Generalized Names Recognition & Context Pattern Induction Method for NEE

PD Dr. Günter Neumann
DFKI and Saarland University

Bootstrapping Learning of Generalized Names

Yangarber, Lin, Grishman, Coling 2002 & Lin, Yangarber, Grishman, ICML 2003

- Much work on ML-NE focuses on classifying *proper names (PNs)*
  - Person/Location/Organization
- IE generally relies on domain-specific lexicon or *Generalized Names (GNs)*
  - Closer to terminology:
    single- or multi-word domain-specific expressions
- Automatic learning of GNs is an important first step towards truly adaptive IE
  - IE system that can automatically adapt itself to new domains

How GNs differ from PNs

- Not necessary capitalized
  - tuberculosis
  - E. coli
  - Ebola haemorrhagic fever
  - Variant Creutzfeldt-Jacob disease
- Name boundaries are non-trivial to identify
  - “the four latest typhoid fever cases”
- Set of possible candidate names is broader and more difficult to determine
  - “National Veterinary Services Director Dr. Gideon Bruckner said no cases of mad cow disease have been in South Africa.”
- Ambiguity
  - E. coli : organism or disease
  - Encephalitis : disease or symptom

NOMEN: the Learning Algorithm

1. <u>Input</u>: Seed names in several chosen categories
2. Tag occurrences of names
3. Generate local patterns around tags
4. Match patterns elsewhere in corpus

   Acquire top-scoring pattern(s)
5. Acquired patterns tags new names

   Acquire top-scoring name(s)
6. Repeat

Pre-processing

- Text-Zoner
  - Extract textual content
  - Strips of headers, footers etc.
- Tokenizer
  - Produces lemmas
- POS tagger
  - Statistically trained on WSJ
  - Unknown or foreign words are not lemmatized and tagged as noun

Seeds

- For each *target* category select N initial *trusted* seeds
    - Diseases:
        - Cholera, dengue, anthrax, BSE, rabies, JE, Japanese encephalitis, influenza, Nipah virus, FMD
    - Locations:
        - United States, Malaysia, Australia, Belgium, China, Europe, Taiwan, Hong Kong, Singapore, France
    - Others
        - Case, health, day, people, year, patient, death, number, report, farm
- Use frequency counts computed form corpus or some external data-base
- Many more additional categories can be defined

Positive vs. Negative Seeds

- A seed name serves as
  - a positive example for its own class, and
  - a negative example for all other classes.

- Negative examples help steer the learner away from unreliable patterns
  - Competing classes
  - Termination of bootstrapping learning

Pattern generation

- Tag every occurrence of each seed in corpus
  - "…new cases of <dis> cholera </dis> this year in …"
- For each tag, generate context rule: start/left-tag
  - [new case of <dis> cholera this year]
- Generalized left-side candidate patterns:
  - [new case of <dis> * * *]
  - [* case of <dis> * * *]
  - [* * of <dis> * * *]
  - [* * * <dis> cholera this year]
  - [* * * <dis> cholera this *]
  - [* * * <dis> cholera * *]

Pattern generation

- For each tag, generate context rule: end/right-tag
  - [case of cholera </dis> this year in]
- Generalized right-side candidate patterns:
  - [case of cholera </dis> * * *]
  - [* of cholera </dis> * * *]
  - [* * cholera </dis> * * *]
  - [* * * </dis> this year in]
  - [* * * </dis> this year *]
  - [* * * </dis> this * *]
- Note: all are potential patterns

Pattern application

- Apply each candidate pattern to corpus, observe where the pattern matches
  - E.g., the pattern [* * of <dis> * * *]
- Each pattern predicts one boundary: search for the partner boundary using a noun group NG regex:
  - [Adj* Noun+]

"…distributed the yellow fever vaccine to the people"

- The resulting NG can be (wrt. currently tagged corpus)
  - Positive: "…case <u>of</u> <dis> dengue </dis> …"
  - Negative: "…North <u>of</u> <loc> Malaysia </loc> …"
  - Unknown: "…symptoms <u>of</u> <?> swine fever </?> in …"

