SEMAPRO 2020

Pynsett: A Programmable Relation Extractor

Alberto Cetoli

alberto.cetoli@uk.qbe.com

QBE Europe

About Me

- Former Condensed Matter Physicist
- 7+ years of industry experience on NLP
  - Mainly Information Extraction from documents
  - Interested in IE and Knowledge Representation
- Data scientist at QBE Europe

Introduction

Pynsett is a rule based relation extraction

Work in progress

- Why rule-based?
- Semantic representation
  - Text as a discourse
- Discourse/Sentence Matching
- Examples and results

Why rule based?

- Rules are precise
- Easy to explain
- Quick to deploy

Why rule based?

Primary goal: *easy rules*

```
MATCH "PERSON works as a ROLE."
CREATE (works_as PERSON ROLE)
```

Semantic representation

Jane is working at ACME Inc as a woodworker. She is quite taller than the average

```
Jane(r1), work(e1), ACME_Inc(r2), woodworker(r3),
AGENT(e1, r1), at(e1, r2), as(e1, r3),

Jane(r4), be(e2), tall(r5), average(r6), quite(r7),
AGENT(e2, r4), ADJECTIVE(e2, r5), than(r5, r6), ADVERB(r5, r7),

REFERS_TO(r1, r4), REFERS_TO(r4, r1)
```

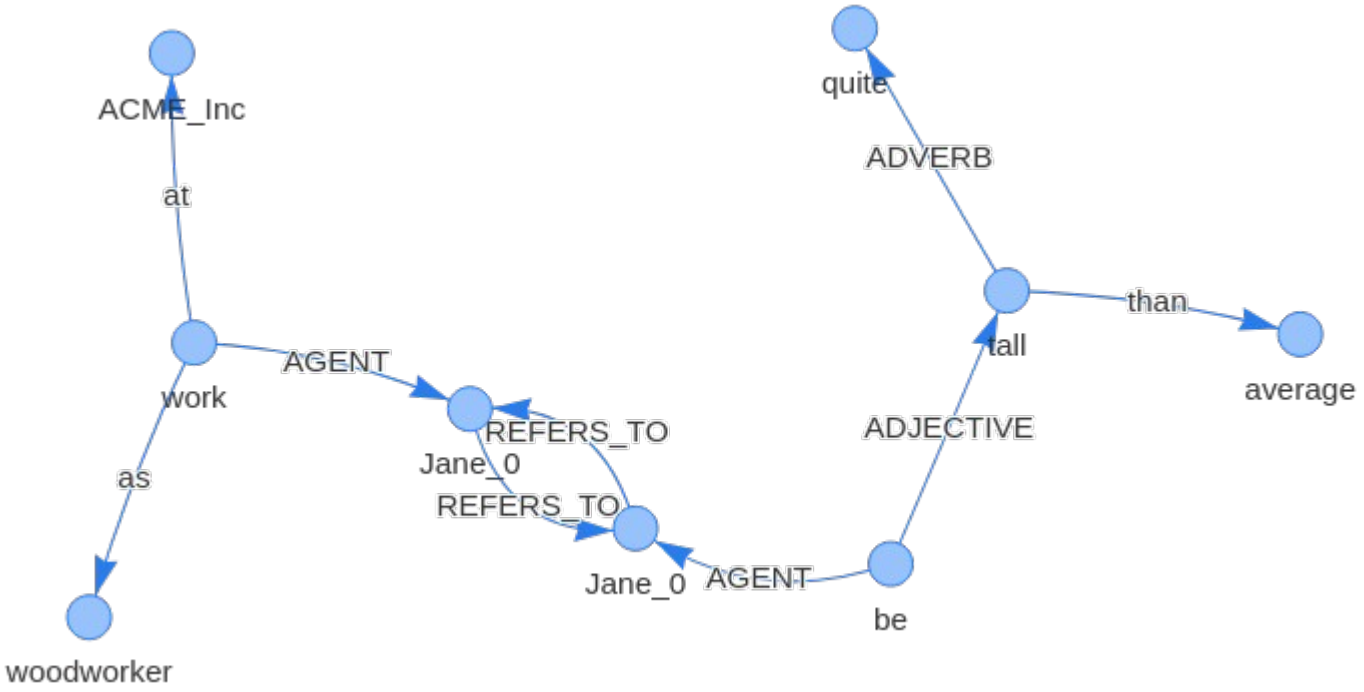
Semantic representation

Jane is working at ACME Inc as a woodworker. She is quite taller than the average

