Open Information Extraction

Dominik Both, Tonio Weidler

Proseminar *Text Mining*
Andrea Zielinski

Institut für Computerlinguistik, Universität Heidelberg, 15.07.2016

Strukturierung

Introduction to Information Extraction

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)

Problems of Information Extraction

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

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:
- V | V P | VW* P
- Always choses longest possible match
- Merge ajacent matches together

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

$$V = \text{verb particle? adv?}$$

$$W = (\text{noun} \mid \text{adj} \mid \text{adv} \mid \text{pron} \mid \text{det})$$

$$P = (\text{prep} \mid \text{particle} \mid \text{inf. marker})$$

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

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 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
	Intro. Phrases: Discovered by Y, X ...
	Relative Clauses: ...the Y that X discovered
3%	Do Not Match 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.

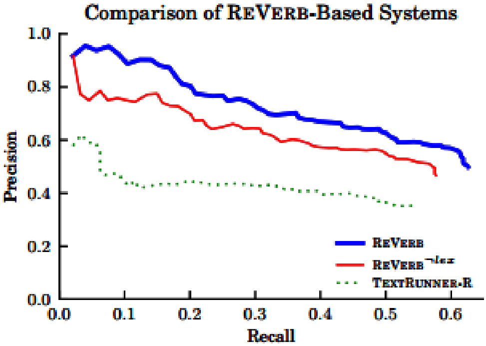
ReVerb Confidence Function

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

ReVerb Confidence Function

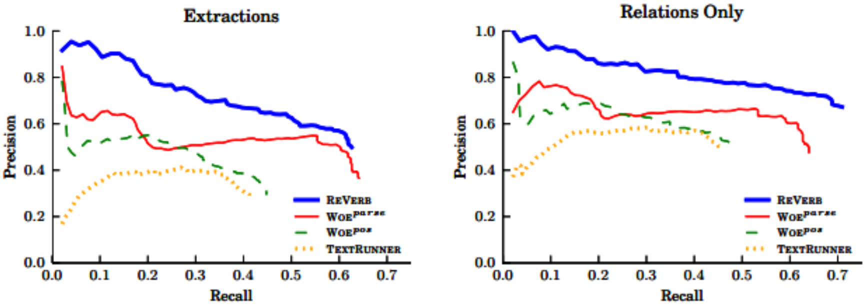
Weight	Feature
1.16	(x, r, y) covers all words in s
0.50	The last preposition in r is *for*
0.49	The last preposition in r is *on*
0.46	The last preposition in r is *of*
0.43	$len(s) \leq 10$ words
0.43	There is a WH-word to the left of r
0.42	r matches VW*P from Figure 1
0.39	The last preposition in r is *to*
0.25	The last preposition in r is *in*
0.23	10 words $< len(s) \leq 20$ 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 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 y in s

Evaluation



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

Why?



Argument extraction is open to improvements

Why?

Evaluating the evaluation:

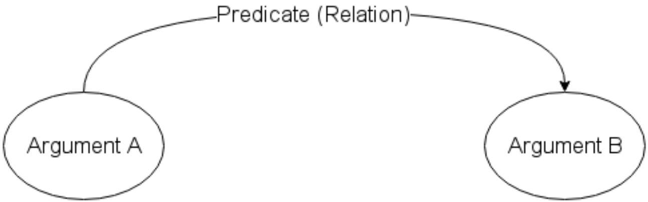
ReVerb - Incorrect Extractions	
65%	Correct relation phrase, incorrect arguments
16%	N-ary relation
8%	Non-contiguous relation phrase
2%	Imperative verb
2%	Overspecified relation phrase
7%	Other, including POS/chunking errors

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

Standard Patterns



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

Unnormalized Annotation

(argument_a, predicate_x, argument_b)
(argument_a, predicate_y, argument_c)
(argument_a, predicate_y, argument_d)

Problems

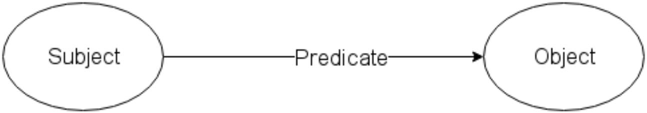
- redundant
- unnormalized
- can only produce binary predicates

RDF and Linked Data

Resource Description Framework

Models propositions by constructing *triples* including **Subjects**, **Objects** and **Predicates**
Generates a directed graph

:subject :predicate :object.



RDF Concepts and Notation

- **URIs**
  identifies ressources (S, R, O) distinctivly 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)

RDF Reification

Motivation: How can I realize embedded propositions?

Example: Peter said, he watched the movie.

