○○○○  
○○○○○  
○○○○○  
○○○○○  
○○○○

# OIE - Principles

## Open Information Extraction





# Open Information Extraction

*IE*: Extractor for each target relation

*Open*: No pre-specified extractors

Unsupervised learning of relation phrases

Extraction of information on every given domain



# Problems of Open Information Extraction

## ■ Incoherent extractions:

This guide contains dead links and omits sites  
*contains omits*

## ■ Uninformative extractions:

Faust made a deal with the devil  
*(Faust, made, a deal)*

○○○  
●○○○○○○○○○○○  
○○○○○○○○○

○○○○  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○○  
○○○○

# OIE - Principles

## Methods



# Text Runner and WOE

- 1 *Label*: Automatic sentence labeling by heuristics
- 2 *Learn*: A relation phrase extractor is learned
- 3 *Extract*: Identifying NP pairs and searching relations words between

○○○  
○○●○○○○○○○○○  
○○○○○○○○○

○○○○  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○

# Problems

- Large number of labeled training examples required
- Alternative heuristic labeling leads to huge noise and stacked uncertainty
- Ignores both holistic and lexical aspects



## Syntactic constraint

- Limits relations to those matching a certain POS Tag pattern:
- Always chooses longest possible match
- Merge adjacent matches together

$V \mid V P \mid V W^* P$

V = verb particle? adv?

W = (noun | adj | adv | pron | det)

P = (prep | particle | inf. marker)

*Faust made a deal with the devil*

○○○  
○○○○●○○○○○○○  
○○○○○○○○○

○○○○  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○

# Lexical constraint

- Only assume relations that appear in the corpus for a certain amount
- The Obama administration is **offering only modest greenhouse gas reduction targets** at the conference

ooo  
 ooooo●ooooo  
 ooooooooo

oooo  
 ooooooooo  
 ooooo  
 ooooo  
 oooo

## Limitations of those constraints

- In a set of 300 hand-annotated sentences 85% relations fell into those constraints
- Model is not complete and has its flaws
- Of all relation phrases in the gold standard:
  - 85% satisfied constraints and were found
  - 8% Non-contiguous phrase structure
    - Coordination*: X is produced and maintained by Y
    - Multiple Args*: X was founded in 1995 by Y
    - Phrasal Verbs*: X turned Y off
  - 4% Relation phrase not between arguments
    - Introductions*: Discovered by Y, X ...
    - Relative Cl.*: ... the Y that X discovered
  - 3% Not matching POS pattern
    - Interrupting Modifiers*: X has a lot of faith in Y
    - Infinitives*: X to attack Y





# ReVerb Extraction Algorithm

- *Relation Extraction*: Find the longest possible string of words that match the relation constraints, merge adjacents
- *Argument Extraction*: Find the nearest NP left and right to the relation that is not a relativ pronoun, WHO-adverb or existential-there.
- How is the lexical constraint being checked? By creating a list of relational phrases by applying this algorithm on a 500 million Web sentences.



# ReVerb Confidence Function

- The Algorithm has a high recall, but low precision
- Now the extracted relation is weighted by a confidence function:

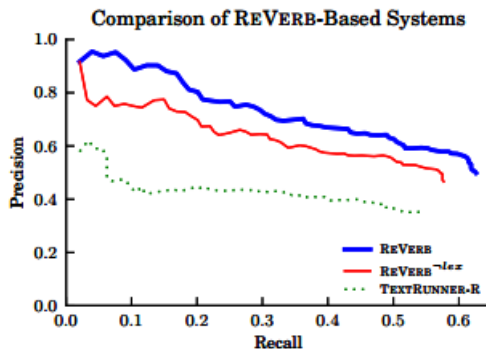
Examples:

- 1.16  $(x, r, y)$  covers all words in  $s$
- 0.50 The last preposition in  $r$  is for
- 0.43  $\text{len}(s)$  under 10
- -0.65 There is a preposition to the left of  $x$  in  $s$
- ...

