

Relation Extraction

Basic methodologies

- Hand-built patterns
- Supervised methods
- Distant supervision
- Unsupervised methods

Hand-built patterns

- Hearst's lexico-syntactic patterns

Y such as X ((, X)* (, and/or) X)

such Y as X...

X... or other Y

X... and other Y

Y including X...

Y, especially X...

Problems with hand-built patterns

- Requires hand-building patterns for each relation!
 - hard to write; hard to maintain
 - there are zillions of them
 - domain-dependent
- Don't want to do this for all possible relations!
- Plus, we'd like better accuracy
 - Hearst: 66 accuracy on hyponym extraction

Supervised Methods

- Defining an inventory of output labels
 - Relation detection: true/false
 - Relation classification: located-in, employee-of, inventor-of, ...
- Collecting labeled training data: MUC, ACE, ...
- Defining a feature representation: words, entity types, ...
- Choosing a classifier: Naïve Bayes, MaxEnt, SVM, ...
- Evaluating the results

Feature based [Zhou et al., ACL'05]

- Lightweight features — require little pre-processing
 - Bags of words & bigrams between, before, and after the entities
 - Stemmed versions of the same
 - The types of the entities
 - The distance (number of words) between the entities
- Medium-weight features — require base phrase chunking
 - Base-phrase chunk paths
 - Bags of chunk heads
- Heavyweight features — require full syntactic parsing
 - Dependency-tree paths
 - Constituent-tree paths
 - Tree distance between the entities
 - Presence of particular constructions in a constituent structure

Features: words

- *American Airlines*, a unit of AMR, immediately matched the move, spokesman *Tim Wagner* said.
- Bag-of-words features
 - WM1 = {American, Airlines}, WM2 = {Tim, Wagner}
- Head-word features
 - HM1 = Airlines, HM2 = Wagner, HM12 = Airlines+Wagner
- Words in between
 - WBNULL = false, WBFL = NULL, WBF = a, WBL = spokesman,
 - WBO = {unit, of, AMR, immediately, matched, the, move}
- Words before and after
 - BM1F = NULL, BM1L = NULL, AM2F = said, AM2L = NULL
- Word features yield good precision (69 %), but poor recall (24 %)

- WM1: bag-of-words in M1
- HM1: head word of M1
- WM2: bag-of-words in M2
- HM2: head word of M2
- HM12: combination of HM1 and HM2
- WBNUL: when no word in between
- WBFL: the only word in between when only one word in between
- WBF: first word in between when at least two words in between
- WBL: last word in between when at least two words in between
- WBO: other words in between except first and last words when at least three words in between
- BM1F: first word before M1
- BM1L: second word before M1
- AM2F: first word after M2
- AM2L: second word after M2

Features: NE type & mention level

- *American Airlines*, a unit of AMR, immediately matched the move, spokesman *Tim Wagner* said.
- Named entity types (ORG, LOC, PER, etc.)
 - ET1 = ORG, ET2 = PER, ET12 = ORG-PER
- Mention levels (NAME, NOMINAL, or PRONOUN)
 - ML1 = NAME, ML2 = NAME, ML12 = NAME+NAME
- Named entity type features help recall a lot (+8)
- Mention level features have little impact

Features: overlap

- *American Airlines*, a unit of AMR, immediately matched the move, spokesman *Tim Wagner* said.
- Number of mentions and words in between
 - #MB = 1, #WB = 9
- Does one mention include in the other?
 - $M1 > M2 = \text{false}$, $M1 < M2 = \text{false}$
- Conjunctive features
 - $ET12 + M1 > M2 = \text{ORG-PER} + \text{false}$
 - $ET12 + M1 < M2 = \text{ORG-PER} + \text{false}$
 - $HM12 + M1 > M2 = \text{Airlines} + \text{Wagner} + \text{false}$
 - $HM12 + M1 < M2 = \text{Airlines} + \text{Wagner} + \text{false}$
- These features hurt precision a lot (-10), but also help recall a lot (+8)

Features: base phrase chunking

- *American Airlines*, a unit of AMR, immediately matched the move, spokesman *Tim Wagner* said.

[_{NP} American Airlines], [_{NP} a unit] [_{PP} of] [_{NP} AMR], [_{ADVP} immediately] [_{VP} matched] [_{NP} the move], [_{NP} spokesman Tim Wagner] [_{VP} said].

- *Phrase heads before and after*
 - CPHBM1F = NULL, CPHBM1L = NULL, CPHAM2F = said, CPHAM2L = NULL
- *Phrase heads in between*
 - CPHBNULL = false, CPHBFL = NULL, CPHBF = unit, CPHBL = move
 - CPHBO = {of, AMR, immediately, matched}
- *Phrase label paths*
 - CPP = [NP, PP, NP, ADVP, VP, NP]
- CPPH = NULL
- *These features increased both precision & recall by 4-6*

Features: syntactic features

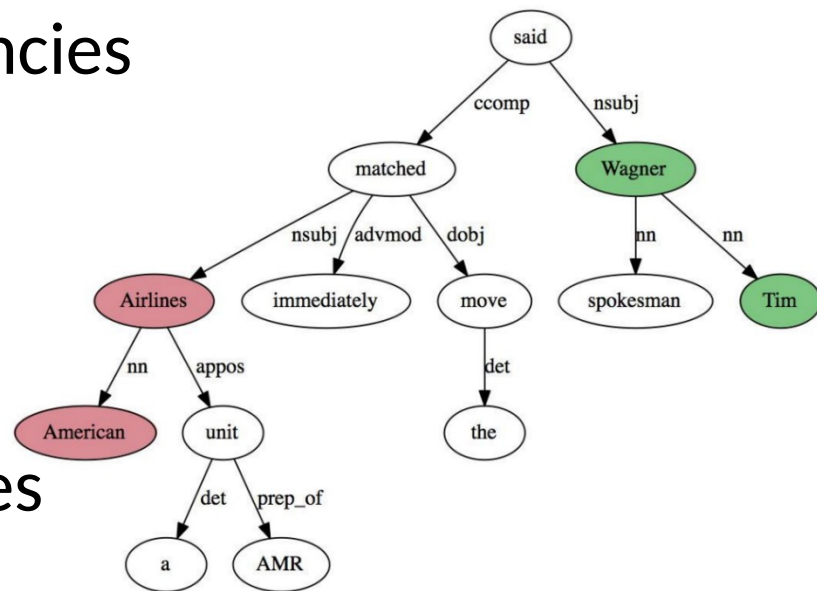
- Features of mention dependencies

- ET1DW1 = ORG:Airlines
- H1DW1 = matched:Airlines
- ET2DW2 = PER:Wagner
- H2DW2 = said:Wagner

- Features describing entity types and dependency tree

- ET12SameNP = ORG-PER-false
- ET12SamePP = ORG-PER-false
- ET12SameVP = ORG-PER-false

- These features had disappointingly little impact!



Results

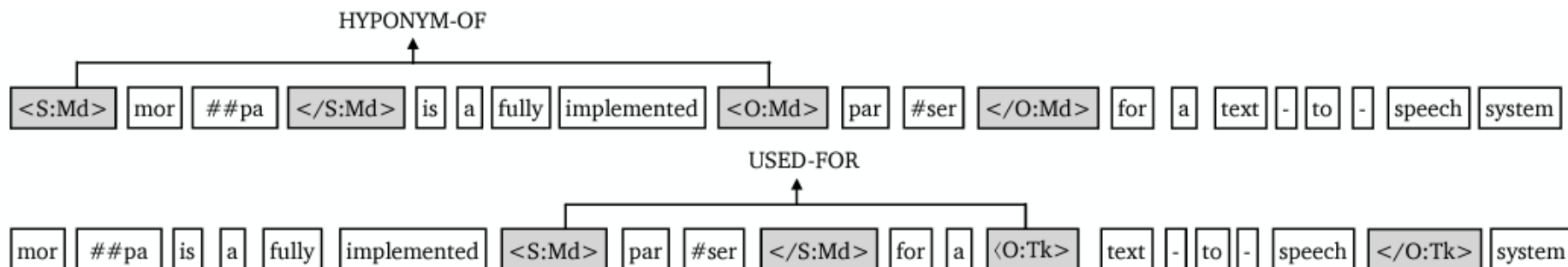
Features	P	R	F
Words	69.2	23.7	35.3
+Entity Type	67.1	32.1	43.4
+Mention Level	67.1	33.0	44.2
+Overlap	57.4	40.9	47.8
+Chunking	61.5	46.5	53.0
+Dependency Tree	62.1	47.2	53.6
+Parse Tree	62.3	47.6	54.0
+Semantic Resources	63.1	49.5	55.5

Table 2: Contribution of different features over 43 relation subtypes in the test data

Deep Learning

- Same intuitions, different models
 - (2012-13) Recursive NN: dependency tree
[Socher et al., EMNLP'12]
 - (2014-15) CNN: shortest dependency path
[Liu et al., ACL'15]
 - (2015+) LSTM: shortest dependency path,
lexical/syntactic/semantic features
[Xu et al., EMNLP'15][Shwartz et al., ACL'16][Nguyen, NAACL'16]