Identify candidate NGs

- Sets of NG that the pattern p matched
  - Pos = distinct matched NG types of correct category
  - Neg = distinct matched NG types of wrong category
  - Unk = distinct matched NGs of unknown category

Collect statistics for each pattern

$$acc(p) = \frac{|Pos|}{(|Pos| + |Neg|)}$$

$$conf(p) = \frac{|Pos|}{(|Pos| + |Neg| + |Unk|)}$$

Pattern selection

- Discard pattern p if acc(p) < θ
- The remaining patterns are ranked by
  - Score(p) = conf(p)*log|Pos(p)|
- Prefer patterns that:
  - Predict the correct category with less risk
  - Stronger support: match more distinct known names
- Choose top n patterns for each category
  - [* die of <dis> * * *]
  - [* vaccinate against <dis> * * *]
  - [* * * </dis> outbreak that have]
  - [* * * </dis> * * *]
  - [* case of <dis> * * *]

To get positive score, a pattern must have at least two distinct NGs as positive example, and more positive than negative exam.

Application phase: Name selection

- Apply each accepted pattern to corpus, to find candidate names (using the NG)
  - "More people die of <dis> profound heartbreak than grief."
- Rank each name *type* t based on quality of patterns that match it:

$$Rank(t) = 1 - \prod_{p \in M_t} (1 - conf(p))$$

M_t is the set of accepted patterns which match any of the instances of t

  - Require $|M_t| \geq 2 \Rightarrow$ t should appear $\geq$ 2 times
  - M_t contains at least on pattern predicting the left boundary of t and one pattern predicting the right boundary
  - Conf(p) assigns more credit to reliable patterns

Name selection

- Accept up to 5 top-ranked candidate names for each category
- Iterate learning algorithm until no more names can be learned
    - Bootstrap by using in each new iteration the extended set of new names to re-annotate the corpus

Salient Features of Nomen

- Generalized names
- A few manually-selected seeds
- Un-annotated corpus
- Un-restricted context (no syntactic restrictions)
- Patterns for left and right contexts independently
- Multiple categories simultaneously

Experiments

- Construction of reference lists for judging recall & precision of NOMEN

Compiled from multiple sources (medical DB, Web, manual review)

Appearing two or more time in development corpus

Manual list + acronyms + strip generic heads

Reference List	Disease	Location
Manual	2492	1785
Recall (26K) Recall (100K)	322 616	641 1134
Precision	3588	2404

Score recal against recall list and precision against precision list; Distinguish type and token tests

Results

- Final recall & precision for 8 categories
  - Around 70% (in case of type-based evaluation)
  - Classical PN: Recall: 86-92%, Precision: above 70%
- Multi-class learning has positive effects
  - A category is less likely to expand beyond its true territory
  - The accepted names in each category serve as negative example for all categories
  - The learners avoid acquiring patterns with too many negatives
  - In some sense, the categories *self-tune* each other
- Comparison with human-in-the-loop
  - “More groups” can be as good as “few groups + human reviewer”
- Using a negative category (noun groups that belong to neither category, but generic terms), then also substantial increase in performance

Context Pattern Induction Method for NEE

- Starting with a few seed entities, it is possible to induce high-precision context patterns by exploiting entity context redundancy.
- New entity instances of the same category can be extracted from unlabeled data with the induced patterns to create high-precision extensions of the seed lists.
- Features derived from token membership in the extended lists improve the accuracy of learned named-entity taggers.

cf. Talukdar et al., CoNLL, 2006

➡ novel combination of grammatical induction and statistical techniques to create high-precision patterns.

➡ method is language independent

Overall induction method

1. Let E = seed set, T = text corpus.
2. Find the contexts C of entities in E in the corpus T.
3. Select trigger words from C.
4. For each trigger word, induce a pattern automaton.
5. Use induced patterns P to extract more entities E‘
6. Rank P and E‘
7. If needed, add high scoring entities in E‘ to E and return to step 2. Otherwise, terminate with patterns P and extended entity list E ∪ E‘ as results.