○○○  
 ○○○○○○○○●○○○  
 ○○○○○○○○

○○○○  
 ○○○○○○○○  
 ○○○○○○  
 ○○○○○○  
 ○○○○

# Evaluation

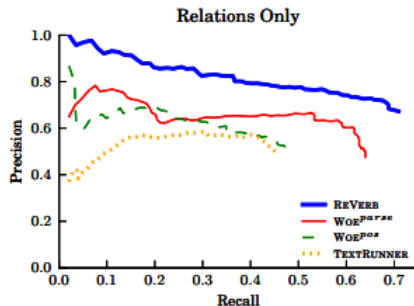
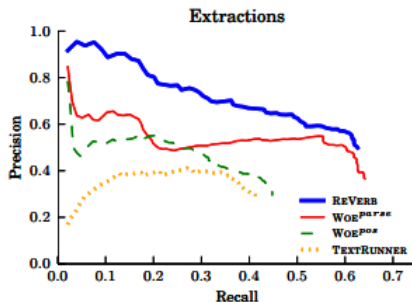


Better results than TextRunner through lexical features, but still low recall.



## Methods

## Why?



Argument extraction is open to improvements

Source: Fader et al., 2011



# Why?

## Evaluating the evaluation:

<b>REVERB - Incorrect Extractions</b>	
65%	Correct relation phrase, incorrect arguments
16%	N-ary relation
8%	Non-contiguous relation phrase
2%	Imperative verb
2%	Overspecified relation phrase
7%	Other, including POS/chunking errors

<b>REVERB - Missed Extractions</b>	
52%	Could not identify correct arguments
23%	Relation filtered out by lexical constraint
17%	Identified a more specific relation
8%	POS/chunking error

Source: Fader et al., 2011

○○○  
○○○○○○○○○○○●  
○○○○○○○○○

○○○○  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○○  
○○○○○

# Conclusion

Acceptable results on easy sentences. Fails on more complex sentences. *Possible points of improvement:*

- Does not support n-ary relations
- Relations seem overly specific in many results
- Coordination is not in all cases correctly recognized
- Coreference is not predicted by the system
- Syntactic fixation on S-V-O word order

○○○  
○○○○○○○○○○○○○  
●○○○○○○○○

○○○○  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○

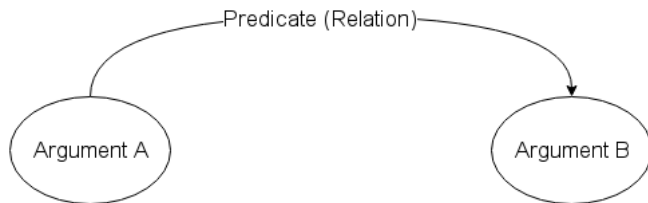
# OIE - Principles

## Data Representation

ooo  
oooooooooooooooo  
o●oooooooo

oooo  
oooooooooo  
ooooo  
ooooo  
oooo

# Standard Patterns



**Argument A** is in a directed relation to **Argument B**.



ooo  
oooooooooooo  
oo●oooooo

oooo  
oooooooooo  
oooooo  
oooooo  
oooooo  
oooo

# Unnormalized Annotation

(argument\_a, predicate\_x, argument\_b)  
(argument\_a, predicate\_y, argument\_c)  
(argument\_a, predicate\_y, argument\_d)

## Problems

- redundant
- unnormalized
- can only produce binary predicates

```
ooo
ooooo
oooooooooooo
ooo●ooooo
```

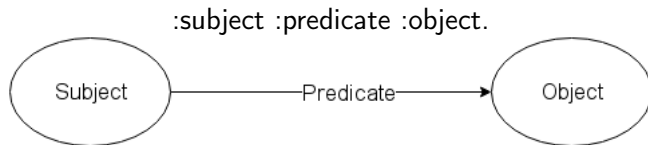
```
oooo
ooooo
oooooooo
ooooo
ooooo
ooooo
oooo
```

# RDF and Linked Data

## Resource Description Framework

Models propositions by constructing *triples* including **Subjects**, **Objects** and **Predicates**

Generates a directed graph





# RDF Concepts and Notation

- **URIs**  
identifies resources (S, R, O) distinctively and references further informations (triples)
- **Conclusions**  
allows to draw conclusions using rules
- **Turtle**  
allows syntax abbreviations
- **Blanknodes**  
placeholder for something without a URI
- **Queries**  
can be searched by querying (eg SPARQL)

ooo  
oooooooooooooooo  
oooo●ooo

oooo  
oooooooooo  
ooooo  
ooooo  
ooooo  
oooo

# RDF Reification

**Motivation:** How can I realize embedded propositions?

**Example:** Peter said, he watched the movie.

ooo  
 ooooooooooooo  
 ooooo●ooo

oooo  
 ooooooooo  
 ooooo  
 ooooo  
 oooo

# RDF Reification

**Motivation:** How can I realize embedded propositions?

**Example:** Peter said, he watched the movie.

**Wrong proposition**

:Peter :watched :movie



# RDF Reification

**Motivation:** How can I realize embedded propositions?

**Example:** Peter said, he watched the movie.

## Reification

```
:Peter :said __:prop.  
__:prop rdf:subject :Peter.  
__:prop rdf:predicate :watched.  
__:prop rdf:object :movie.
```

ooo  
ooooooooooooo  
oooooooo●oo

oooo  
oooooooooooo  
ooooo  
ooooo  
ooooo  
oooo

# Vocabularies & Ontologies

Several vocabularies provide useful relations and functionality, eg.:

