Open Domain Information Extraction

Günter Neumann, DFKI, 2012
Improving TextRunner

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

- Novel form of **self-supervised learning** for open extractors: heuristic matches between Wikipedia infobox attribute values and corresponding sentences to construct training data.

- Like TextRunner: avoid lexicalized features and handles unbounded set of semantic relations.

- System WOE (Web-based Open Information Extraction):
  - $WOE^{pos}$: POS tag features $\rightarrow$ as fast as TextRunner but better P & R
  - $WOE^{parse}$: dependency-parse features $\rightarrow$ even higher P & R, but 30 times slower.
Open Extractor - Problem Definition

• An open information extractor is:

  • a function from a document \( d \) to a set of triples \( \{<\text{arg}_1, \text{rel}, \text{arg}_2>\} \), where

    • \( \text{arg}_i \) is a noun phrase and \( \text{rel} \) is a textual fragment indicating an implicit, semantic relation between \( \text{arg}_1 \) and \( \text{arg}_2 \)

• Restriction: Triples represent facts stated explicitly in the text, no inference of implicit facts

• Assumption: all relational instances are stated within a single sentence
Self-Supervision using Wikipedia

- Goal: Learn an open extractor without direct supervision!

- Input: Wikipedia (as source for sentences) and DBpedia (as source for cleaned infoboxes)

- Output: unlexicalized and relation-independent open extractor

- Objection: Extractor should generalize beyond Wikipedia, e.g., that can handle the general Web
Wikipedia-based Open IE

- Key idea: automatic construction of training examples by heuristically matching Wikipedia infobox values and corresponding text.

- Used to generate generalized relation-independent extractors.

From infoboxes to a training set

Clearfield County was created in 1804 from parts of Huntingdon and Lycoming Counties but was administered as part of Centre County until 1812.

Its county seat is Clearfield.

2,972 km² (1,147 mi²) of it is land and 17 km² (7 mi²) of it (0.56%) is water.

As of 2005, the population density was 28.2/km².
Architecture of WOE

• Preprocessor converts raw Wikipedia text into a sequence of sentences

• The matcher constructs training data from attribute-value pairs of infoboxes and matching sentences

• Learner acquires the open extractors using either parser features or POS features.

Figure 1: Architecture of WOE.
Preprocessor

- OpenNLP for sentence splitting

- NLP annotation:
  - OpenNLP for POS tagging and NP chunks
  - Stanford Parser for dependency parse; hyperlinked anchor text handled as a single token (using underscore)

- Compiling synonymies (to increase recall of the matcher)
  - Wikipedia articles contain different mentions of same entities (across pages and between infobox and Wikipedia page)
  - Wikipedia redirection pages and backward links are used to construct automatically synonym sets.
Matcher - Constructing Training Data from Infoboxes

- Given a Wikipedia page with an infobox - assumption here: such a page describes an entity - iterate through all its attributes looking for a unique sentence that contain references to both the subject of the article and the attribute value (or its synonym).

- „Stanford University“ article:  
  match <established, 1891> with „The university was founded in 1891 by ...“  
  → <arg1=Stanford University, rel=???, arg2=1891>

- Ordered heuristics for matching the subject, e.g.,  
  *full match*, *synonym match*, *partial match* (prefix/suffix of the entity‘s name),  
  *patterns of „the <type>“* (type identification using simple patterns from Wikipedia), *article‘s most frequent pronoun*.

- Use DBpedia‘s cleaned infobox data (1,027,744 articles) as basis for attribute-value pairs → leads to 301,962 labeled sentences
Extraction with Parser Features

- Construction of relation text:
  - corePath = shortest dependency path between arg₁ and arg₂
    (expandPath = adding all adverbial and adjectival modifiers, „neg“ and „auxpass“ labels of the root node)
  - select all tokens of expanded Path as value for rel („was not born“ in our example)

- Generalization of patterns:
  - ignoring corePaths that don’t start with subject like dependencies, s.a. nsubj, nsubj-pass → 259,046 corePaths
  - generalized-corePaths: substitute lexical words by their POS; map all noun tags to N, verb tags to V, prep tags to prep etc.
  - yields a database DBₚ of 29,005 distinct patterns
  - each pattern receives its number of matching sentences as frequency p (311 patterns have fₚ ≥ 100, and 3,519 have fₚ ≥ 5)
WOEparse - A Simple Pattern Classifier

• Lookup the generalized-corePath of a test triple in DB_p, and compute normalized logarithmic frequency as the probability:

\[ w(p) = \frac{\max(log(f_p) - log(f_{min}), 0)}{log(f_{max}) - log(f_{min})} \]

• \( f_{max} \) = max. freq. of patterns in DB_p, \( f_{min} \) = a controlling threshold

• Example: „Dan was not born in Berkeley“
  let \( f_{max} = 54,274 \), \( f_{min} = 1 \), and \( f_p = 31,767 \), where p = “N \text{nsubjpass} V \text{prep} N”.