Extracting Context

- For each occurrence of the seed elements, extract a fixed number W (context window size) of tokens immediately preceding and immediately following the matched seed elements.
- C = all such extended matching sequences, where all seed elements are substituted by the generic token -ENT-

Example: extracted context for known gene names

increased expression of -ENT- *in vad mice*

the expression of -ENT- *mrna was greater*

expression of the -ENT- *gene in mouse*

Trigger Word Selection

- Observation:
  - some tokens are more specific to particular entity classes than others (e.g., the word *expression* in our example)
  - whenever we identify such a word in a text then the probability of finding an entity (of the corresponding entity class) in its vicinity is high
- such *staring* tokens are called trigger words, since they mark the beginning of a pattern

Condition on Trigger Words

- It is frequent in the set C of extracted contexts.
- It is specific to entities of interest and thereby to extracted contexts.

Potential Trigger Words

- For each context $c \in C$ compute the **dominating word** d_c

$$d_c = \arg\max_{w \in c} f_w$$

- where f_w is the **inverse document frequency** (IDF) of word w (N = |C| in our case; note, other term weighting functions are also possible):

$$f_w = \log\left(\frac{N}{n_w}\right)$$

- a dominating word can occur in many different contexts, thus order them in decreasing order and select the N top elements

Pattern Induction

- Merge all **contexts of a trigger word** in to a single automaton.
- Prune the automaton by removing transitions with weak evidence so as to increase match precision.

Initial Induction

- Change context elements
  - cut left (right) context segment s.t. it **starts with the trigger word** (called predictive left/right context)
  - for the other context segment only retain the token that immediately follows -ENT-

Example

expression of -ENT- *in*
expression of -ENT- *mrna*
expression of the -ENT- *gene*

Pattern

An automaton A is *k-reversible* iff (1) A is deterministic and (2) A^r is deterministic with k tokens of lookahead, where A^r is the automaton obtained by reversing the transitions of A. K-reversible grammars are close to power of general regular grammars, and it can be shown that they can be learned with positive examples only.

- Merge all updated context elements of a trigger word into **a single 1-reversible automaton**
    - no recursion, deterministic
- In the 1-reversible automaton induced for each trigger word, **all transitions labeled by a given token go to the same state**, which is **identified with that token**.
- Assign to each transition e(v,w) the probability P(w|v), where C(v,w) **number of occurrences of the bigram vw** in contexts for W.

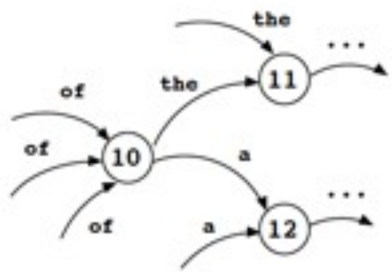

Figure 1: Fragment of a 1-reversible automaton

$$P(w|v) = \frac{C(v,w)}{\Sigma_{w'} C(v,w')}$$

Pruning

The simplest pruning method is to set a count threshold c below which transitions are removed. However, this is a poor method. Consider state 10 in the automaton of Figure 2, with c = 20. Transitions (10, 11) and (10, 12) will be pruned. C(10, 12) #c but C(10, 11) just falls short of c. However, from the transition counts, it looks like the sequence "the -ENT-" is very common. In such a case, it is not desirable to prune (10, 11). Using a local threshold may lead to overpruning.