- RDF (rdf:type, ...)
- RDFS (rdfs:subClassOf, rdfs:domain, rdfs:range, ...)
- OWL (owl:sameAs, owl:SymmetricProperty, ...)
- FOAF

Ontologies are huge RDF Graphs containing many triples, eg.:

- DBpedia
- Wikidata
- WordNet

ooo  
ooooooooooooo  
oooooooo●o

oooo  
ooooooooo  
ooooo  
ooooo  
ooooo  
oooo

# RDF Syntax

```
dbr:Barack_Obama a foaf:person, :President;  
    dbo:spouse dbr:Michelle_Obama.  
dbr:Bernie_Sanders dbo:birthPlace dbr:New_York,  
    dbr:Brooklyn;  
dbr:Brooklyn dbo:isPartOf dbr:New_York
```

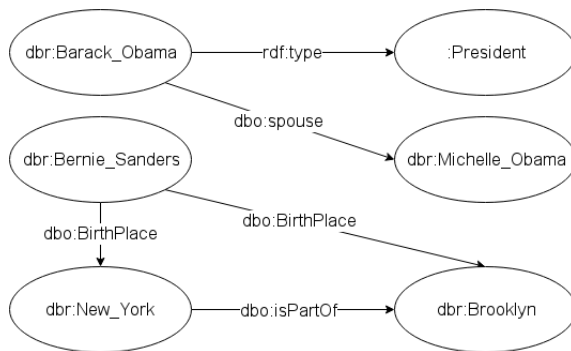


ooo  
oooooooooooooooo  
ooooooooo●

oooo  
oooooooooo  
oooooo  
ooooo  
ooooo  
oooo

## Data Representation

## ... as Graph



ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo  
oooo

# Example: LODifier

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
oooooo  
oooooo  
oooo

# LODifier: Generating Linked Data from Unstructured Text (Augenstein et al., 2012)

Generate an RDF Graph from unstructured Text

**Past Approaches:** Use Patterns to trade recall for precision

**LODifier:** Process the entire text

ooo  
ooooooooooooo  
oooooooooooo

●ooo  
oooooooooooo  
ooooo  
ooooo  
oooo

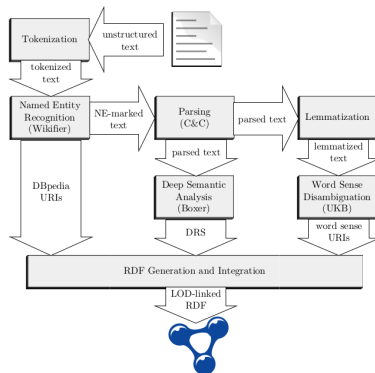
## Example: LODifier Architecture

○○○  
 ○○○○○○○○○○○○  
 ○○○○○○○○

●○○○  
 ○○○○○○○○  
 ○○○○  
 ○○○○  
 ○○○○

## Architecture

## Architecture



(Augenstein et al., 2012)

○○○  
○○○○○○○○○○○○○  
○○○○○○○○○

○○●○  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○

# Approach

- 1 **Parse** the input text (POS, Treetagging, NER)
- 2 Apply **Deep Semantic Analysis** to get relations
- 3 Enrich NEs and words with **URIs** (DBpedia and WordNet)
- 4 Forge an **RDF Graph** of this information

○○○  
○○○○○○○○○○○○○  
○○○○○○○○○

○○○●  
○○○○○○○○○  
○○○○○  
○○○○○  
○○○○

## Architecture

# How does it happen?

Lets go through the process step-by-step!

## Example Text:

The New York Times reported that John McCarthy died. He invented the programming language LISP.

example taken from Augenstein et al., 2012

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
●ooooooooo  
ooooo  
ooooo  
ooooo  
oooo

# Example: LODifier Preprocessing





# Named Entity Recognition - Wikifier

## Wikifier

Recognizes NE and replaces them with the Wikipedia Page Link  
Disambiguates by comparing links between pages.

