

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
oooo
oooo

Open Information Extraction

Dominik Both, Tonio Weidler

Proseminar *Text Mining*
Andrea Zielinski

Institut für Computerlinguistik, Universität Heidelberg, 15.07.2016

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
oooo
oooo
oooo

Papers

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

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

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
oooo
oooo

Strukturierung

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

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
oooo
oooo
oooo

Introduction to Information Extraction

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
oooo
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
oooo
oooo
oooo

Problems of Information Extraction

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

ooo
oooooooooooooooo
oooooooooooo

oooo
oooooooooooo
oooooo
oooo
oooo

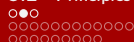
OIE - Principles

●○○
○○○○○○○○○○○○
○○○○○○○○○

○○○○
○○○○○○○○
○○○○○○
○○○○○
○○○○
○○○○

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

○○○
○○○●○○○○○○○○
○○○○○○○○○

○○○○
○○○○○○○○○
○○○○○
○○○
○○○

Methods

Syntactic constraint

- Limits relations to those matching a certain POS Tag pattern:
- $V \mid VP \mid VW^*P$
- Always chooses longest possible match
- Merge adjacent matches together

$$V \mid VP \mid VW^*P$$

V = verb particle? adv?

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

P = (prep | particle | inf. marker)

ooo
oooo●oooooooo
ooooooooo

oooo
ooooooooo
ooooo
oooo
oooo
oooo

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

```

○○○
○○○○○●○○○○○
○○○○○○○○○

```

```

○○○
○○○○○○○
○○○○○
○○○
○○○
○○○

```

Methods

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

| Binary Verbal Relation Phrases | |
|--------------------------------|--|
| 85% | Satisfy Constraints |
| 8% | Non-Contiguous Phrase Structure Coordination: X <u>is produced</u> and maintained <u>by</u> Y Multiple Args: X <u>was founded</u> in 1995 <u>by</u> Y Phrasal Verbs: X <u>turned</u> Y <u>off</u> |
| 4% | Relation Phrase Not Between Arguments Intro. Phrases: <u>Discovered by</u> Y, X ... Relative Clauses: ... the Y that X <u>discovered</u> |
| 3% | Do Not Match POS Pattern Interrupting Modifiers: X <u>has a lot of faith in</u> Y Infinitives: X <u>to attack</u> 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.

ooo
oooooooo●oooo
ooooooooo

oooo
ooooooooo
ooooo
oooo
oooo

ReVerb Confidence Function

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

```

ooo
oooooooooooo●ooo
oooooooooooo

```

```

oooo
oooooooooooo
oooooo
oooo
oooo
oooo

```

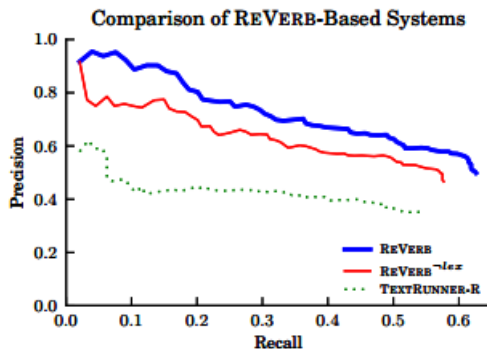
ReVerb Confidence Function

| Weight | Feature |
|--------|---|
| 1.16 | (x, r, y) covers all words in s |
| 0.50 | The last preposition in r is <i>for</i> |
| 0.49 | The last preposition in r is <i>on</i> |
| 0.46 | The last preposition in r is <i>of</i> |
| 0.43 | $len(s) \leq 10$ words |
| 0.43 | There is a WH-word to the left of r |
| 0.42 | r matches VW*P from Figure 1 |
| 0.39 | The last preposition in r is <i>to</i> |
| 0.25 | The last preposition in r is <i>in</i> |
| 0.23 | $10 \text{ words} < len(s) \leq 20 \text{ words}$ |
| 0.21 | s begins with x |
| 0.16 | y is a proper noun |
| 0.01 | x is a proper noun |
| -0.30 | There is an NP to the left of x in s |
| -0.43 | $20 \text{ words} < len(s)$ |
| -0.61 | r matches V from Figure 1 |
| -0.65 | There is a preposition to the left of x in s |
| -0.81 | There is an NP to the right of u in s |