- Goal: remove „useless“ transitions
- Simple threshold counter applied locally might perform poor
- Idea: Keep transitions that are used in relatively many probable paths through the automaton.
- Let P(p) the prob. of path p defined as the product of its transitions‘ probabilities.
  - Then the posterior probability of edge (v,w)
  - Then, remove all transitions that leave state v whose posterior probability is lower than p_v

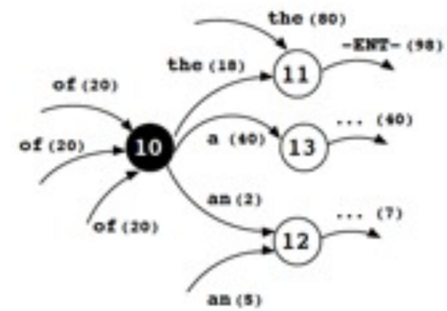

Figure 2: Automaton to be pruned at state 10. Transition counts are shown in parenthesis.

$$P(v,w) = \frac{\sum_{(v,w)\in p} P(p)}{\sum_p P(p)}$$

$$p_v = k(\max_w P(v,w))$$

Can be efficiently computed using forward-backward algorithm known from HMM.

Automata as Extractors

- Each automata represents a **high-precision pattern** that starts with a given trigger word
- **For each extracted segment**, decide whether to keep all tokens
  - distinguish between keep (K) or droppable (D) tokens
  - a token is droppable, if it belongs to a stop-word, non-capitalized or a number
- Label each token as K or D, and extract the longest token sequence matching „K [D K] * K
- Pair all patterns and their extracted entities

Example
Pattern:
„analyst at -ENT- and"
Text:
„He is an analyst at the University of California and ..."
Extracted segment:
„the University of California"
Labelled segment:
„the/D University/K of/D California/K"
Extracted entity:
„University of California"
Pair:
(„analyst at -ENT- and",
{„University of California"})

Filtering Patterns

- Pairs of patterns and extracted entities might vary in quality, so rank them
- Difficult: no negative examples
- Solution: with seeds of multiple classes, consider seed instances of one class as negative instances for the other classes, cf. Yangarber et al.
- Let pos(p) and neg(p) the number of positive and negative seeds extracted by pattern p
- Conservative strategy:
  - discard all patterns with positive neg(p) value
  - and pos(p) < $n_{patterns}$

Ranking Entities

- Let G be the set of all patterns that are retained after filtering
- Let I(e,p) = 1 if entity e has been extracted by pattern p, and 0 otherwise
- The score S(e) of an entity e is defined as
  - $S(e) = \sum_{p \in G} I(e,p)$
- Iteration of algorithm:
  - Use threshold n_{entity} for deciding which entity should be added to the initial seed sets
  - The call complete learning algorithm with newly expanded seed sets

Experiments

- Unlabeled data: 18 billion tokens (31 million documents) of news data
- Experimentation with trigger words: 500 and 1000
- Only left prediction context is considered and a single iteration
- Two seed lists:
  - CoNLL POL types
  - Watch Brand Names

CoNLL Experiments

- Person (PER), Organization (ORG), Location (Loc) from CoNLL data set as seed elements
- Evaluation: precision on 100 randomly selected instances from each of the extended list (no seeds)
- Result:

Category	Seed Size	Patterns Used	Extended Size	Precision
LOC	379	29	3001	70%
ORG	1597	276	33369	85%
PER	3616	265	86265	88%

Table 3: Results of LOC, ORG & PER entity list extension experiment with $\eta_{\text{pattern}} = 10$ set manually.

Examples of Learned Patterns

Induced LOC Patterns
troops in –ENT–to
Cup qualifier against –ENT–in
southern –ENT–town
war - torn –ENT–.
countries including –ENT–.
Bangladesh and –ENT–,
England in –ENT–in
west of –ENT–and
plane crashed in –ENT–.
Cup qualifier against –ENT–,

Extracted LOC Entities
US
United States
Japan
South Africa
China
Pakistan
France
Mexico
Israel
Pacific

Induced PER Patterns
compatriot –ENT–.
compatriot –ENT–in
Rep. –ENT–,
Actor –ENT–is
Sir –ENT–,
Actor –ENT–,
Tiger Woods , –ENT–and
movie starring –ENT–.
compatriot –ENT–and
movie starring –ENT–and