### Example Text Output:

[The New York Times] reported that [John McCarthy (computer scientist)|John McCarthy] died. He invented the [Programming language|programming language] [Lisp (programming language)|LISP].



# Parsing Syntax - C&C

## C&C Parser

Syntactical Parser that tags POS and builds Parse Trees in CCG.

## Combinatory Categorical Grammar (CCG)

Grammatical formalism allows parallel analysis of syntax and semantics

Associates words with categories that can be combined (rule-based) to form a sentence

Syntax via Category Combination, Semantics via lambda calculus

```

ooo
ooooo
oooooooooooo
oooooooooo

```

```

oooo
ooo●oooo
ooooo
ooooo
ooooo
oooo

```

## Preprocessing

## Parsing - Output

```

ccg(1, rp(s:dcl,
  ba(s:dcl,
    lx(np, n,
      t(n, 'The_New_York_Times', 'The_New_York_Times', 'NNS', 'I-NP', 'O')),
    fa(s:dcl\np,
      t((s:dcl\np)/s:em, 'reported', 'report', 'VBD', 'I-VP', 'O')),
    fa(s:em,
      t(s:em/s:dcl, 'that', 'that', 'IN', 'I-SBAR', 'O')),
    ba(s:dcl,
      lx(np, n,
        t(n, 'John_McCarthy', 'John_McCarthy', 'NNP', 'I-NP', 'I-PER')),
        t(s:dcl\np, 'died', 'die', 'VBD', 'I-VP', 'O'))))))),
  t(period, '.', '.', '.', 'O', 'O'))).
ccg(2, rp(s:dcl,
  ba(s:dcl,
    t(np, 'He', 'he', 'PRP', 'I-NP', 'O'),
    fa(s:dcl\np,
      t((s:dcl\np)/np, 'invented', 'invent', 'VBD', 'I-VP', 'O'),
      fa(np:nb,
        t(np:nb/n, 'the', 'the', 'DT', 'I-NP', 'O'),
        fa(n,
          t(n/n, 'programming_language', 'programming_language', 'NN', 'I-NP', 'O'),
          t(n, 'LISP', 'LISP', 'NNP', 'I-NP', 'O'))))))),
  t(period, '.', '.', '.', 'O', 'O'))).

```

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooo●ooo  
ooooo  
ooooo  
ooooo  
oooo

## Preprocessing

## Find Relations - Boxer

## Boxer

Creates DRSs from C&C Output

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooo●ooo  
ooooo  
ooooo  
ooooo  
oooo

## Find Relations - Boxer

### Boxer

Creates DRSs from C&C Output

### Discours Representation Structure (DRS)

Represents the discourse via *relations* between *entities*  
Allows referencing over the entire discourse

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooo●ooo  
ooooo  
ooooo  
ooooo  
oooo

## Find Relations - Boxer

### Boxer

Creates DRSs from C&C Output

### Discours Representation Structure (DRS)

Represents the discourse via *relations* between *entities*  
Allows referencing over the entire discourse

### Boxers DRS Relations (Conditions):

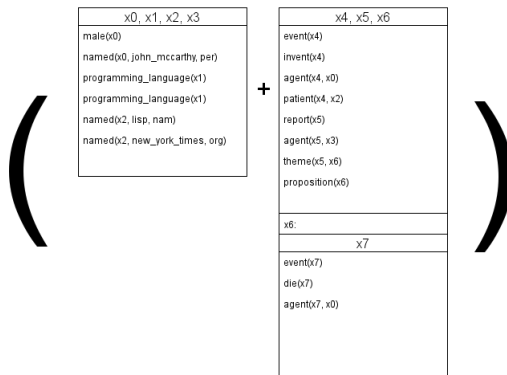
- **Unary Relations (Classes):** eg. *topic*, *person*, *event*, *male*, ...  
+ all verbs
- **Binary Relations:** agent, patient, ... (semantic roles)

○○○  
 ○○○○○○○○○○○○  
 ○○○○○○○○○

○○○○  
 ○○○○○●○○  
 ○○○○  
 ○○○○  
 ○○○○  
 ○○○○

## Preprocessing

## Boxer Output



ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooo●o  
ooooo  
ooooo  
ooooo  
oooo

# Assign WordNet URIs

## RDF WordNet

**WN:** Lexicography containing senses linked by semantic relations

**RDF WN:** LD Representation of WN providing URIs for words

### Steps:

- 1 Lemmatization
- 2 WSD (UKB)
- 3 Assign RDF WN URIs to word senses



ooo  
ooooooooooooo  
oooooooooooo

oooo  
ooooooooo●  
ooooo  
ooooo  
ooooo  
oooo

## Preprocessing

# Preprocessing Result

**We now have ...**

- URIs for all NEs
- URIs for all (disambiguated) words
- Relations between entities (those URIs)

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
●oooo  
ooooo  
oooo

# Example: LODifier

## RDF Construction

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
o●oooo  
ooooo  
oooo

# What now?

Let's now construct the RDF Graph from this information!