Bert-based RE



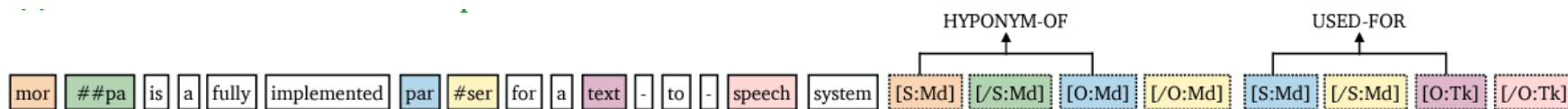
- Entity markers
 - S/O indicate subject/object
 - Md/Tk indicate entity types
- span-pair representation

$$\mathbf{h}_r(s_i, s_j) = [\hat{\mathbf{x}}_{\widehat{\text{START}}(i)}; \hat{\mathbf{x}}_{\widehat{\text{START}}(j)}]$$

- feedforward network to predict

A Frustratingly Easy Approach for Entity and Relation Extraction

Bert-based RE



- Batch mode to speed-up
- The tokens of the same color share the positional embeddings

Different ways to include entity info

- Entity mask: [SUBJ-TYPE] or [OBJ-TYPE] to mask the subject or object entities in the original text
- Entity marker: [E1], [/E1] and [E2], [/E2] to enclose the subject and object entities
- Entity marker (punct): @ SUBJ @ ... # OBJ #
- Typed entity marker
- Typed entity marker (punct)

Technique	Names	Spans	NER Types
Entity mask	✗	✓	✓
Entity marker	✓	✓	✗
Entity marker (punct)	✓	✓	✗
Typed entity marker	✓	✓	✓
Typed entity marker (punct)	✓	✓	✓

Results

Method	Input Example	BERT _{BASE}	BERT _{LARGE}	RoBERTa _{LARGE}
Entity mask	[SUBJ-PERSON] was born in [OBJ-CITY].	69.6	70.6	60.9
Entity marker	[E1] Bill [/E1] was born in [E2] Seattle [/E2].	68.4	69.7	70.7
Entity marker (punct)	@ Bill @ was born in # Seattle #.	68.7	69.8	71.4
Typed entity marker	$\langle S:PERSON \rangle$ Bill $\langle /S:PERSON \rangle$ was born in $\langle O:CITY \rangle$ Seattle $\langle /O:CITY \rangle$.	71.5	72.9	71.0
Typed entity marker (punct)	@ * person * Bill @ was born in # ^ city ^ Seattle #.	70.9	72.7	74.6

An Improved Baseline for Sentence-level
Relation Extraction

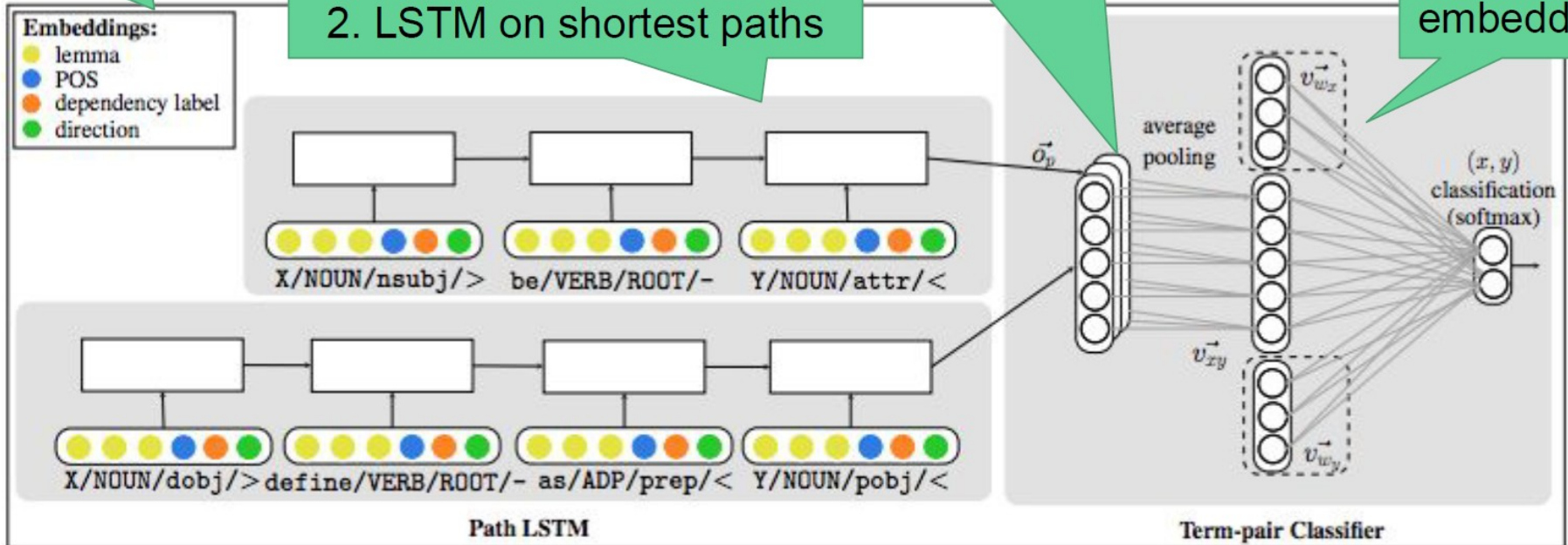
Hypernym Relation

1. Diff features

2. LSTM on shortest paths

3. Combine all paths

4. Term embedding

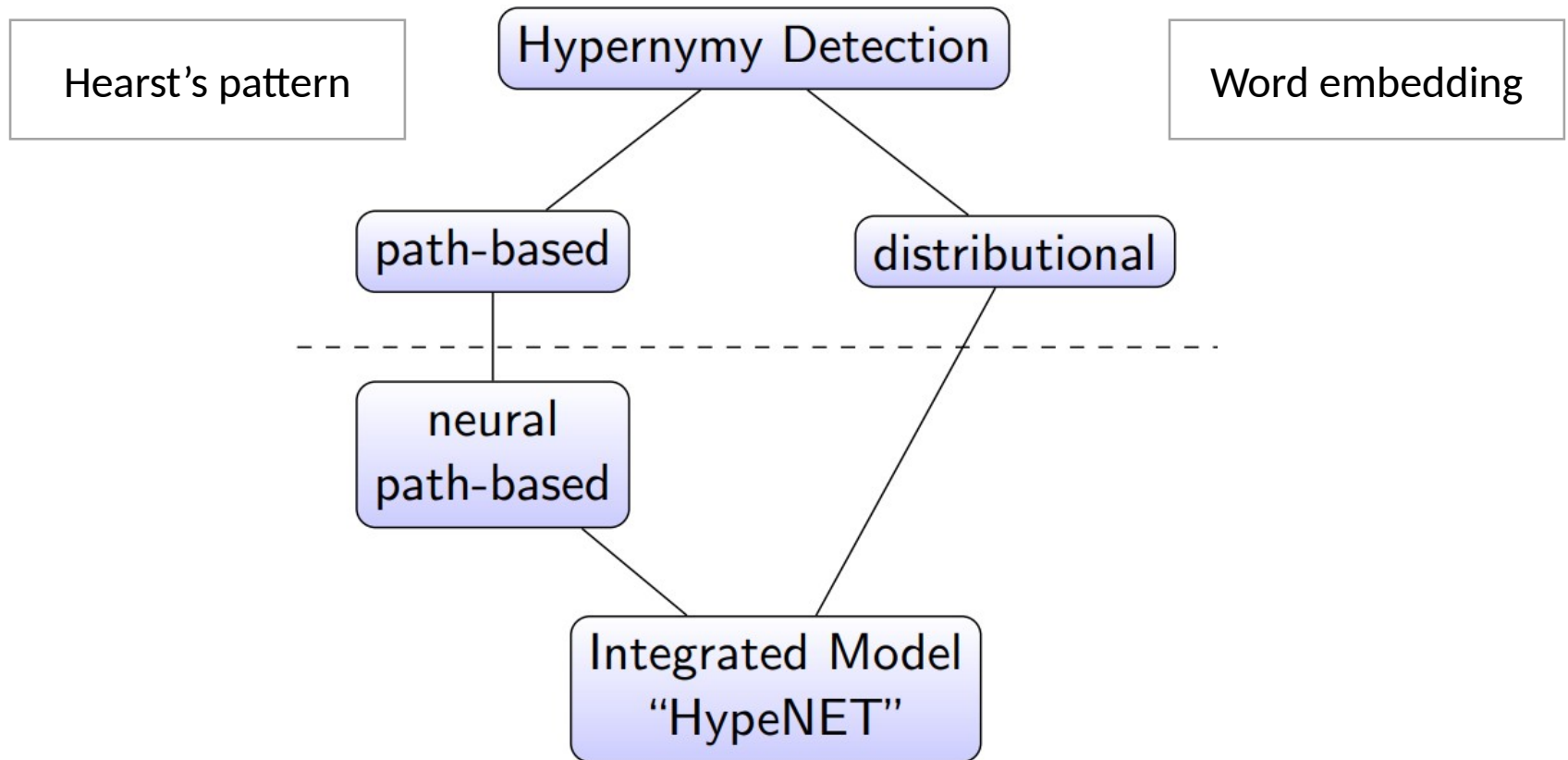


Quality in identifying hypernyms: Prec = 0.9, Rec = 0.9

Hypernym Detection Task

- Hypernym: A semantic relation between two terms (x, y)
 - the hyponym (x) is a type of / instance of the hypernym (y)
 - e.g. (pineapple, fruit), (green, color), (Obama, president)
- Given two terms, x and y, decide whether y is a hypernym of x
 - in some senses of x and y, e.g. (apple, fruit), (apple, company)