Extracted PER Entities
Tiger Woods
Andre Agassi
Lleyton Hewitt
Ernie Els
Serena Williams
Andy Roddick
Retief Goosen
Vijay Singh
Jennifer Capriati
Roger Federer

Induced ORG Patterns
analyst at –ENT–.
companies such as –ENT–.
analyst with –ENT–in
series against the –ENT–tonight
Today 's Schaeffer 's Option Activity Watch features –ENT–(
Cardinals and –ENT–,
sweep of the –ENT–with
joint venture with –ENT–(
rivals –ENT–Inc.
Friday night 's game against –ENT–.

Extracted ORG Entities
Boston Red Sox
St. Louis Cardinals
Chicago Cubs
Florida Marlins
Montreal Expos
San Francisco Giants
Red Sox
Cleveland Indians
Chicago White Sox
Atlanta Braves

Table 9: Top ranking LOC, PER, ORG induced pattern and extracted entity examples.

Watch Brand Names

Seeds

Corum, Longines, Lorus, Movado, Accutron, Audemars Piguet, Cartier, Chopard, Franck Muller, IWC, Jaeger-LeCoultre, A. Lange & Sohne, Patek Philippe, Rolex, Ulysse, Nardin, Vacheron Constantin

- 17 watch brand names as seeds
- Extended pattern filter: only retain patterns that contain token „watch“
- P_{100}=85.7%
- Remarks:
  - small seed set
  - no negative information
- Observation:
  - the unambiguous nature of seed instances is much more important than the size of the seed list
  - for relatively unambiguous categories, it is possible to successfully rank patterns with positive information only

Rolex	Fossil	Swatch
Cartier	Tag Heuer	Super Bowl
Swiss	Chanel	SPOT
Movado	Tiffany	Sekonda
Seiko	TechnoMarine	Rolexes
Gucci	Franck Muller	Harry Winston
Patek Philippe	Versace	Hampton Spirit
Piaget	Raymond Weil	Girard Perregaux
Omega	Guess	Frank Mueller
Citizen	Croton	David Yurman
Armani	Audemars Piguet	Chopard
DVD	DVDs	Chinese
Breitling	Montres Rolex	Armitron
Tourneau	CD	NFL

Expanded list

Additional Experiment: Can another NE tagger benefit from expanded seed list?

- Observation: supervised models usually outperform unsupervised model, but needs expensive training data
- Question: can a supervised NE tagger benefit from an automatically generated entity list s.t. it performs well for smaller training data?
- Starting point:
  CRF baseline tagger
  trained on full
  CoNLL-2003
  shared task training dat

System	F1 (Precision, Recall)
Florian et al. (2003), best single, no list	89.94 (91.37, 88.56)
Zhang and Johnson (2003), no list	90.26 (91.00, 89.53)
CRF baseline, no list	89.52 (90.39, 88.66)

Table 6: Baseline comparison on 4 categories (LOC, ORG, PER, MISC) on Test-a dataset.

Results

Training Data (Tokens)	Test-a			Test-b		
	No List	Seed List	Unsup. List	No List	Seed List	Unsup. List
9268	68.16	70.91	**72.82**	60.30	63.83	**65.56**
23385	78.36	79.21	**81.36**	71.44	72.16	**75.32**
46816	82.08	80.79	**83.84**	76.44	75.36	**79.64**
92921	85.34	83.03	**87.18**	81.32	78.56	**83.05**
203621	89.71	84.50	**91.01**	84.03	78.07	**85.70**

Table 7: CRF tagger F-measure on LOC, ORG, PER extraction.

Training Data (Tokens)	Test-a			Test-b		
	No List	Seed List	Unsup. List	No List	Seed List	Unsup. List
9229	68.27	70.93	**72.26**	61.03	64.52	**65.60**
204657	89.52	84.30	**90.48**	83.17	77.20	**84.52**

Table 8: CRF tagger F-measure on LOC, ORG, PER and MISC extraction.