○○○
 ○○○○○○○○○●○○
 ○○○○○○○○

○○○
 ○○○○○○○○
 ○○○○
 ○○○
 ○○○

Evaluation



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

```

○○○
○○○○○○○○○○●○
○○○○○○○○○

```

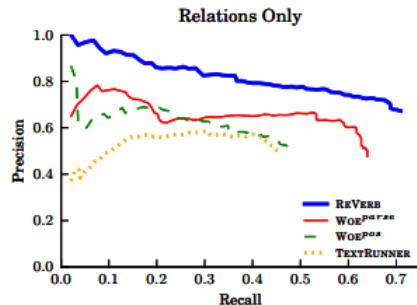
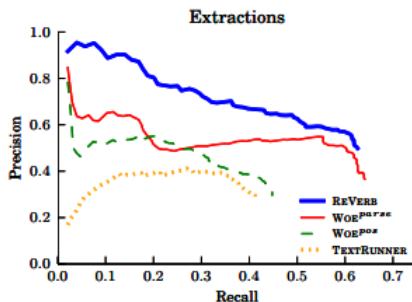
```

○○○
○○○○○○○
○○○○○
○○○
○○○
○○○

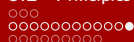
```

Methods

Why?



Argument extraction is open to improvements



Why?

Evaluating the evaluation:

REVERB - Incorrect Extractions

| | |
|-----|--|
| 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 |

REVERB - Missed Extractions

| | |
|-----|---|
| 52% | Could not identify correct arguments |
| 23% | Relation filtered out by lexical constraint |
| 17% | Identified a more specific relation |
| 8% | POS/chunking error |

○○○
○○○○○○○○○○○○○
●○○○○○○○○○

○○○○
○○○○○○○○○
○○○○○
○○○○
○○○○

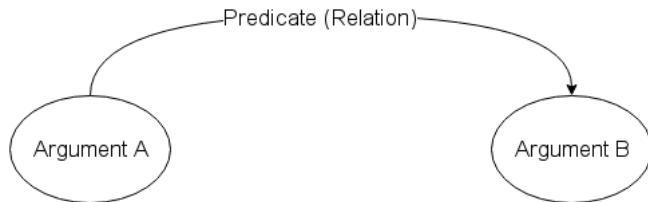
OIE - Principles

Data Representation

ooo
oooooooooooooooo
o●oooooooo

oooo
oooooooooooo
ooooo
oooo
oooo
oooo

Standard Patterns



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

ooo
oooooooooooo
oo●oooooo

oooo
oooooooooo
oooooo
oooo
oooo
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
```

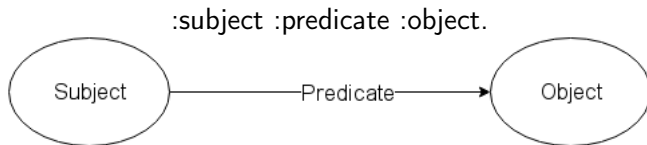
```
oooo
ooooo
oooooooo
ooooo
oooo
oooo
```

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
oooooooooooo
ooooo
oooo
oooo
oooo

RDF Reification

Motivation: How can I realize embedded propositions?

Example: Peter said, he watched the movie.

ooo
oooooooooooooooo
oooo●ooo

oooo
oooooooooo
ooooo
oooo
oooo
oooo

RDF Reification

Motivation: How can I realize embedded propositions?

Example: Peter said, he watched the movie.

Wrong proposition

:Peter :watched :movie

ooo
oooooooooooooooo
ooooo●ooo

oooo
oooooooooo
ooooo
oooo
oooo
oooo

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.
```