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.
```

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

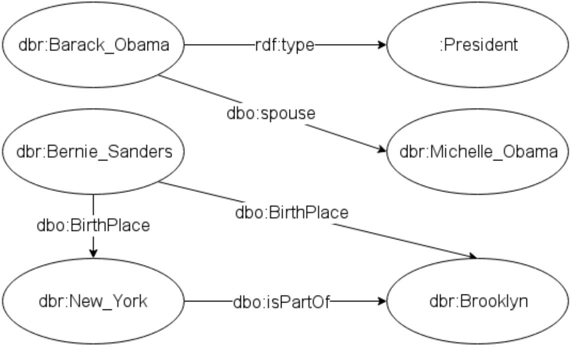
Ontologies are huge RDF Graphs containing many triples, eg.:

- DBpedia
- Wikidata
- WordNet

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
```

... as Graph


dbr:Barack_Obama
rdf:type
:President
dbo:spouse
dbr:Bernie_Sanders
dbr:Michelle_Obama
dbo:BirthPlace
dbo:BirthPlace
dbr:New_York
dbo:isPartOf
dbr:Brooklyn

Example: LODifier

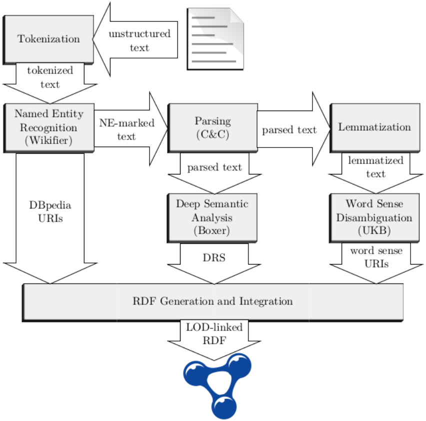
LODifier: Generating Linked Data from Unstructured Text (Augenstein et al., 2012

Generate an RDF Graph from unstructured Text

Past Approaches: Use Patterns to trade recall for precision
LODifier: Process the entire text

Example: LODifier
Architecture

Architecture



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

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

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 (CCG).