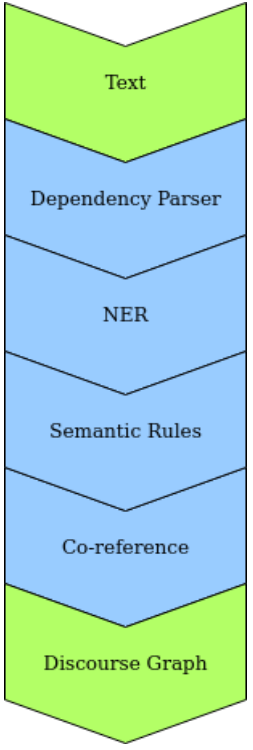
```
Jane(r1), work(e1), ACME_Inc(r2), woodworker(r3),
AGENT(e1, r1), at(e1, r2), as(e1, r3),

Jane(r4), be(e2), tall(r5), average(r6), quite(r7),
AGENT(e2, r4), ADJECTIVE(e2, r5), than(r5, r6), ADVERB(r5, r7),

REFERS_TO(r1, r4), REFERS_TO(r4, r1)
```

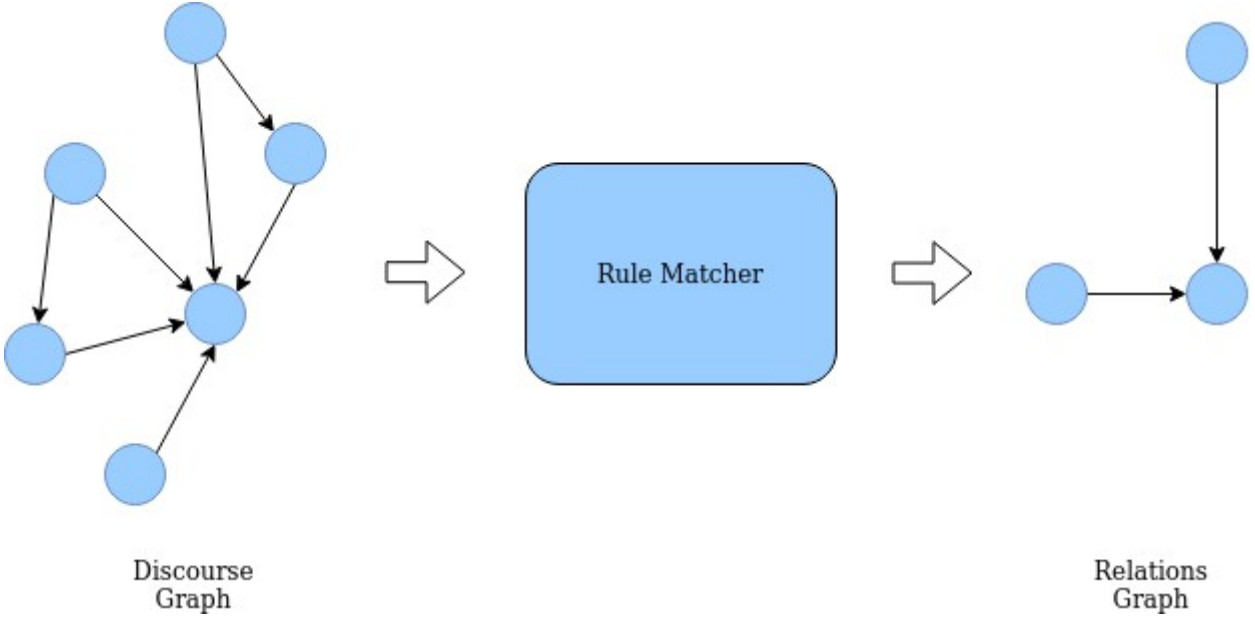
Semantic representation

The Text becomes a graph

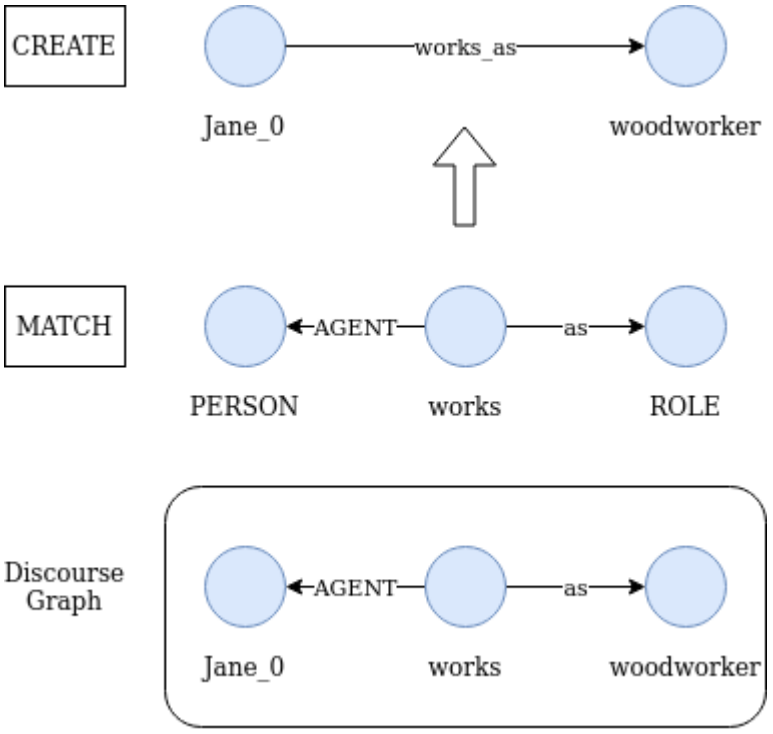


Semantic representation





Discourse Matching



A text becomes a *Discourse graph*

The text in the rules become a *Matching Graph*

A rule is triggered if

- Rule graph is *sub-isomorphic* to discourse
- Each node matches

The result is an edge in the *Relations Graph*

Discourse Matching

Nodes are represented as a list of features

```
{
    vector: EMBEDDING[woodworker]
    lemma: "woodworker",
    negated: False,
    entity_type: None,
    node_type: noun
}
```

→ Glove vector of “woodworker”

Discourse Matching

“Woodworker” node

```
vector: EMBEDDING[woodworker],
negated: False,
entity_type: None,
node_type: noun
```