Key idea



Path Representation

Split each path to edges

X is a Y ⇒
'X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<' ⇒
'X/NOUN/nsubj/>' 'be/VERB/ROOT/-' 'Y/NOUN/attr/<'

- Each edge consists of 4 components:

dependent lemma / dependent POS / dependency label / direction

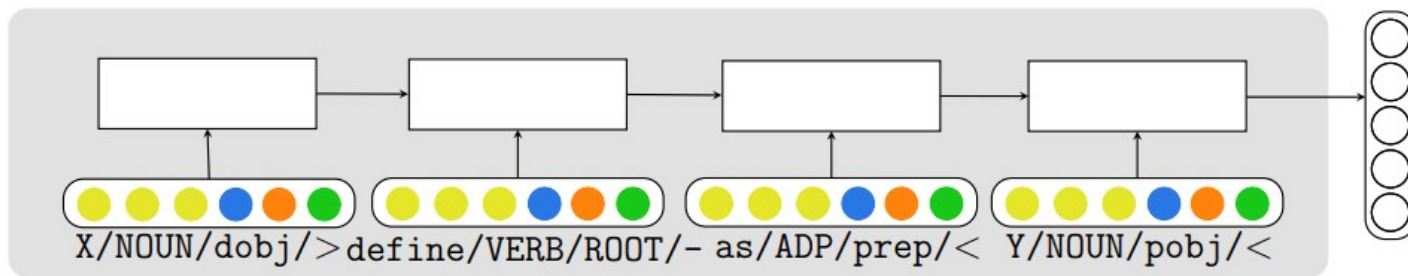
- We learn embedding vectors for each component
 - Lemma embeddings are initialized with pre-trained word embeddings
- The edge's vector is the concatenation of its components' vectors:



- Generalization: similar edges should have similar vectors!

Path Representation

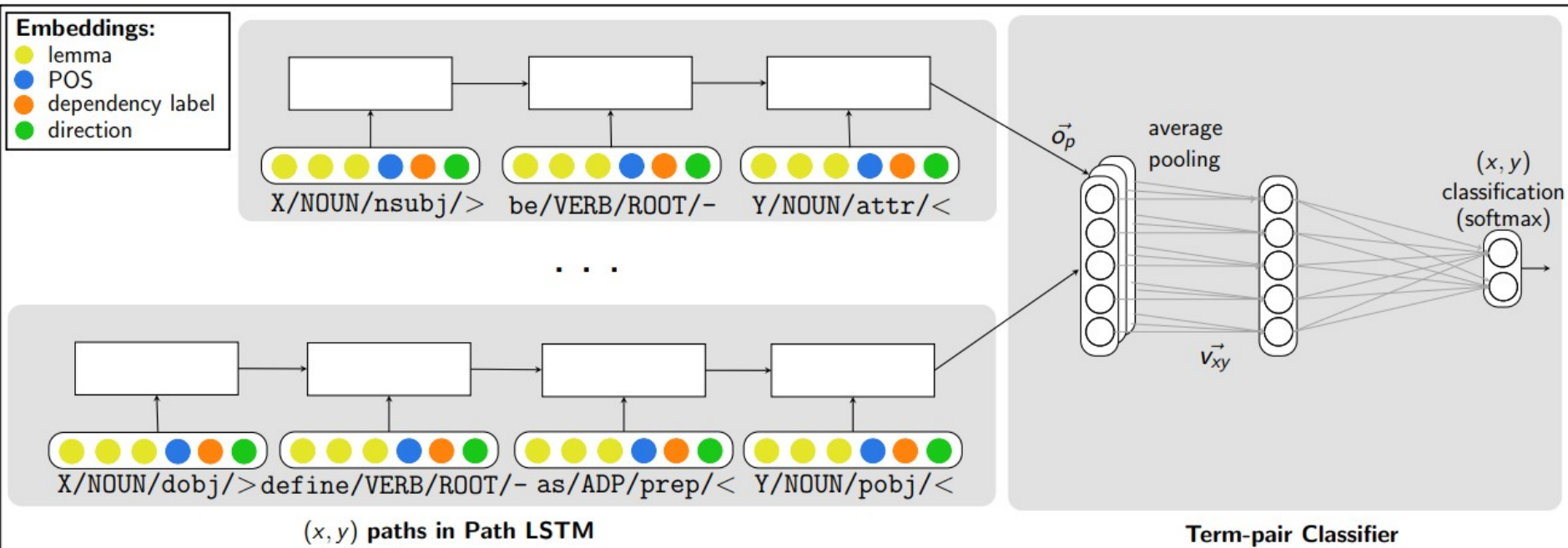
Feed the edges sequentially to an LSTM



- Use the last output vector as the path embedding
- The LSTM may focus on edges that are more informative for the classification task, while ignoring others

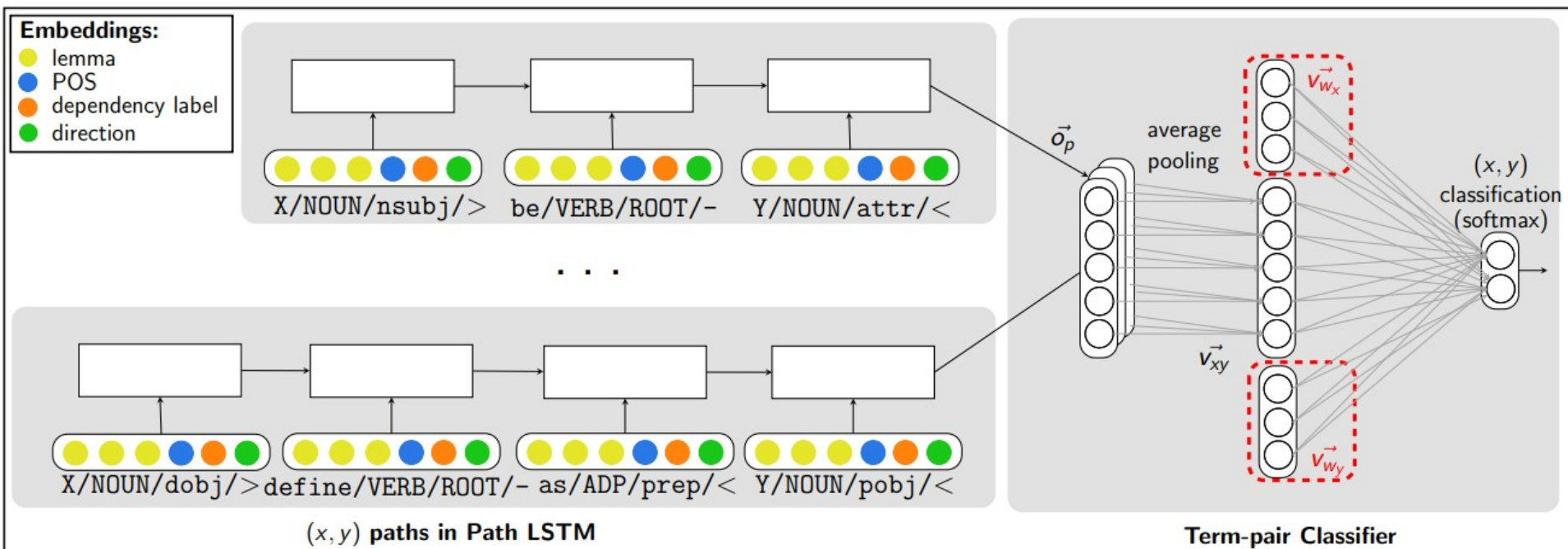
Term-pair Classification by path

- The LSTM encodes a single path
- Each term-pair has multiple paths
 - Represent a term-pair as its averaged path embedding
- Classify for hypernymy (path-based network):



Integrating Distributional Information

- Integrated network: add distributional information
 - Simply concatenate x and y's word embeddings to the averaged path
- Classify for hypernymy (integrated network):



Supervised RE: summary

- Supervised approach can achieve high accuracy
 - At least, for some relations
 - If we have lots of hand-labeled training data
- But has significant limitations!
 - Labeling 5,000 relations (+ named entities) is expensive
 - Doesn't generalize to different relations
- Next: beyond supervised relation extraction
 - Distantly supervised relation extraction
 - Unsupervised relation extraction
 - Semi-supervised relation extraction

Distant supervision

[Mintz et al., ACL'09]

Corpus Text

Bill Gates founded Microsoft in 1975.
Bill Gates, founder of Microsoft, ...
Bill Gates attended Harvard from ...
Google was founded by Larry Page ...

Freebase

(Bill Gates, Founder, Microsoft)
(Larry Page, Founder, Google)
(Bill Gates, CollegeAttended, Harvard)

Training Data

(Bill Gates, Microsoft)
Label: Founder
Feature: X founded Y
Feature: X, founder of Y

(Bill Gates, Harvard)
Label: CollegeAttended
Feature: X attended Y

For negative examples, sample
unrelated pairs of entities.