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
oooo
oooo

RDF Syntax

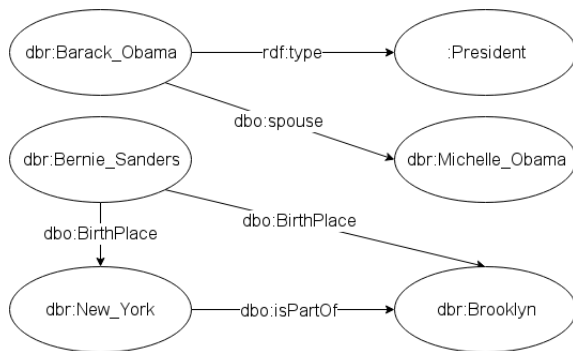
```
dbr:Barack_Obama a foaf:person, :President;  
    dbo:spouse dbr:Michelle_Obama.  
dbr:Bernie_Sanders dbo:birthPlace dbr:New_York,  
    dbr:Brooklyn;  
dbr:Brooklyn dbo:isPartOf dbr:New_York
```


ooo
ooooooooooooo
ooooooooo●

oooo
ooooooooo
ooooo
oooo
oooo

Data Representation

... as Graph



ooo
ooooooooooooo
oooooooooooo

oooo
oooooooooooo
oooooo
oooo
oooo
oooo

Example: LODifier

ooo
ooooooooooooo
oooooooooooo

oooo
oooooooooooo
oooooo
oooo
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

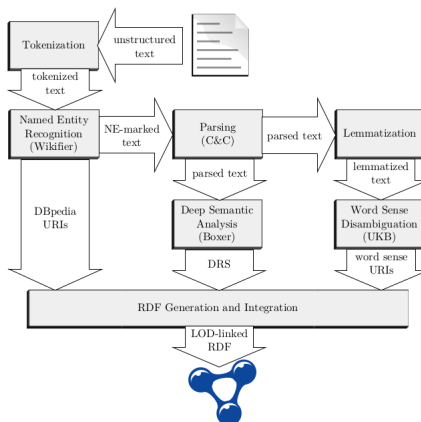
ooo
ooooooooooooo
ooooooooooooo

●ooo
ooooooooooooo
oooooo
oooo
oooo

Example: LODifier Architecture

Architecture

Architecture



ooo
oooooooooooooooo
oooooooooooo

oo●o
oooooooooo
oooooo
oooo
oooo

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

ooo
ooooooooooooo
oooooooooooo

ooo●
oooooooooooo
oooooo
oooo
oooo

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
oooo
oooo

Example: LODifier

Preprocessing

ooo
ooooooooooooo
oooooooooooo

oooo
o●oooooo
ooooo
oooo
oooo

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].

ooo
oooooooooooooooo
oooooooooooo

oooo
ooo●ooooo
ooooo
oooo
oooo
oooo

Parsing Syntax - C&C

C&C Parser

Syntactical Parser that tags POS and builds Parse Trees (CCG).

```

ooo
ooooo
oooooooooooo
ooooooooo