“Carpenter” node

```
vector: EMBEDDING[carpenter],
negated: False,
entity_type: None,
node_type: noun
```

Discourse Matching

$$\text{EMBEDDING[woodworker]} \cdot \text{EMBEDDING[carpenter]} > \text{threshold}$$

The threshold is hardcoded but might be learned in future versions

Discourse Matching

“Jane” node

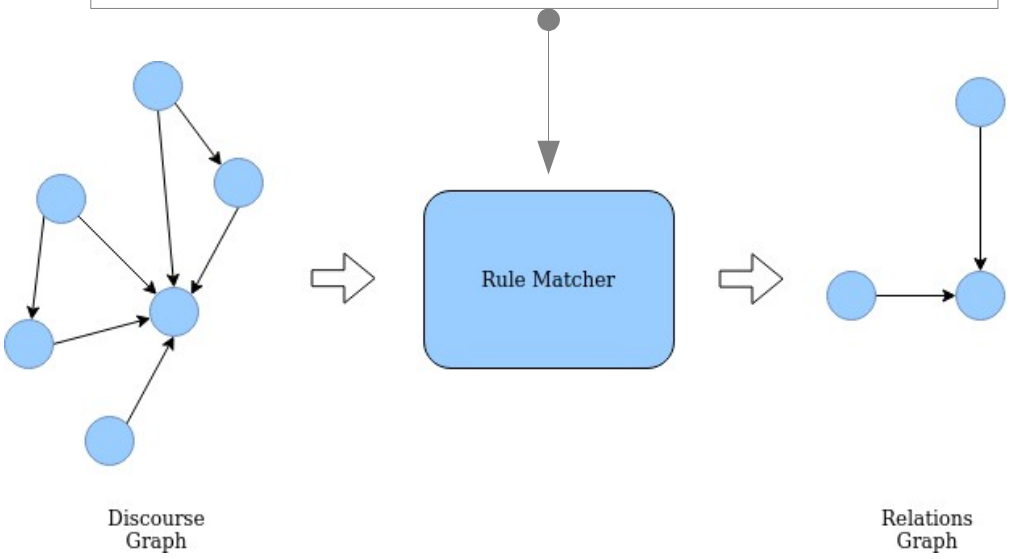
```
vector: EMBEDDING[Jane],
negated: False,
entity_type: PERSON,
node_type: noun
```

PERSON node

```
vector: *,
negated: False,
entity_type: PERSON,
node_type: noun
```

Semantic representation

```
DEFINE ROLE AS [carpenter, painter];

MATCH "PERSON works as a ROLE."
CREATE (works_as PERSON ROLE);
```



Conclusions and future work

- Presented an open-source rules-based relation extractor
- Rules are easy to write and explain
- Glove vectors are old fashioned
  - TODO: Use modern embeddings
- The native named entities are from Ontonotes 5.0
  - TODO: allow for custom NER
- Only one-level matching
  - TODO: build a proper resolution tree for matching

Conclusions and future work

Thank you!