[Adapted example from Luke Zettlemoyer]

Distant supervision paradigm

- Like supervised classification:
 - Uses a classifier with lots of features
 - Supervised by detailed hand-created knowledge
 - Doesn't require iteratively expanding patterns
- Like unsupervised classification:
 - Uses very large amounts of unlabeled data
 - Not sensitive to genre issues in training corpus

Basic Procedure

- 1 For each relation
- 2 For each tuple in big database
- 3 Find sentences in large corpus with both entities
- 4 Extract frequent features (parse, words, etc)
- 5 Train supervised classifier using thousands of patterns

Born-In

<Edwin Hubble, Marshfield>
<Albert Einstein, Ulm>

Hubble was born in Marshfield
Einstein, born (1879), Ulm
Hubble's birthplace in Marshfield

PER was born in LOC
PER, born (XXXX), LOC
PER's birthplace in LOC

$P(\text{born-in} \mid f_1, f_2, f_3, \dots, f_{70000})$

Potential Issues

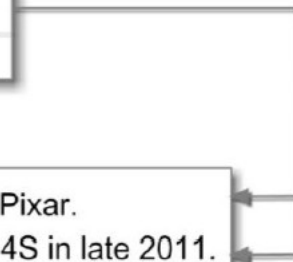
- The distant supervision assumption is too strong and causes the wrong label problem. (A sentence that mentions two entities does not necessarily express their relation in a knowledge base)
- Since errors exist in NLP tools, the use of traditional features leads to error propagation or accumulation

Freebase

Relation	Entity1	Entity2
/business/company/founders	Apple	Steve Jobs
...

Mentions from free texts

1. Steve Jobs was the co-founder and CEO of Apple and formerly Pixar.
2. Steve Jobs passed away the day before Apple unveiled iPhone 4S in late 2011.



Piecewise Convolutional Neural Networks (PCNNs)

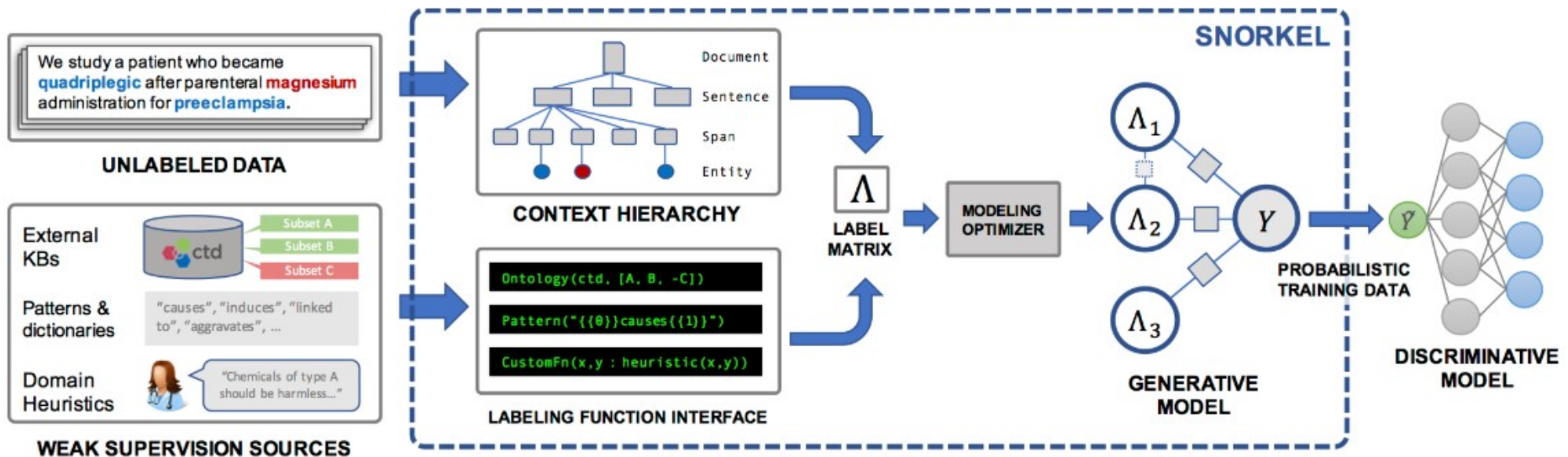
- To address the first problem, distant supervised relation extraction is treated as a **multi-instance problem**
- To address the second problem, adopt convolutional architecture to automatically learn relevant features without complicated NLP preprocessing

Multi-instance Learning

- The training set consists of many bags, and each contains many instances. The labels of the bags are known, but the labels of the instances in the bags are unknown
- For prediction, a bag is positively labeled if and only if the output of the network on at least one of its instances is assigned a positive label.

Semi supervision

- Data programming [Ratner et al., NIPS'16]



Unsupervised Method

- OpenIE (e.g., [Banko et al., IJCAI'07])

OpenIE

Predicate
Identification



Subject/Object
Identification



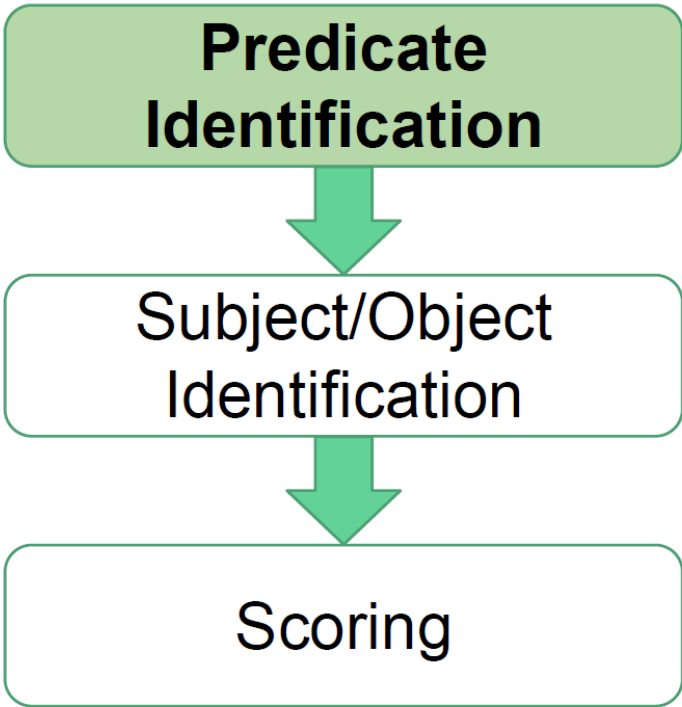
Scoring

Bill Gates founded
Microsoft in 1975.

Unsupervised Method

OpenIE

**Predicate
Identification**



```
graph TD; A[OpenIE] --> B[Predicate Identification]; B --> C[Subject/Object Identification]; C --> D[Scoring];
```

The diagram illustrates the OpenIE process as a three-step flowchart. It begins with 'OpenIE' at the top, followed by 'Predicate Identification' in a green box, then 'Subject/Object Identification' in a white box, and finally 'Scoring' in a white box. Green arrows indicate the downward flow from one step to the next.

Subject/Object
Identification

Scoring

Bill Gates founded
Microsoft in 1975.

- Predicate: longest sequence of words as light verb construction

Unsupervised Method

OpenIE

Predicate
Identification



**Subject/Object
Identification**



Scoring

Bill Gates founded
Microsoft in 1975.

- Predicate: longest sequence of words as light verb construction
- Subject: learn left and right boundary
- Object: learn right boundary

Unsupervised Method

OpenIE

Predicate
Identification



Subject/Object
Identification



Scoring

Bill Gates founded
Microsoft in 1975.

- Predicate: longest sequence of words as light verb construction
- Subject: learn left and right boundary
- Object: learn right boundary
- LR for triple confidence

Problem & Motivation

- Standard Information Extraction (IE):
 - *Yesterday, New York based Foo Inc. announced their acquisition of Bar Corp.*
 - `MergerBetween(company1, company2, date)`
- OpenIE
 - Information extracted: $t = (e_i, r_{i,j}, e_j)$
 - (`<proper noun>`, `works with`, `< proper noun >`)

ClausIE

Table 1: Patterns and clause types (based on [15]).

Pattern	Clause type	Example	Derived clauses
Basic patterns			
S_1 : SV_i	SV	AE died.	(AE, died)
S_2 : SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3 : SV_cC	SVC	AE is smart.	(AE, is, smart)
S_4 : $SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5 : $SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6 : $SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7 : $SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)
Some extended patterns			
S_8 : SV_iAA	SV	AE died in Princeton in 1955.	(AE, died) (AE, died, in Princeton) (AE, died, in 1955) (AE, died, in Princeton, in 1955)
S_9 : SV_eAA	SVA	AE remained in Princeton until his death.	(AE, remained, in Princeton) (AE, remained, in Princeton, until his death)
S_{10} : SV_cCA	SVC	AE is a scientist of the 20th century.	(AE, is, a scientist) (AE, is, a scientist, of the 20th century)
S_{11} : $SV_{mt}OA$	SVO	AE has won the Nobel Prize in 1921.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)
S_{12} : $ASV_{mt}O$	SVO	In 1921, AE has won the Nobel Prize.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)

S: Subject, V: Verb, C: Complement, O: Direct object, O_i : Indirect object, A: Adverbial, V_i : Intransitive verb, V_c : Copular verb, V_{ct} : Extended-copular verb, V_{mt} : Monotransitive verb, V_{dt} : Ditransitive verb, V_{ct} : Complex-transitive verb