• then \( w(p) = 0.95 \)

• Performance: 79% to 90% higher F-measure than TextRunner! But: TextRunner is still 30X faster!
WOE\textsuperscript{pos} - Extraction with Shallow Patterns

- Like TextRunner learn a CRF extractor WOE\textsuperscript{pos} based on shallow features. (CRF - conditional random fields - classifier from the Mallet ML toolkit.)

- However, use \textbf{generated labeled examples}, where TextRunner learns from a small set of hand-written labeled sentences.

- Use same matching sentence set behind DB\textsubscript{p}:
  
  - positive examples: use matching arg\textsubscript{1} and arg\textsubscript{2} as ends of text sequences, rel used from expandedPath

  - negative examples: random NP pairs in other sentences which are not covered via DB\textsubscript{p}

- Features: POS-tags, regular expressions, combined features from 6-sized window

- Performance: 15\% to 34\% higher F-measure than TextRunner! And: as fast as TextRunner!!

Freitag, 27. Januar 2012
Experiments

- 300 randomly selected sentences form three corpora:
  - WSJ from Penn Treebank
  - Wikipedia
  - general Web

- Each sentence was examined by two people to label all reasonable triples. These triples are mixed with pseudo-negative ones.

- Using Amazon Mechanical Turk for verification: each triple was examined by 5 workers, and was labeled positive when more than 3 workers marked it as positive.
Overall Performance Analysis

- \( \text{WOE}^{\text{pos}} \) is better than TextRunner, especially on precision.

- \( \text{WOE}^{\text{parse}} \) performed best, especially in recall; parser helps to handle complicated and long-distance relations in difficult sentences; further experiments showed that parser errors have negligible effect!

- Extraction errors (numbers based on WSJ corpus): 1) incorrect args from NP chunking (18.6%), 2) erroneous parses (11.9%), 3) inaccurate meaning (27.1%), 4) a pattern inapplicable for the test sentence (42.4%)

Figure 2: \( \text{WOE}^{\text{pos}} \) performs better than TextRunner, especially on precision. \( \text{WOE}^{\text{parse}} \) dramatically improves performance, especially on recall.
Further Performances

• WOE$_{\text{parse}}$ achieves best results (Figure 3), due to deeper parsing, however also has highest costs (Figure 5; 0.679 sec/sent vs. 0.022 sec/sent)

• WOE$_{\text{parse}}$ decreases more slowly with sentence length, also due to deeper parsing (figure 4)
Self-supervision with Wikipedia Results in Better Training Data

- Generating training examples from Wikipedia differently: 1) tr = pos/neg examples selected using TextRunners patterns, 2) w = WOE’s heuristic infobox matchers, 3) r = randomly

- CRF_{h1-h2}, where h_i = {tr,w,r}; using same feature set as TextRunner

Figure 6: Matching sentences with Wikipedia infoboxes results in better training data than the hand-written rules used by TextRunner.
Design Criteria for WOS\textsuperscript{parse}

Figure 7: Filtering prepositional phrase attachments (PPa) shows a strong boost to precision, and we see a smaller boost from enforcing a lexical ordering of relation arguments (1 < 2).

- two interesting design choices:

- require arg\textsubscript{1} to appear before arg\textsubscript{2} (1 < 2)

- allow corePath to contain PP-attachment (PPa)
Conclusion

• Novel self-supervised method using automatically generated training examples from Wikipedia

• CRF extractor trained with shallow features (WOE\textsuperscript{pos}) and dependency-parse features (WOE\textsuperscript{parse})

• Much better P&R compared to TextRunner

• Two sources of WOE’s strong performance:
  1) the Wikipedia heuristic is responsible for the bulk of WOE’s improved accuracy
  2) dependency-parse features are highly informative when performing unlexicalized extraction.