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'))).
```

Find Relations - Boxer

Boxer

Creates DRSs from C&C Output

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

Find Relations - Boxer

Boxer

Creates DRSs from C&C Output

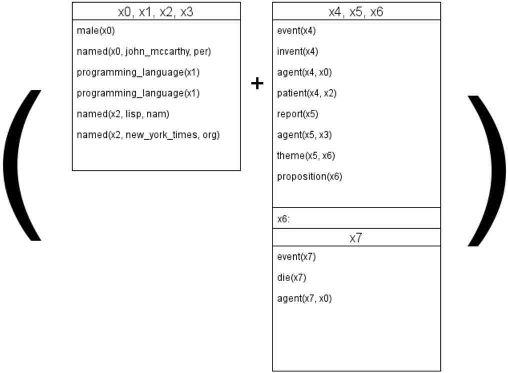
Discours Representation Structure (DRS)

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)

Boxer Output



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

Preprocessing Result

We now have ...

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

Example: LODifier

RDF Construction

What now?

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

Namespaces/Vocabularies

LODifier introduces several namepaces:

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

Namespaces/Vocabularies

And uses standard namespaces:

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

Namespaces/Vocabularies

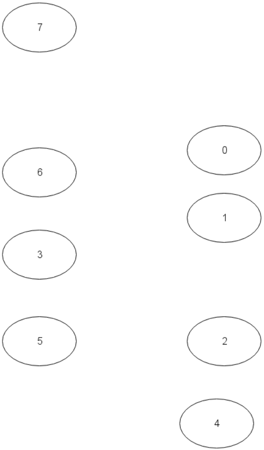
Finally the two ontologies:

- **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, ...)

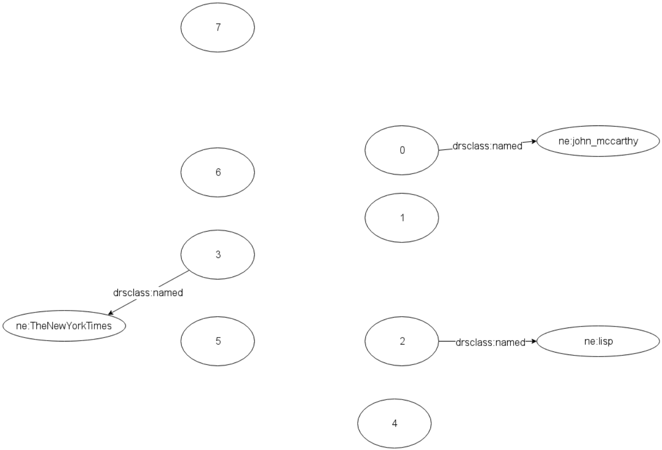
RDF Construction Strategy II



RDF Construction Strategy III

if NE, then create
_:x drsclass:named ne:URI

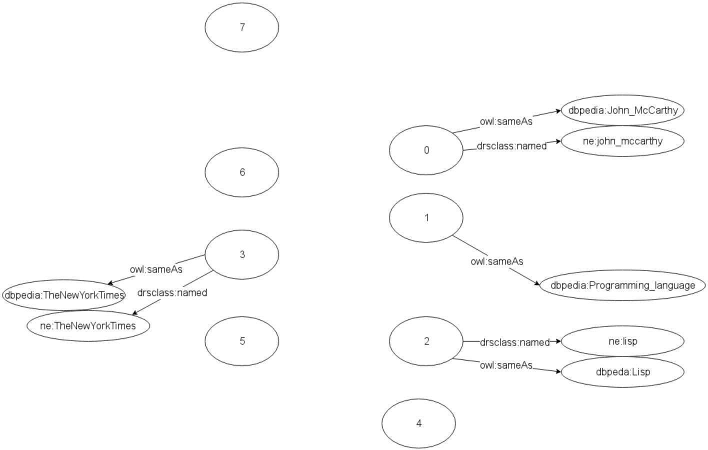
RDF Construction Strategy IV



RDF Construction Strategy V

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

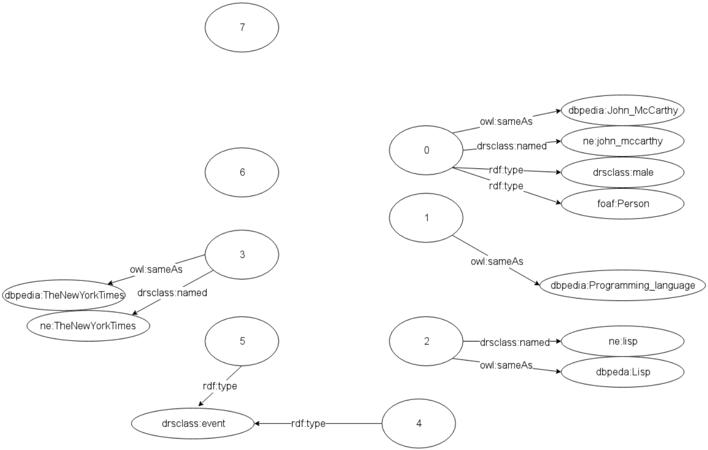
RDF Construction Strategy VI



RDF Construction Strategy VII

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

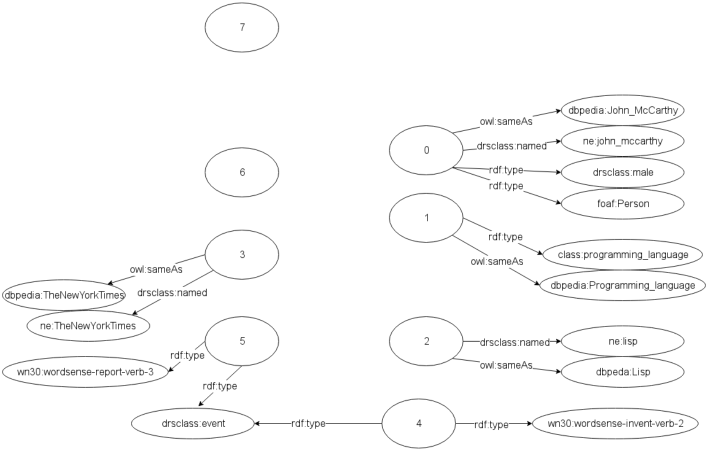
RDF Construction Strategy VIII



RDF Construction Strategy IX

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

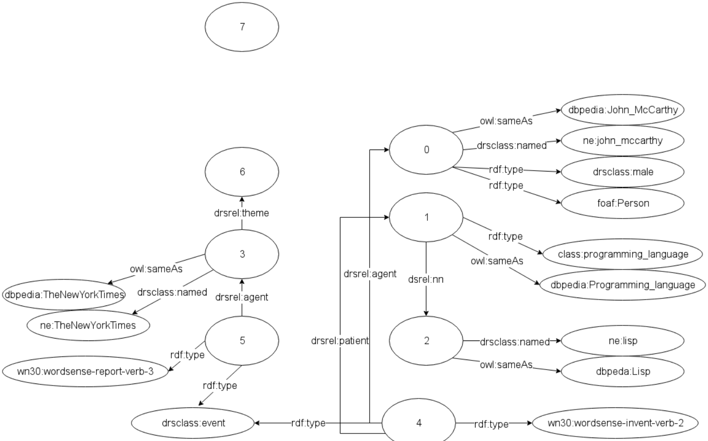
RDF Construction Strategy X



RDF Construction Strategy XI

create triples from binary relations (agent, theme, ...)
_:x drsrel:RELATION _:y

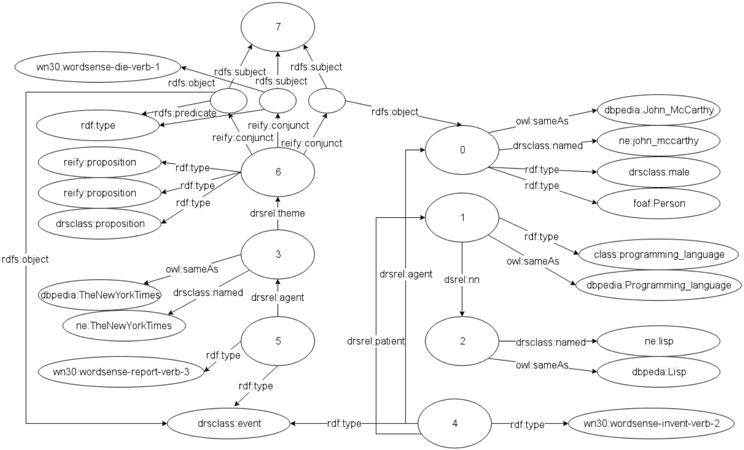
RDF Construction Strategy XII



RDF Construction Strategy XIII

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

RDF Construction Strategy XIV



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 . ]
```