ClausIE

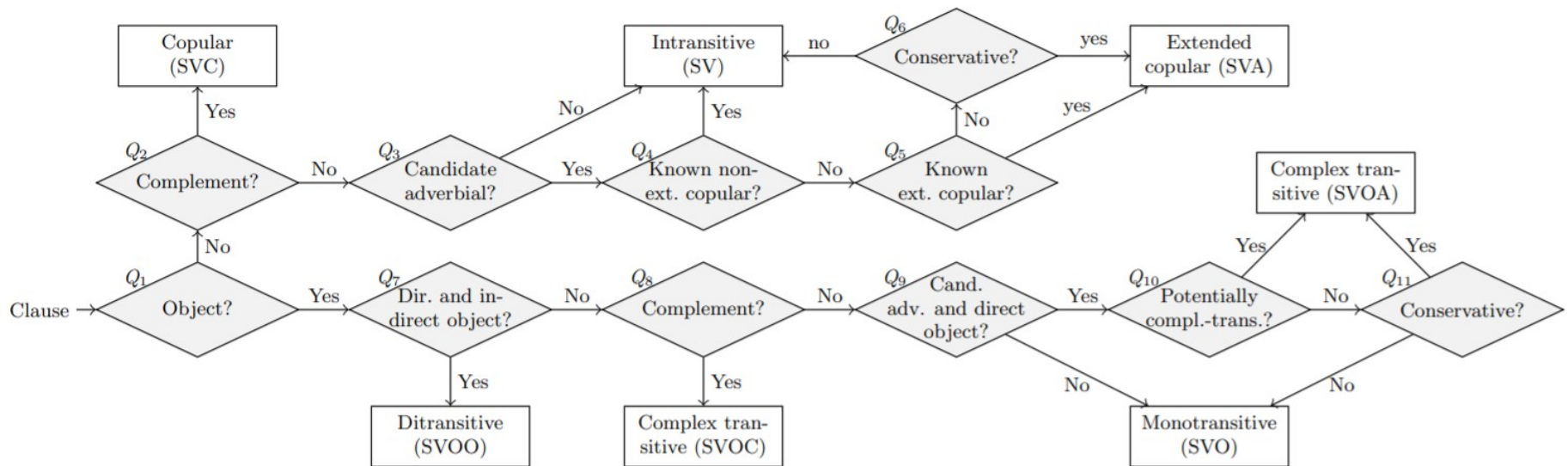


Figure 2: Flow chart for verb-type and clause-type detection

ClausIE

Step 1: Dependency Parsing

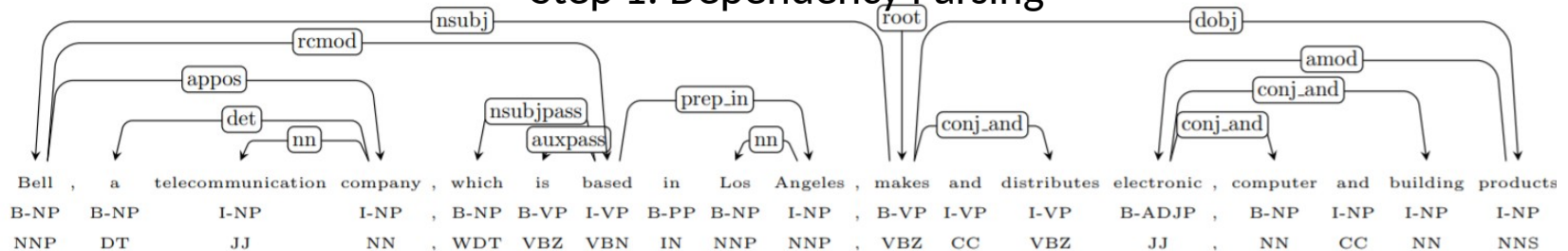


Figure 1: An example sentence with dependency parse, chunks, and POS tags (chunks by Apache OpenNLP)

Step 2: From Dependencies to Clauses

(S: *Bell*, V: *makes*, O: *products*),
 (S: *Bell*, V: *based*, A: *Angeles*),
 (S: *Bell*, V_c: “*is*”, C: *company*).

Step 3: Identifying Clause Types

(S: *Bell*, V: *makes*, O: *products*),
 (S: *Bell*, V: *based*, A!: *Angeles*),
 (S: *Bell*, V: “*is*”, A!: *company*),

where “A!” indicates essential adverbials

Step 4: Coordinated conjunctions (CC)

(S: *Bell*, V: *makes*, O: [*electronic*] *products*),
 (S: *Bell*, V: *makes*, O: [*computer*] *products*),
 (S: *Bell*, V: *makes*, O: [*building*] *products*),
 (S: *Bell*, V: *distributes*, O: [*electronic*] *products*),
 (S: *Bell*, V: *distributes*, O: [*computer*] *products*),
 (S: *Bell*, V: *distributes*, O: [*building*] *products*).

Step 5: From Clauses to Propositions

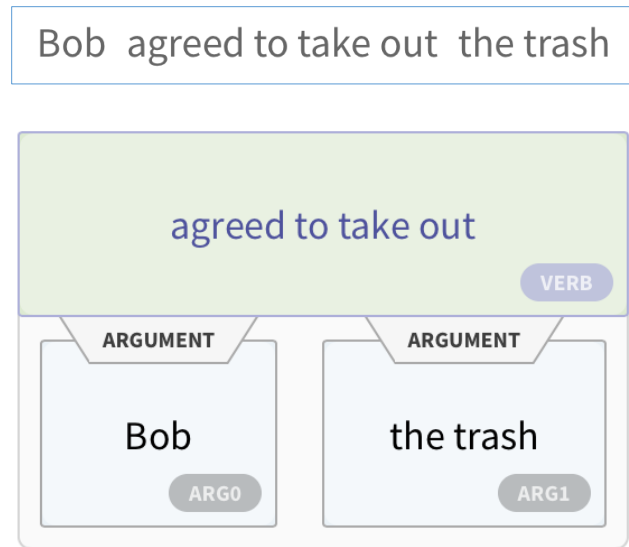
(“*Bell*”, “*is*”, “*a telecommunication company*”),
 (“*Bell*”, “*is based*”, “*in Los Angeles*”),
 (“*Bell*”, “*makes*”, “*electronic products*”),
 (“*Bell*”, “*makes*”, “*computer products*”),
 (“*Bell*”, “*makes*”, “*building products*”),
 (“*Bell*”, “*distributes*”, “*electronic products*”),
 (“*Bell*”, “*distributes*”, “*computer products*”),
 (“*Bell*”, “*distributes*”, “*building products*”).

Stanford OpenIE

- short, entailed clauses from sentences
- Training Data Generation
 - Take 66 880 sentences (newswire, newsgroups, Wikipedia).
 - Apply distant supervision to label relations in sentence.
 - Run exhaustive search.
 - Positive Labels: A sequence of actions which yields a known relation.
 - Negative Labels: All other sequences of actions.
- Features:
 - Edge label; incoming edge label.
 - Neighbors of governor; neighbors of dependent; number of neighbors.
 - Existence of subject/object edges at governor; dependent.
 - POS tag of governor; dependent.

TextRunner

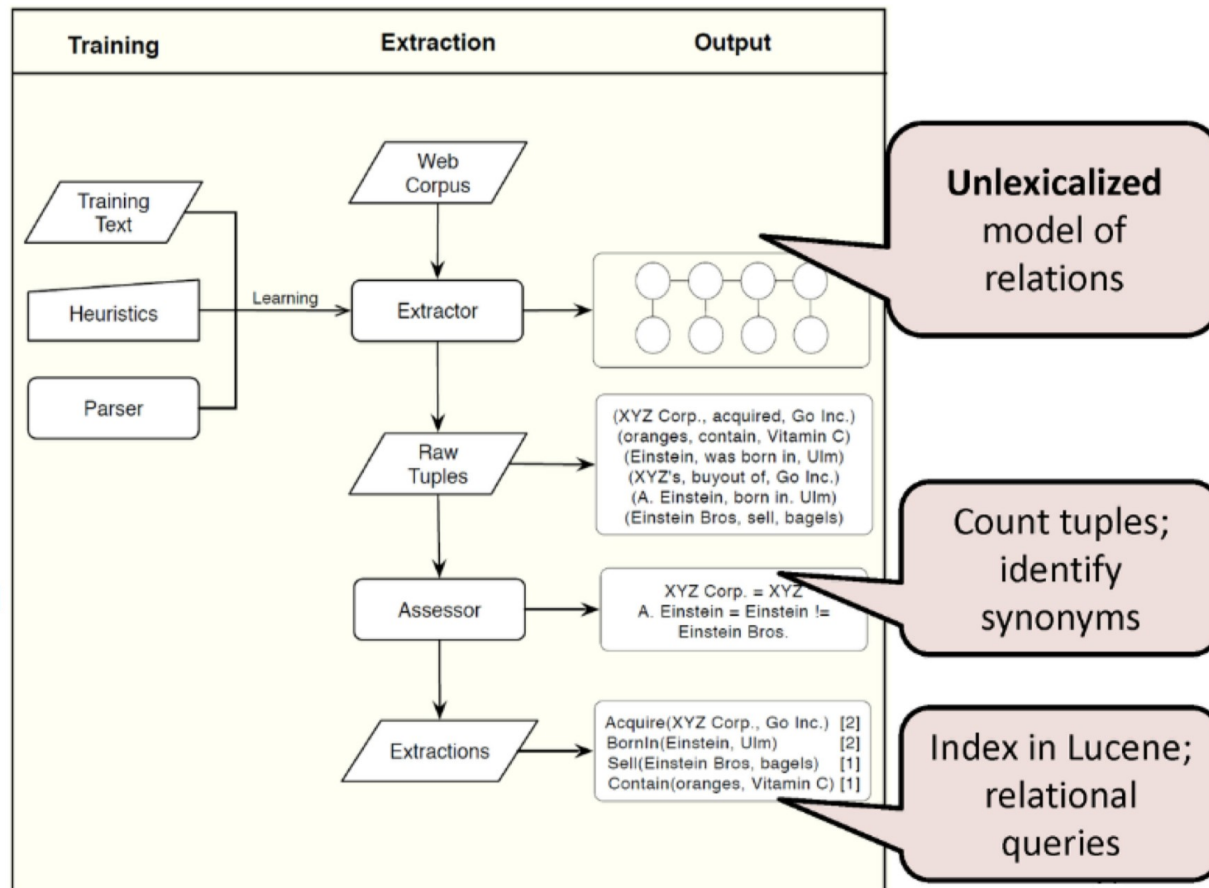
- Scalable & domain-independent
- Extracts relational tuples from text
- Indexed to support extraction via queries
- Input to TextRunner is corpus, output is set of extractions