# Namespaces/Vocabularies

LODifier introduces several namespaces:

- **drsclass**: contains Boxer classes (event, person, ...) and :named relation
- **drsrel**: contains Boxer relations (agent, patient, ...)
- **ne**: contains the named entity URLs
- **reify**: reification (embedding propositions into propositions)

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
ooo●ooo  
ooooo  
ooooo

# Namespaces/Vocabularies

And uses standard namespaces:

- **rdf:** mainly for `rdf:type` and reification
- **owl:** for `owl:sameAs`

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
ooo●ooo  
ooooo  
oooo

# Namespaces/Vocabularies

Finally the two ontologies:

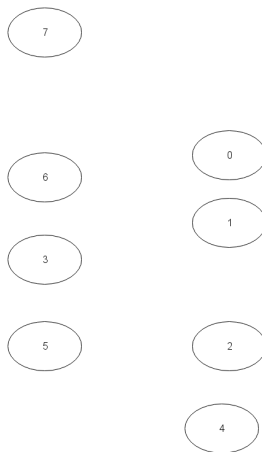
- **wn30**: contains all WordNet URIs
- **dbpedia**: contains the dbpedia URIs
- **class**: contains classes not in wn30 nor in dbpedia



# RDF Construction Strategy I

Create a blanknode `_:x` for each discourse referent (`x0`, `x1`, ...)