```

```

oooo
ooo●oooo
ooooo
ooooo
oooo
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
oooo
oooo
oooo

Find Relations - Boxer

Boxer

Creates DRSs from C&C Output

ooo
oooooooooooooooo
oooooooooooo

oooo
oooo●ooo
ooooo
oooo
oooo
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
oooooooooooo
oooooooooo

oooo
oooo●ooo
ooooo
oooo
oooo
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):

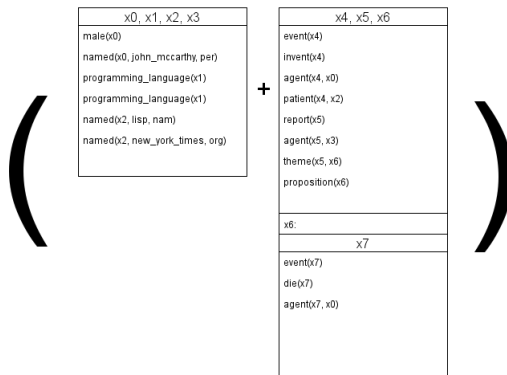
- **Unary Relations (Classes):** eg. *topic*, *person*, *event*, *male*, ...
+ all verbs
- **Binary Relations:** agent, patient, ... (semantic roles)

○○○
 ○○○○○○○○○○○○
 ○○○○○○○○

○○○○
 ○○○○○●○○
 ○○○○
 ○○○
 ○○○

Preprocessing

Boxer Output



ooo
oooooooooooooooo
oooooooooooo

oooo
oooooooo●o
ooooo
oooo
oooo
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
oooo
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
oooo
oooo

Example: LODifier

RDF Construction

ooo
oooooooooooooooo
oooooooooooo

oooo
oooooooooooo
o●oooo
oooo
oooo

RDF Construction

What now?

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

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oo●ooo
oooo
oooo

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
oooooooooooo

oooo
oooooooooooo
ooo●ooo
oooo
oooo

Namespaces/Vocabularies

And uses standard namespaces:

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

ooo
oooooooooooooooo
oooooooooooo

oooo
oooooooooooo