Preliminary evaluation

```
DEFINE PERSON AS {PERSON};
DEFINE DAY AS {DATE};
DEFINE YEAR AS {DATE};
DEFINE AT_TIME AS {DATE};
DEFINE AT_MOMENT AS [Monday, Tuesday
     , Wednesday, Thursday, Friday, Saturday, Sunday];

MATCH "PERSON#1 is born AT_TIME#2"
CREATE (DATE_OF_BIRTH 1 2);

MATCH "PERSON#1 is born on DAY#2"
CREATE (DATE_OF_BIRTH 1 2);

MATCH "PERSON#1 is born in YEAR#2"
CREATE (DATE_OF_BIRTH 1 2);

MATCH "PERSON#1 dies AT_MOMENT#2"
CREATE (DATE_OF_DEATH 1 2);

MATCH "PERSON#1 dies on DAY#2"
CREATE (DATE_OF_DEATH 1 2);

MATCH "PERSON#1 dies in YEAR#2"
CREATE (DATE_OF_DEATH 1 2);
```

Test on 2 relations on TACRED:
- Date of birth
- Date of death

Precision 100%

DOB Recall 33%
DOD Recall 3.6%

TACRED SOTA F1=71.2%

Combining rules

WORKS_AT ∧ TALL

```
MATCH "PERSON works at ORG as a ROLE. PERSON is tall."
CREATE (tall_worker_at PERSON ORG)
```

Combining rules

WORKS_AT ∧ TALL

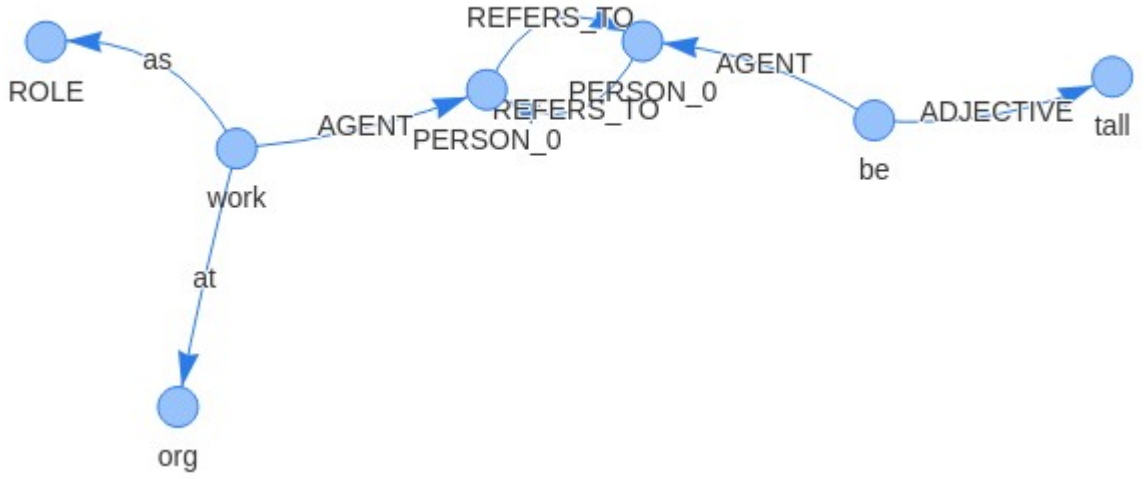
```
MATCH "PERSON works at ORG as a ROLE. PERSON is tall."
CREATE (tall_worker_at PERSON ORG)
```

PERSON co-refer

Combining rules

WORKS_AT ∧ TALL

PERSON works at ORG as a ROLE. **PERSON** is tall.



Ad-hoc entities

```
DEFINE TEAM AS [team, group, club];
DEFINE UNIVERSITY AS [university, academy, polytechnic];
DEFINE LITERATURE AS [book, story, article, series];
```

Types of edges

AGENT, PATIENT: the subject and object of the sentence are converted to agent and patient edges coherently with the verb's voice. In addition, these relations are propagated to relevant subordinates.

ADJECTIVE, ADVERB: adjectives and adverbs are connected to the relevant node through an edge. The only exceptions are negation adverbs, which become part of the node's attributes to facilitate the matching procedure, as explained in Section II-E.

OWNS: possessive pronouns are translated into a relation induced by the pronoun's semantics.

PREPOSITIONS: all the prepositions become edges (Figure 1). Ideally - in a future work - a further semantic layer should be added to classify the preposition's meaning in context.

SUBORDINATES: the subordinate clauses are linked to the main one through the SUBORDINATE edge. One additional type is the ADVOCATIVE_CLAUSE, marking a conditional relation among sentences. This is a placeholder for future versions of the system where ideally rules can be extracted from the text.

Conjunctions

In order to facilitate graph matching, the conjunction list is flattened and linked to the head node whenever possible. For example, the sentence *Jane is smart and wise* becomes, in predicate form

```
Jane(r1), be(e1), smart(r2), wise(r3),
AGENT(e1, r1), ADJECTIVE(e1, r2), ADJECTIVE(e1, r3)
```

Effectively, 'AND' and 'OR' disappear from the graph. This is a crude approximation that facilitates the relation extraction at the expense of semantic correctness.