Identifying Relations for open Information Extraction

Anthony Fader
Stephen Soderland
Oren Etzioni

Presented by:
Sindhura Tokala
Refresher

- Sentence: "Hampstead is a suburb of London"
- (arg1, relation, arg2) = (Hampstead, is a suburb of, London)
OPEN Information Extraction

- Does not require pre-specified vocabulary
Why do we need open IE?

- Traditional closed IE systems: learn an extractor for each target relation from labeled training examples.

- Drawbacks:
  - does not scale
  - cannot be used where target relations cannot be used in advance
How does Open IE address these drawbacks?

- automatically identifies "relation phrases"
- this enables the extraction of arbitrary relations
- no need pre-specify vocabulary!
Contributions

- Identified problems with existing Open IE systems.
- Established constraints on relation phrase extractions
- ReVerb Open IE system.
Existing Open IE systems:

- TextRunner
- WOE
Problems with existing Open IE systems:

- Incoherent extractions
- Uninformative extractions
Incoherent extractions:

- Relation phrase has no meaningful interpretation

| Sentence | Incoherent Relation |
|----------|---------------------|
| The guide *contains* dead links and *omits* sites. | contains omits |
| The Mark 14 *was central* to the *torpedo* scandal of the fleet. | was central torpedo |
| They *recalled* that Nungesser *began* his career as a precinct leader. | recalled began |
Uninformative Extractions

- Omit critical information.

| is    | is an album by, is the author of, is a city in |
|-------|-----------------------------------------------|
| has   | has a population of, has a Ph.D. in, has a cameo in |
| made  | made a deal with, made a promise to |
| took  | took place in, took control over, took advantage of |
| gave  | gave birth to, gave a talk at, gave new meaning to |
| got   | got tickets to, got a deal on, got funding from |
Traditional Open IE systems:

- Three step method to extract binary relations of the form (arg1, relation phrase, arg2).
- Label: Automatically label using heuristics or distant supervision
- Learn: Extractor is learned using a graphical model
- Extract:
  1. extracts a pair of arguments
  2. labels each word between the arguments as part of the relation or not.
Problems with this approach:

- Needs large amount of training data.
- Hueristic labelling leads to noisy data.
- Extraction step is sequential.
- Extractor cannot backtrack.
Example

- Input sentence: "Faust made a deal with the devil"
- Possibility 1:
  - (Argument 1, Relation, Argument 2) = (Faust, made, a deal)
- Possibility 2: [More desirable]
  - (Argument 1, Relation, Argument 2) = (Faust, made a deal with, the devil)
Paper Contributions:

- Impose constraints on Relation Phrases to avoid Incoherent and Uniformative extractions.
- Syntactic Constraint
- Lexical constraint
Syntactic Constraint:

- Every multi-word relation phrase must:
  - Begin with a verb
  - End with a preposition
  - Be a contiguous sequence of words
  - Occur between its two arguments
Syntactic Constraint:

\[
V \mid VP \mid VW^*P
\]

\[
V = \text{verb particle? adv?}
\]

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

\[
P = (\text{prep} \mid \text{particle} \mid \text{inf. marker})
\]
Lexical constraint

"The Obama administration is offering only modest greenhouse gas reduction targets at the conference"

Syntactic Constraint will match:
"is offering only modest greenhouse gas reduction targets at"
Lexical Constraint:

- Avoids overspecification.
- Imposes a minimal number of distinct argument pairs.
Limitations:

- How much recall is lost due to the constraints?

| 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

- phrases are identified holistically
- filtered based on statistics
- "Relation first", instead of "Argument First"
Extraction Algorithm:

- Step 1: Identifies Relation Phrases
- Step 2: Finds pair of NP arguments for each relation
- Step 3: Assign confidence score using logistic regression
Extraction Algorithm:

- Input: POS-tagged and NP-chunked sentence
- Output: set of (x, y, z) tuples
Relation Extraction

For each verb $v$ in $s$, find the longest sequence of words $r_v$ such that:

- $r_v$ starts at $v$
- $r_v$ satisfies the syntactic constraint
- $r_v$ satisfies the lexical constraint
- If any pair of matches are adjacent or overlap in $s$, merge them into a single match.
Argument Extraction

- For each relation phrase \( r \) identified, find the nearest noun phrase \( x \) to the left of \( r \) in \( s \) such that \( x \) is
- not a relative pronoun, WHO-adverb, or existential “there”
- Find the nearest noun phrase \( y \) to the right of \( r \) in \( s \).
Example

- Input: "Hudson was born in Hampstead, which is a suburb of London"
- Step 1:
  - "was", "born in", "is a suburb of" are identified
  - "was" and "born in" are merged
- Step 2:
  - (Hudson, Hampstead) and (Hampstead, London) are selected respectively
- Output:
  - e1: (Hudson, was born in, Hampstead)
  - e2: (Hampstead, is a suburb of, London)
Confidence Function

- Extraction algorithm has high recall, low precision.
- Trade recall for precision
- Logistic regression classifier to assign confidence score to each extraction
Experiments:

ReVerb versus:

- ReVerb \ lex: ReVerb without the lexical constraint

TextRunner:
- Uses a second order linear-chain CRF
- Trained on the Penn Treebank
- Same POS tagger and NP-chunker as ReVerb
TextRunner – R:
  - Similar to TextRunner
  - Trained on ReVerb extractions

WOE-pos: TextRunner with relations learned from Wikipedia

WOE-parse:
  - uses a dictionary of dependency path patterns
  - extracted from Wikipedia
Experiments
Experiments

Comparison of REVERB-Based Systems

Precision

Recall

\- ReVERB
\- ReVERB-\text{lex}
\- TextRunner-R
Experiments
Experiments
ReVerb Error analysis

**ReVerb - Incorrect Extractions**

| Percentage | Type                                      |
|------------|-------------------------------------------|
| 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**

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

- ReVerb's performs better at all frequency thresholds.
- ReVerb's frequency 1 extractions :: TextRunner frequency 10 extractions.
- ReVerb returns with greater precision even when redundancy is taken into consideration.
Future work

- Uses Syntactic and Lexical constraints to improve learned CRF models
- Improved methods for argument extraction