oo●ooo
oooo
oooo

Namespaces/Vocabularies

Finally the two ontologies:

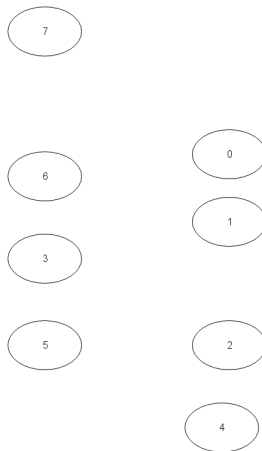
- **wn30**: contains all WordNet URIs
- **dbpedia**: contains the dbpedia URIs
- **class**: contains classes not in wn30 nor in dbpedia



RDF Construction Strategy I

Create a blanknode `_:x` for each discourse referent (x_0, x_1, \dots)

RDF Construction Strategy II



RDF Construction Strategy III

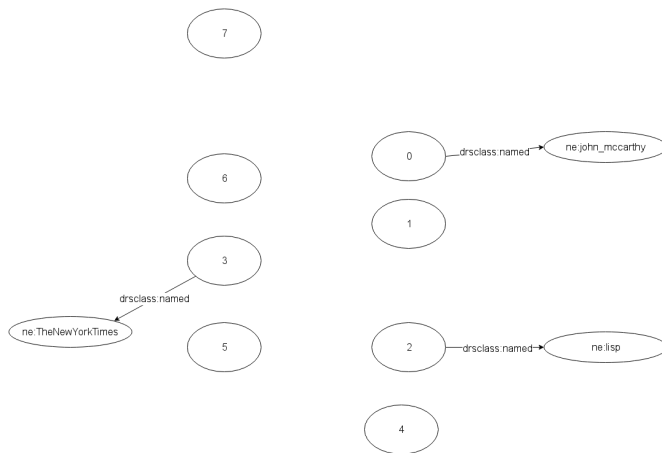
if NE, then create

`_:x drsclass:named ne:URI`



RDF Construction

RDF Construction Strategy IV

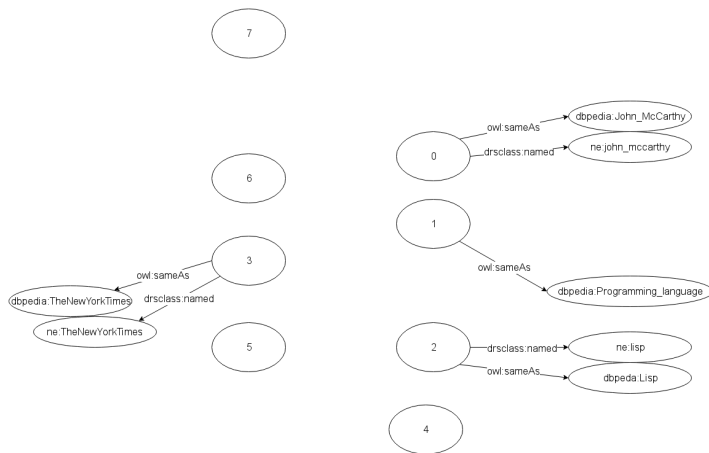


RDF Construction Strategy V

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

RDF Construction

RDF Construction Strategy VI



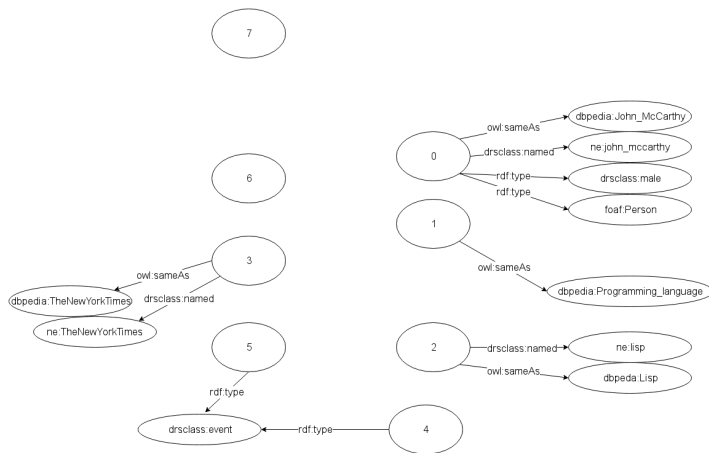


RDF Construction Strategy VII

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

RDF Construction

RDF Construction Strategy VIII





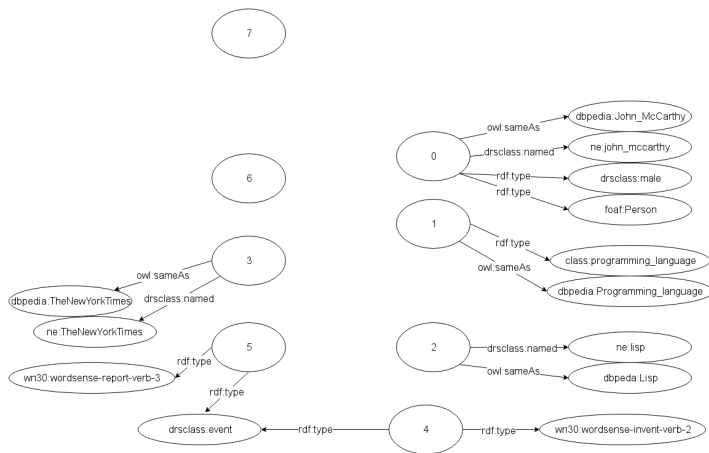
RDF Construction Strategy IX

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



RDF Construction

RDF Construction Strategy X





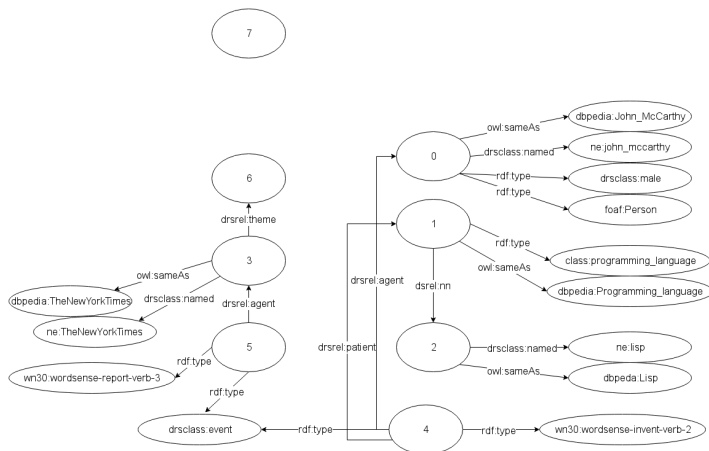
RDF Construction Strategy XI

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●
oooo
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
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
o●ooo
oooo

Method

- TDT-2 benchmark dataset: 84.000 news documents