TextRunner Architecture

- TextRunner consists of 3 key modules:
 - *Self-supervised learner*: given a corpus sample, learner outputs classifier that labels extractions
 - *Single Pass Extractor*: Makes single pass over entire corpus to extract tuples
 - *Self-supervised learner* classifies tuples
 - *Redundancy-based assessor*: computes probability of correct tuple instance

TextRunner Architecture



Shortcomings of TextRunner

Scientists from many universities are studying string theory, but Massachusetts Institute of Tech is one of the pioneers

- Use of linguistic parser to label extractions not scalable
 - If Massachusetts Institute of Tech is not defined in parser, learner may not classify as a trustworthy
 - WOE (Wu and Weld): does not need lexicalized features for self-supervised learning; higher precision & recall
- No definition on the form that entities may take
 - Massachusetts Institute of Tech may not be recognized as one entity
 - ReVerb (Fader, et al): improves precision & recall with relation-phrase identifier with lexical constrains

Bootstrap method

Extract cities:

it's underconstrained!!

Paris
Pittsburgh
Seattle
Cupertino

San Francisco
Austin
denial

anxiety
selfishness
Berlin



mayor of arg1
live in arg1

arg1 is home of
traits such as arg1

Bootstrapping

- <Mark Twain, Elmira> Seed tuple
- Grep (google) for the environments of the seed tuple
- “Mark Twain is buried in Elmira, NY.”
 - X is buried in Y
- “The grave of Mark Twain is in Elmira”
 - The grave of X is in Y
- “Elmira is Mark Twain’s final resting place”
 - Y is X’s final resting place.
- Use those patterns to grep for new tuples
- Iterate

Finding E-A-V and Typed Patterns

- ❑ Task 1: Finding E-A-V at the Instance Level
 - ❑ Stanford OpenIE [ACL'15], AI²'s Open IE-Ollie [EMNLP'12] Ignore entity-typing information!
 - ❑ Learn syntactic and lexical patterns of expressing relations
 - ❑ Input: “President Blaise Compaoré’s government of Burkina Faso was founded...”
 - ❑ Output: ⟨President Blaise Compaoré, **have**, government of Burkina Faso⟩ 🔍

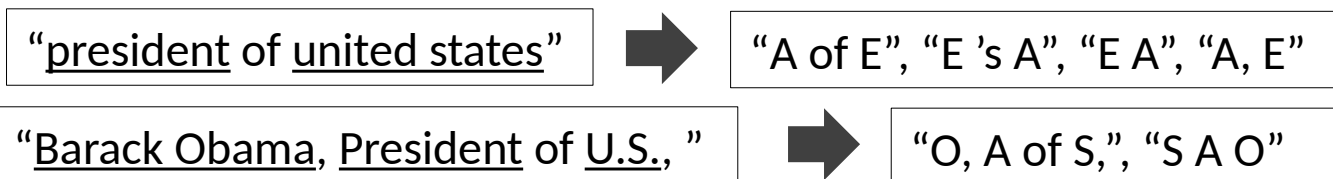
Finding E-A-V and Typed Patterns

- Task 2: Finding Typed Patterns

- Google's Biperpedia+ARI [VLDB'14, WWW'16],
ReNoun [EMNLP'15]:

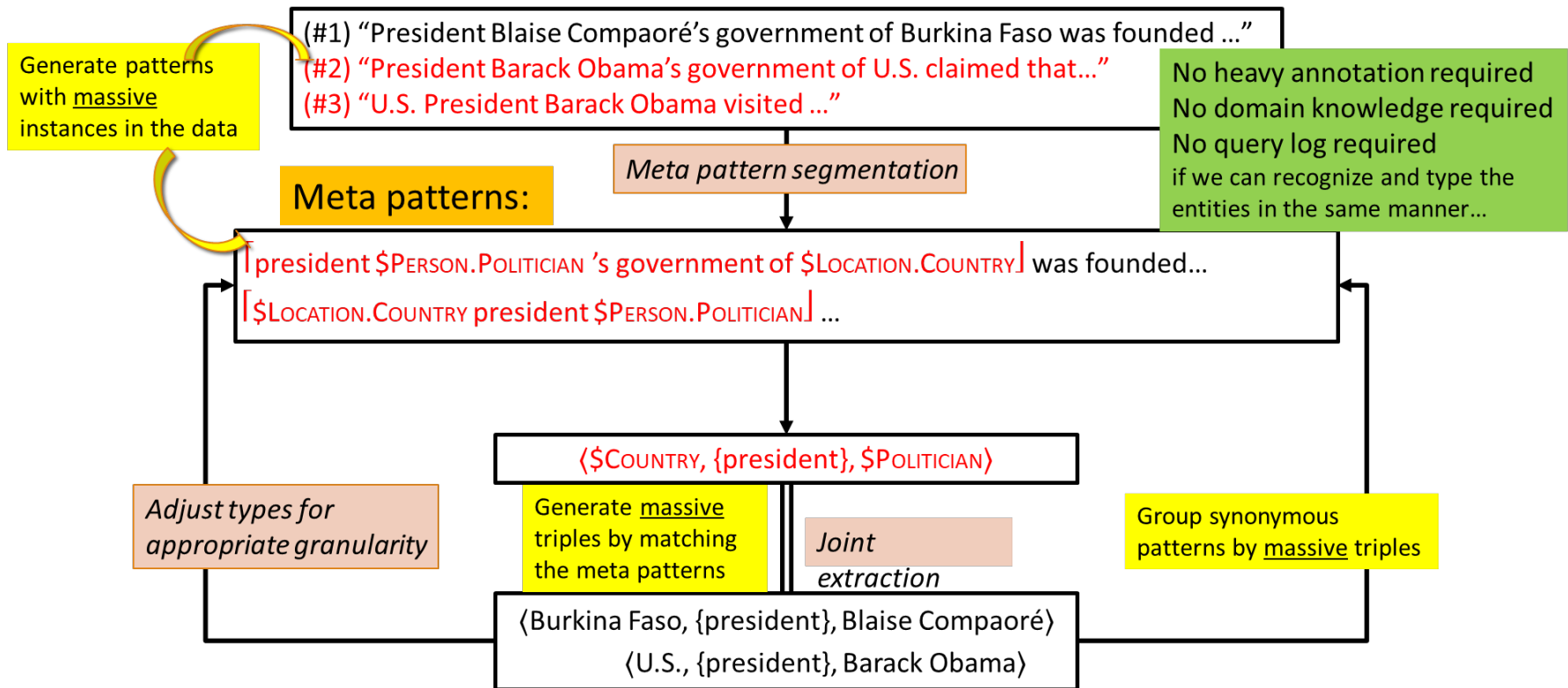
- Input: "...Sunday night, Burkina Faso..." and the "A, E" pattern

- Output: { \$COUNTRY, Sunday night } 🔍



Query log: Highly constrained and unavailable

Meta-Pattern Methodology



Pattern Discovery by Phrase Mining and Entity Typing

“President Blaise Compaoré’s government of Burkina Faso was founded ...”

Phrase mining (SegPhrase and AutoPhrase)

“president **blaise_compaoré**’s government of **burkina_faso** was founded ...”

Entity recognition and typing with Distant Supervision (ClusType)

“president **\$PERSON**’s government of **\$LOCATION** was founded ...”

Fine-grained typing (PLE by Ren et al. KDD’16)

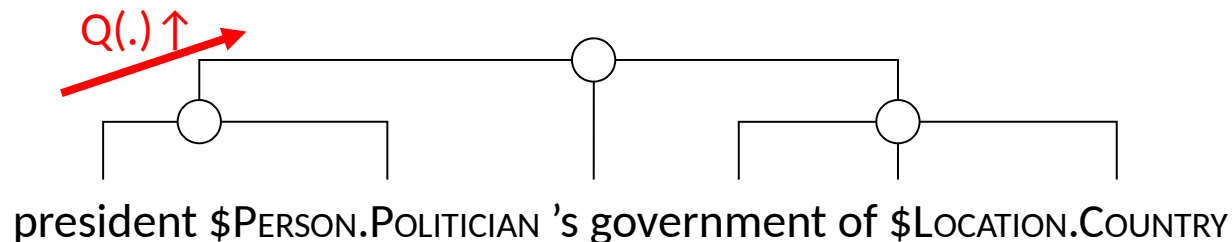
“president **\$PERSON.POLITICIAN**’s government of **\$LOCATION.COUNTRY** was founded ...”