# RDF Construction Strategy II





# RDF Construction Strategy III

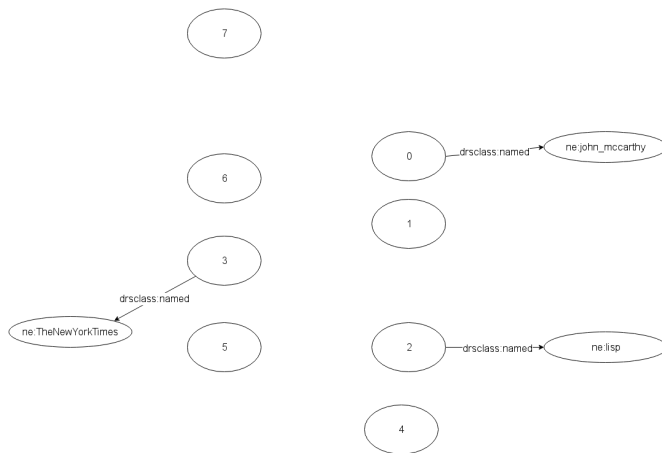
if NE, then create

`_:x drsclass:named ne:URI`



## RDF Construction

## RDF Construction Strategy IV

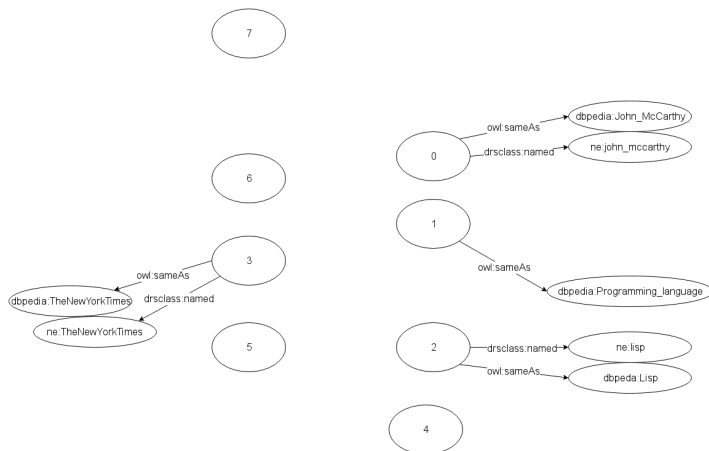


# RDF Construction Strategy V

if NE and DBpedia URI exists create  
\_\_ :x owl:sameAs dbpedia:URI

## RDF Construction

## RDF Construction Strategy VI





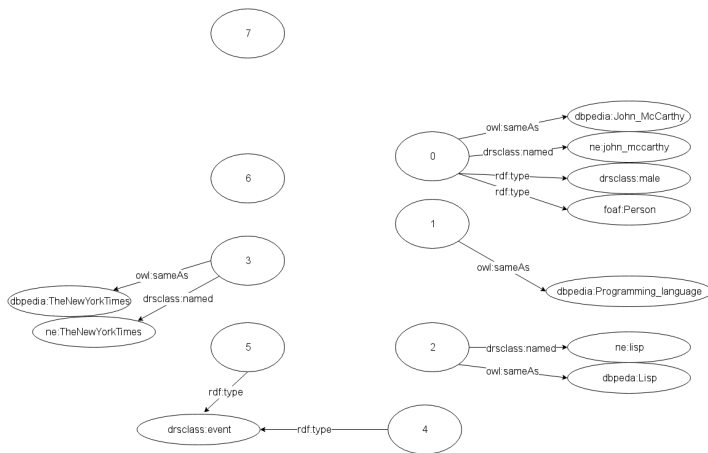
# RDF Construction Strategy VII

via `rdf:type` assign closed classes (event, person, ...)  
`_:x rdf:type drsclass:CLOSEDCLASS`



## RDF Construction

## RDF Construction Strategy VIII





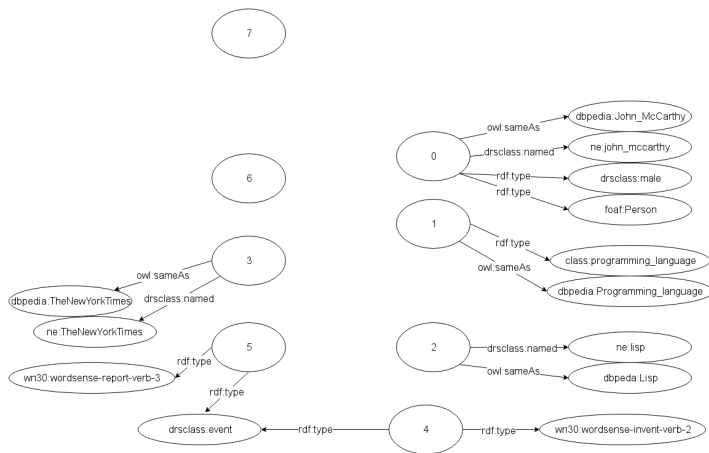
# RDF Construction Strategy IX

via `rdf:type` assign open classes (die, programming\_language, ...)  
`_:x rdf:type wn30:OPENCLASS, class:OPENCLASS`



## RDF Construction

## RDF Construction Strategy X







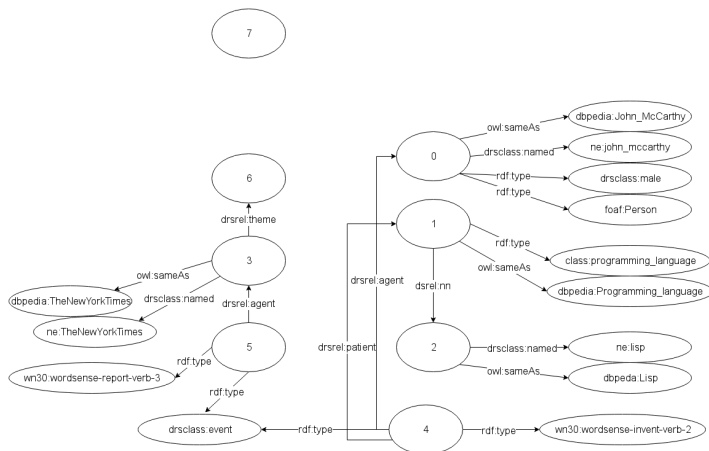
# RDF Construction Strategy XI

create triples from binary relations (agent, theme, ...)