```

○○○
○○○○○○○○○○○○
○○○○○○○○○○
○○○○○○○○

```

```

○○○○
○○○○○○○○
○○○○○○
○○○○○
○○●○○
○○○○

```

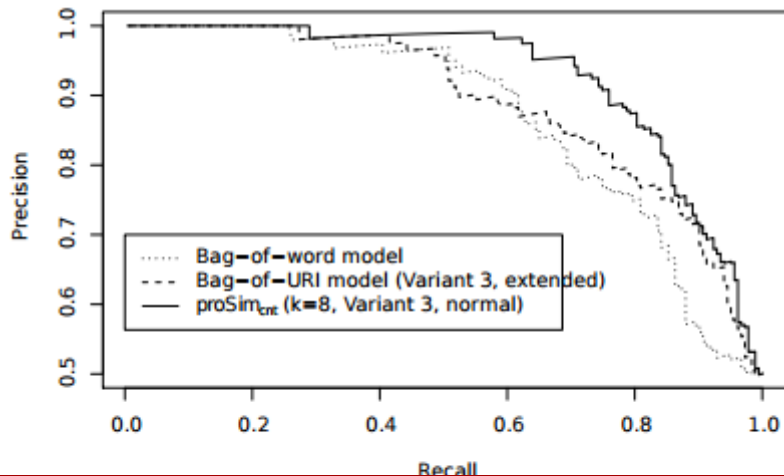
Experiments

Accuracy

Table 1. Accuracy on Story Link Detection Task

| Model | normal extended | |
|--|-----------------|------|
| Similarity measures without structural knowledge | | |
| 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) | 73.4 | 76.4 |
| Similarity measures with structural knowledge | | |
| proSim _{cut} (k=6, Variant 1) | 79.0 | 78.9 |
| proSim _{cut} (k=6, Variant 2) | 80.3 | 80.3 |
| proSim _{cut} (k=6, Variant 3) | 81.6 | 81.6 |
| proSim _{cut} (k=8, Variant 1) | 77.7 | 77.6 |
| proSim _{cut} (k=8, Variant 2) | 79.2 | 79.0 |
| proSim _{cut} (k=8, Variant 3) | 82.1 | 81.9 |
| proSim _{len} (k=6, Variant 3) | 81.5 | 81.4 |
| proSim _{len} (k=8, Variant 3) | 80.3 | 80.1 |
| proSim _{len} (k=10, Variant 3) | 80.0 | 79.8 |
| proSim _{sqen} (k=6, Variant 3) | 80.4 | 80.4 |
| proSim _{sqen} (k=8, Variant 3) | 81.1 | 80.9 |
| proSim _{sqen} (k=10, Variant 3) | 80.5 | 80.4 |

Precision - Recall



ooo
ooooooooooooo
oooooooooooo

oooo
oooooooooooo
ooooo
oooo
oooo
●ooo

Example: LODifier

Conclusions

ooo
oooooooooooooooo
oooooooooooo

oooo
oooooooooooo
oooooo
oooo
oooo
o●oo

Conclusions

What to draw from this?

TODO

ooo
ooooooooooooo
ooooooooooooo

oooo
ooooooooooooo
oooooo
oooo
oooo
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
- domain-independent

ooo
oooooooooooooooo
oooooooooooo

oooo
oooooooooooo
ooooo
oooo
oooo
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
- Some points are unclear / not even described

ooo
ooooooooooooo
oooooooooooo

oooo
oooooooooooo
oooooo
oooo
oooo
oooo

Conclusion

ooo
ooooooooooooo
oooooooooooo

oooo
oooooooooooo
oooooo
oooo
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
- restricted usability of results due to low precision

Future Opportunities

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