Meta-Pattern Quality Assessment and Segmentation

A set of features:

- ✓ Frequency
- ✓ Concordance: “\$PERSON’s wife”
- ✓ Completeness: “\$COUNTRY president” vs. “\$COUNTRY president \$POLITICIAN”
- ✓ Informativeness: “\$PERSON and \$PERSON ” vs. “\$PERSON ’s wife, \$PERSON”

Regression Q(.): random forest with only 300 labels



Adjusting Types in Meta Patterns for Appropriate Granularity

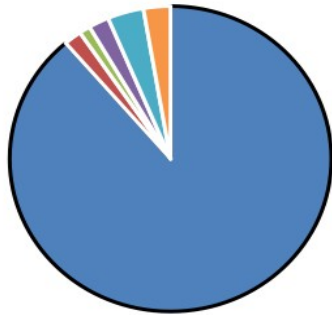
\$PERSON, \$DIGIT,

\$PERSON's age is \$DIGIT

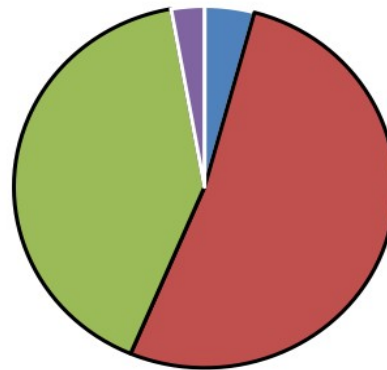
\$PERSON, a \$DIGIT -year-old

\$COUNTRY president \$POLITICIAN

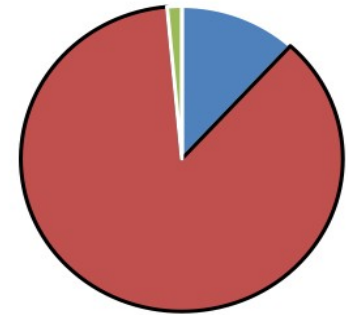
president \$POLITICIAN of \$COUNTRY



■ \$PERSON ■ \$ATTACKER
■ \$ARTIST ■ \$ATHLETE
■ \$POLITICIAN ■ \$VICTIM



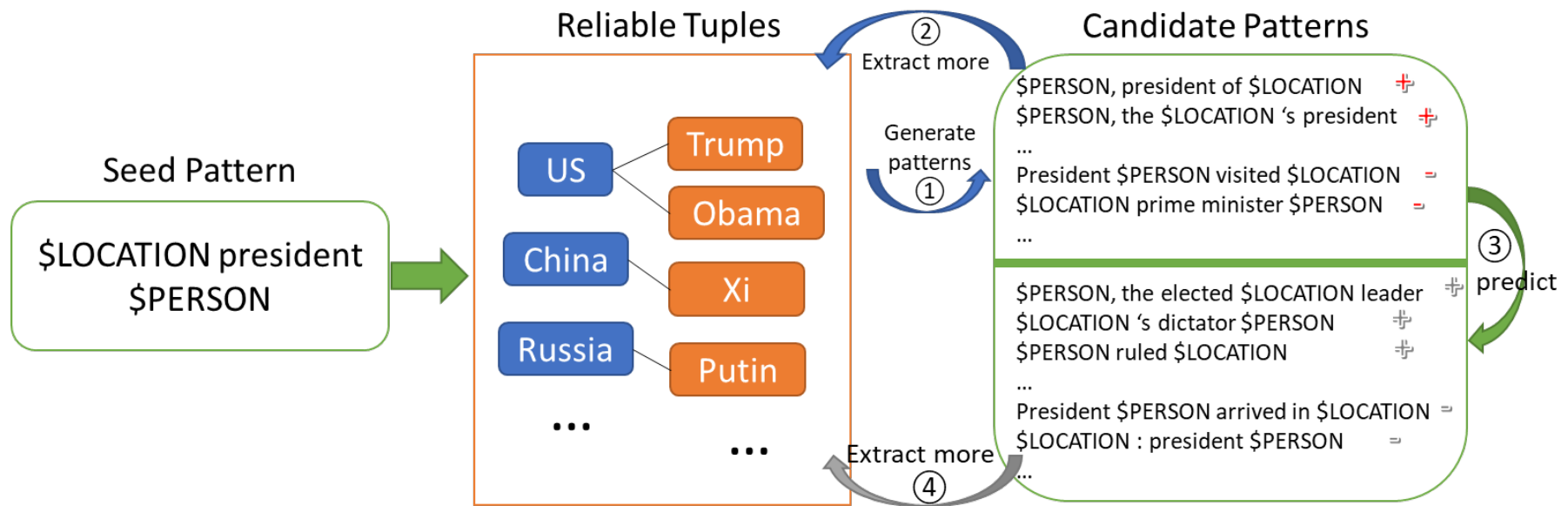
■ \$LOCATION ■ \$COUNTRY
■ \$ETHNICITY ■ \$CITY



■ \$PERSON
■ \$POLITICIAN
■ \$ARTIST

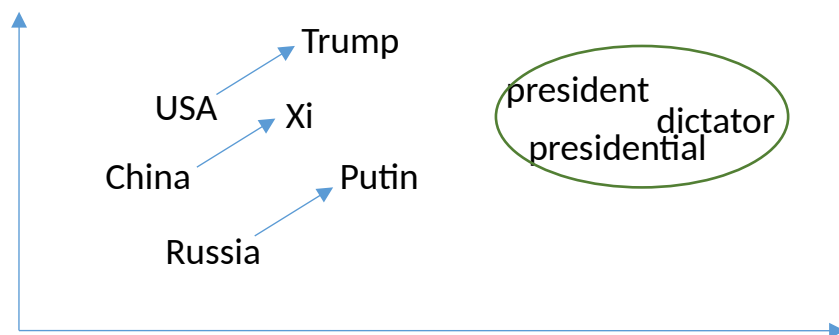
Pattern Grouping

- **Given** the text corpus, couple of seed patterns for a specific extraction task *on attribute*
- **Find** as many as possible reliable patterns and extractions $\langle \text{entity } e, \text{attribute } a, \text{value } v \rangle$



An Intuitive Solution

- **Reliable patterns are semantically similar to the seed patterns**
 - Joint consider pattern constructing words and extractions
 - Eg., \$Person , president of \$Country
 - Constructing words: president, of
 - Extractions: <Russian, Putin>, <China, Xi>, <USA, Trump>,...
- **Pattern embedding**
 - Adapting word embedding technique



An Intuitive Solution

- **Reliable patterns are semantically similar to the seed patterns**

- Joint consider pattern constructing words and extractions
- Eg., \$Person , president of \$Country
- Constructing words: president, of
- Extractions: <Russian, Putin>, <China, Xi>, <USA, Trump>,...

- **Pattern embedding**

- Adapting word embedding technique

-



$$\frac{1}{2}(\mathbf{v}(\textit{president}) + \mathbf{v}(\textit{of}))$$

An Intuitive Solution

- **Reliable patterns are semantically similar to the seed patterns**

- Joint consider pattern constructing words and extractions
- Eg., \$Person , president of \$Country
- Constructing words: president, of
- Extractions: <Russian, Putin>, <China, Xi>, <USA, Trump>,...

- **Pattern embedding**

- Adapting word embedding technique

-



$$\frac{1}{3}[(v(Russian) - v(Putin)) + (v(China) - v(Xi)) + (v(USA) - v(Trump))]$$

An Intuitive Solution

- **Reliable patterns are semantically similar to the seed patterns**
 - Joint consider pattern constructing words and extractions
 - Eg., \$Person , president of \$Country
 - Constructing words: president, of
 - Extractions: <Russian, Putin>, <China, Xi>, <USA, Trump>,...
- **Pattern embedding**
 - Adapting word embedding technique
 -
 - Reliable patterns are those who are close to the seed patterns

Issue of the Intuitive Solution

- **Lack of supervision to determine an accurate boundary**
- **Solution**
 - Use the pattern embedding as features
 - Build a training set from the seed patterns
 - Positive patterns (highly reliable patterns)
 - Negative patterns (highly unreliable patterns)
 - Train a classifier

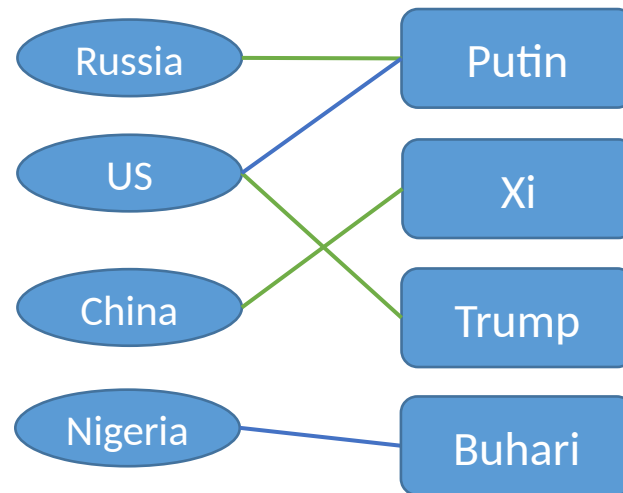
How to Detect Negative Samples

- **Challenge: open world assumption**

- Eg., the seed pattern does not extract <US, president, Putin> nor <Nigeria, president, Buhari>

- **Arity-constraint**

- Constraint on degrees of entities and values in an entity-value bipartite graph