`_:x drsrel:RELATION _:y`

## RDF Construction

## RDF Construction Strategy XII



# RDF Construction Strategy XIII

recursive reification of embedded propositions (eg. by *report* or *says*)

## RDF Construction Strategy XIV



```

ooo
oooooooooooooooo
oooooooooooo

```

```

oooo
oooooooooooo
oooo●●
ooooo
oooo

```

## RDF Construction

## RDF Construction: Output

```

_:var0x0 drsclass:named ne:john_mccarthy ;
        rdf:type drsclass:male , foaf:Person ;
        owl:sameAs dbpedia:John_McCarthy_(computer_scientist) .
_:var0x1 rdf:type class:programming_language ;
        owl:sameAs dbpedia:Programming_language .
_:var0x2 drsrel:nn _:var0x1 .
_:var0x2 drsclass:named ne:lisp ;
        owl:sameAs dbpedia:Lisp_(programming_language) .
_:var0x3 drsclass:named ne:the_new_york_times ;
        owl:sameAs dbpedia:The_New_York_Times .
_:var0x4 rdf:type drsclass:event , wn30:wordsense-invent-verb-2 .
        drsrel:agent _:var0x0 ; drsrel:patient _:var0x2 .
_:var0x5 rdf:type drsclass:event , wn30:wordsense-report-verb-3 ;
        drsrel:agent _:var0x3 ; drsrel:theme _:var0x6 .
_:var0x6 rdf:type drsclass:proposition , reify:proposition , reify:conjunction ;
        reify:conjunct [ rdf:subject _:var0x7 ;
                        rdf:predicate rdf:type ;
                        rdf:object drsclass:event . ]
        reify:conjunct [ rdf:subject _:var0x7 ;
                        rdf:predicate rdf:type ;
                        rdf:object wn30:wordsense-die-verb-1 . ]
        reify:conjunct [ rdf:subject _:var0x7 ;
                        rdf:predicate drsrel:agent ;
                        rdf:object _:var0x0 . ]

```

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
●oooo  
oooo

## Example: LODifier Experiments

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
o●oooo  
oooo

# Method

- Evaluate by testing similarity of two given documents:
- *the problem of deciding whether two randomly selected stories discuss the same news topic* (Augenstein et al., 2012)
- TDT-2 benchmark dataset: 84.000 news documents
- Extract 183 positive and 183 negative pairs (avg. 11.2 per topic)
- Calculate similarity and evaluate the system on the result

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oo●ooo  
oooo

# Similarity measurements

- 1 NEs identified by Wikifier and successfully disambiguated words in UKB
- 2 add NEs recognized by Boxer
- 3 add URIs of unrecognized words

*Further add structural features:* Measurement of the similarity of the RDF graphs of the two documents



# Accuracy

Model	normal	extended
<b>Similarity measures without structural knowledge</b>		
Random Baseline	50.0	–
Bag of Words	63.0	–
Bag of URIs (Variant 1)	61.6	75.1
Bag of URIs (Variant 2)	70.6	76.0
Bag of URIs (Variant 3)	<b>73.4</b>	<b>76.4</b>
<b>Similarity measures with structural knowledge</b>		
proSim <sub>cnt</sub> (k=8, Variant 1)	77.7	77.6
proSim <sub>cnt</sub> (k=8, Variant 2)	79.2	79.0
proSim <sub>cnt</sub> (k=8, Variant 3)	<b>82.1</b>	<b>81.9</b>

Source: Augenstein et al., 2012

# Precision - Recall

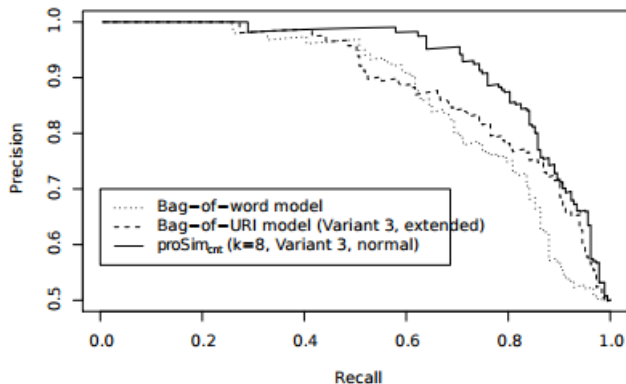


Fig. 5. Precision-Recall-plot for best Story Link Detection models

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
●oooo

## Example: LODifier

# Conclusions

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
oooooo  
oooooo  
oooooo  
o●ooo

## Conclusions

# What to draw from this?

- deep semantic analysis can work for OIE
- combining several existing NLP Systems provides a well functioning extraction system
- information gained by unsupervised OIE can already improve real world tasks

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oo●oo

## Conclusions

# What we liked

- full-text OIE

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oo●oo

## Conclusions

# What we liked

- full-text OIE
- uses many strengths of RDF

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oo●oo

## Conclusions

# What we liked

- full-text OIE
- uses many strengths of RDF
- relations aren't overspecified

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oo●oo

## Conclusions

# What we liked

- full-text OIE
- uses many strengths of RDF
- relations are not overspecified
- extendable/improvable by improving/swapping Systems in the architecture



ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oo●oo

# What we liked

- full-text OIE
- uses many strengths of RDF
- relations are not overspecified
- extendable/improvable by improving/swapping Systems in the architecture
- part of the LOD-Cloud

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oo●oo

# What we liked

- full-text OIE
- uses many strengths of RDF
- relations are not overspecified
- extendable/improvable by improving/swapping Systems in the architecture
- part of the LOD-Cloud
- results in standardized notation

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
oooooo  
oooooo  
ooooo  
oo●oo

## Conclusions

# What we liked

- full-text OIE
- uses many strengths of RDF
- relations are not overspecified
- extendable/improvable by improving/swapping Systems in the architecture
- part of the LOD-Cloud
- results in standardized notation
- domain-independent

# What we didnt like

- Redundant processes like NER

# What we didnt like

- Redundant processes like NER
- BlankNode Massacre

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
ooooo  
oooo●

## Conclusions

# What we didnt like

- Redundant processes like NER
- BlankNode Massacre
- confusing boxer relations not simplified for RDF (will be hard to search through)

ooo  
oooooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
ooooo  
ooooo  
ooo●

## Conclusions

# What we didnt like

- Redundant processes like NER
- BlankNode Massacre
- confusing boxer relations not simplified for RDF (will be hard to search through)
- Paper scratches only the surface of the system

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo●

## Conclusions

# What we didnt like

- Redundant processes like NER
- BlankNode Massacre
- confusing boxer relations not simplified for RDF (will be hard to search through)
- Paper scratches only the surface of the system
- Some points are unclear / not even described



# Conclusion

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo  
oooo

# Conclusion Assessing OIE

# Weaknesses and Strengths of OIE

- trades precision for recall

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
oooooo  
oooooo  
oooo

## Assessing OIE

# Weaknesses and Strengths of OIE

- trades precision for recall
- $OIE > IE$  if no sepcial task/domain is defined

ooo  
ooooooooooooo  
ooooooooooooo

oooo  
ooooooooooooo  
oooooo  
oooooo  
oooo

## Assessing OIE

# Weaknesses and Strengths of OIE

- trades precision for recall
- $OIE > IE$  if no special task/domain is defined
- theory independent

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo  
oooo

## Weaknesses and Strengths of OIE

- trades precision for recall
- $OIE > IE$  if no special task/domain is defined
- theory independent
- relations may be redundant/overspecified/unintended

ooo  
oooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo

## Weaknesses and Strengths of OIE

- trades precision for recall
- $OIE > IE$  if no special task/domain is defined
- theory independent
- relations may be redundant/overspecified/unintended
- restricted usability of results due to low precision

# Future Opportunities

- better subsystems



# Future Opportunities

- better subsystems
  - Coreference Resolution

# Future Opportunities

- better subsystems
  - Coreference Resolution
  - NER

# Future Opportunities

- better subsystems
  - Coreference Resolution
  - NER
  - Disambiguation

# Future Opportunities

- better subsystems
  - Coreference Resolution
  - NER
  - Disambiguation
- improve semantic analysis

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooooo  
oooo

# Conclusion References

ooo  
ooooooooooooo  
oooooooooooo

oooo  
ooooooooo  
ooooo  
ooooo  
ooooo  
oooo

# References I

A. Fader, S. Soderland, O. Etzioni. Identifying relations for open information extraction. Proc. of the Conf. on Empirical Methods in Natural Language, 2011

Augenstein, Isabelle, Sebastian Padó, and Sebastian Rudolph. Lodifier: Generating linked data from unstructured text. The Semantic Web: Research and Applications. Springer Berlin Heidelberg, 2012. 210-224.

ooo  
ooooooooooooo  
oooooooooooo

oooo  
oooooooooooo  
oooooo  
oooooo  
oooo

## References II

James R. Curran, Stephen Clark, and Johan Bos. 2007.  
Linguistically motivated large-scale NLP with C&C and boxer. In  
Proceedings of the 45th Annual Meeting of the ACL on Interactive  
Poster and Demonstration Sessions (ACL '07). Association for  
Computational Linguistics, Stroudsburg, PA, USA, 33-36.

Unpublished draft of Andreas Harths Linked Data"Book (Version  
February 2016) as provided in his Linked Data lecture at Heidelberg  
University.