Example: LODifier
Experiments

Method

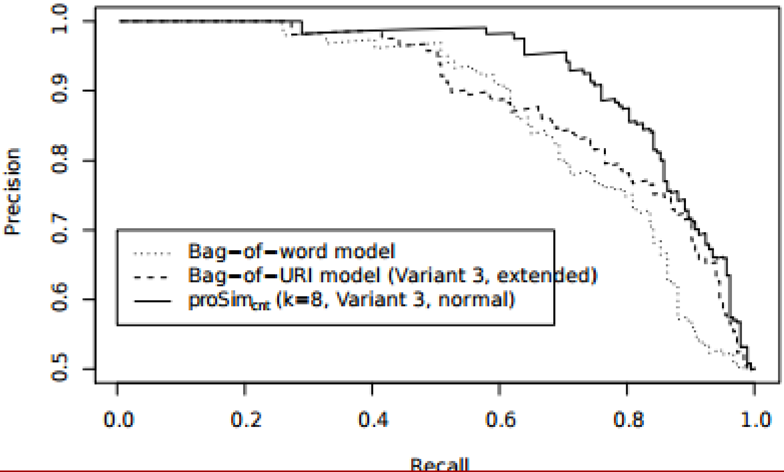
- TDT-2 benchmark dataset: 84.000 news documents

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		
$\text{proSim}_{\text{cnt}}$ (k=6, Variant 1)	79.0	78.9
$\text{proSim}_{\text{cnt}}$ (k=6, Variant 2)	80.3	80.3
$\text{proSim}_{\text{cnt}}$ (k=6, Variant 3)	81.6	81.6
$\text{proSim}_{\text{cnt}}$ (k=8, Variant 1)	77.7	77.6
$\text{proSim}_{\text{cnt}}$ (k=8, Variant 2)	79.2	79.0
$\text{proSim}_{\text{cnt}}$ (k=8, Variant 3)	**82.1**	**81.9**
$\text{proSim}_{\text{len}}$ (k=6, Variant 3)	81.5	81.4
$\text{proSim}_{\text{len}}$ (k=8, Variant 3)	80.3	80.1
$\text{proSim}_{\text{len}}$ (k=10, Variant 3)	80.0	79.8
$\text{proSim}_{\text{sqlen}}$ (k=6, Variant 3)	80.4	80.4
$\text{proSim}_{\text{sqlen}}$ (k=8, Variant 3)	81.1	80.9
$\text{proSim}_{\text{sqlen}}$ (k=10, Variant 3)	80.5	80.4

Precision - Recall



Example: LODifier

Conclusions

What to draw from this?

What we liked

- full-text OIE
- relations arent overspecified
- TO BE EXTENDED

What we didnt like

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

- Paper scraches only the surface of the system
- Some points are unclear / not even described

OIE Systems in Context

OIE Systems in Context

Comparison

OIE Systems in Context

Evaluating the Approaches

Conclusion

Conclusion

Problems and Obstacles

Conclusion

Future Opportunities