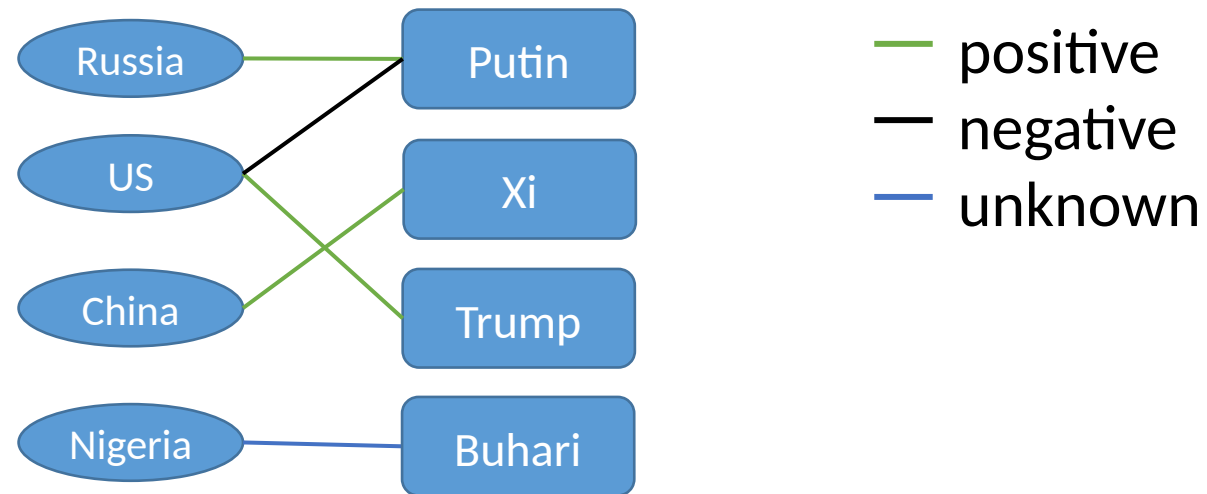
Arity-Constraint

- The arity-constraint is equivalent to setting constraints on the degree of entities and degree of values .
- **Hard arity-constraint:**
 - If the -Quantiles, we set it as hard arity-constraint
 - For hard arity-constraint, **no violation** is allowed; e.g., **#country of a president = 1**
- **Soft arity-constraint:**
 - If the -Quantiles, we set it as soft arity-constraint
 - For soft arity-constraint, **some violations** are allowed; e.g., **#president of a country**
 - If a tuple has a high reliability score, we can add it into the truth tuple set even it may violate the soft arity-constraint.

Arity-constraint-based Conflict Finding

- **Tuple's Polarity**

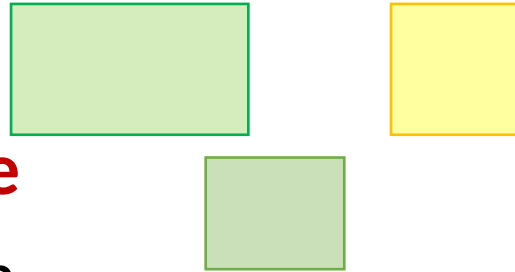
- A tuple t is **positive**, if $t \in T$ (i.e., the true tuple set);
- t is **negative**, if $t \notin T$, and adding t to T will cause violation of arity-constraints.
- t is **unknown**, if $t \notin T$ and t is not negative



Pattern Reliability

Pattern reliability score

- Extension of precision
 - Number of positive tuples
 - Number of unknown tuples
 - Total number of tuples
- Positive and negative patterns
 - Positive patterns:
 - Negative patterns:



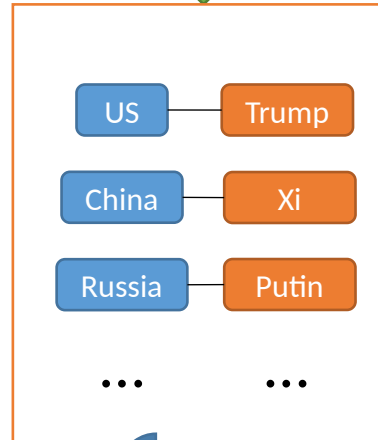
Tuple Reliability



Tuple reliability score

- Edge weight of the entity-value bipartite graph
 - Positive patterns' reliability score
 - Frequency
- Optimization problem: Find the bipartite graph with the maximal sum of edge weights under the arity-constraints
 - Hard arity-constraint: no violation allowed, penalty
 - Soft arity-constraint: violation allowed with a positive penalty

\$Country president
\$Person



\$PERSON, president of \$LOCATION +

...

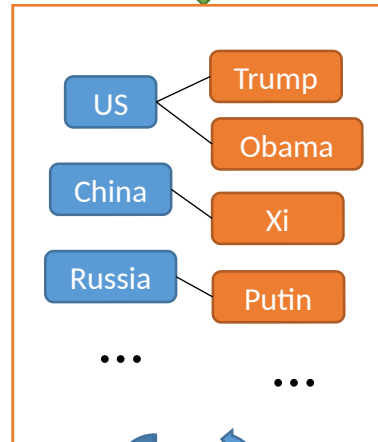
\$Person, daughter of \$Country 's president, -

...

\$PERSON, the elected \$LOCATION leader
\$LOCATION 's dictator \$PERSON
\$PERSON ruled \$LOCATION
President \$PERSON arrived in \$LOCATION
\$LOCATION : president \$PERSON

...

\$Country president
\$Person



\$PERSON, president of \$LOCATION +
\$PERSON, the \$LOCATION 's president +

...

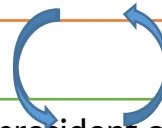
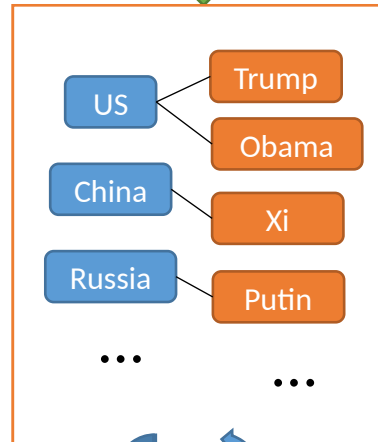
\$Person, daughter of \$Country 's president, -
\$LOCATION prime minister \$PERSON -
\$Person, son of \$Country 's president, -

...

\$PERSON, the elected \$LOCATION leader
\$LOCATION 's dictator \$PERSON
\$PERSON ruled \$LOCATION
President \$PERSON arrived in \$LOCATION
\$LOCATION : president \$PERSON

...

\$Country president
\$Person



\$PERSON, president of \$LOCATION +
\$PERSON, the \$LOCATION 's president +

...

\$Person, daughter of \$Country 's president, -
\$LOCATION prime minister \$PERSON -
\$Person, son of \$Country 's president, -

...

\$PERSON, the elected \$LOCATION leader +
\$LOCATION 's dictator \$PERSON +
\$PERSON ruled \$LOCATION +
President \$PERSON arrived in \$LOCATION -
\$LOCATION : president \$PERSON -

...

classification



References

- Hearst, Marti A. "Automatic acquisition of hyponyms from large text corpora." ACL 1992.
- GuoDong, Zhou, Su Jian, Zhang Jie, and Zhang Min. "Exploring various knowledge in relation extraction." ACL 2005.
- Shwartz, Vered, Yoav Goldberg, and Ido Dagan. "Improving hypernymy detection with an integrated path-based and distributional method." ACL 2016.
- Socher, Richard, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. "Semantic compositionality through recursive matrix-vector spaces." EMNLP 2012.
- Liu, Yang, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. "A dependency-based neural network for relation classification." ACL 2015.
- Xu, Yan, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. "Classifying relations via long short term memory networks along shortest dependency paths." EMNLP 2015.
- Nguyen, Dat Quoc, Kairit Sirts, Lizhen Qu, and Mark Johnson. "STransE: a novel embedding model of entities and relationships in knowledge bases." NAACL 2016.
- Qu, Meng, Xiang Ren, Yu Zhang, and Jiawei Han. "Weakly-supervised relation extraction by pattern-enhanced embedding learning." WWW 2018.

- Carlson, Andrew, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. "Toward an architecture for never-ending language learning." In AAAI 2010.
- Mintz, Mike, Steven Bills, Rion Snow, and Dan Jurafsky. "Distant supervision for relation extraction without labeled data." In ACL 2009.
- Qin, Pengda, Weiran Xu, and William Yang Wang. "DSGAN: generative adversarial training for distant supervision relation extraction." ACL 2018.
- Zeng, Daojian, Kang Liu, Yubo Chen, and Jun Zhao. "Distant supervision for relation extraction via piecewise convolutional neural networks." EMNLP 2015.
- Ratner, Alexander J., Christopher M. De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. "Data programming: Creating large training sets, quickly." NIPS 2016.
- Ratner, Alexander, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. "Snorkel: Rapid training data creation with weak supervision." VLDB 2017.
- Angeli, Gabor, Melvin Jose Johnson Premkumar, and Christopher D. Manning. "Leveraging linguistic structure for open domain information extraction." ACL 2015.
- Del Corro, Luciano, and Rainer Gemulla. "Clausie: clause-based open information extraction." WWW 2013.
- Schmitz, Michael, Robert Bart, Stephen Soderland, and Oren Etzioni. "Open language learning for information extraction." EMNLP 2012.
- Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. "Open information extraction from the web." *Ijcai* 2007.
- Fader, Anthony, Stephen Soderland, and Oren Etzioni. "Identifying relations for open information extraction." EMNLP 2011.