Processing of large document collections

Part 9 (Information extraction: learning extraction patterns)

Helena Ahonen-Myka

Spring 2006

Learning of extraction patterns

- motivation: portability of IE systems
- learning methods
  - AutoSlog
  - AutoSlog-TS
  - Multi-level bootstrapping

Portability of information extraction systems

- one of the barriers to making IE a practical technology is the cost of adapting an extraction system to a new scenario
- in general, each application of extraction will involve a different scenario
- implementing a scenario should not require too much time and not the skills of the extraction system designers

Portability of information extraction systems

- the basic question in developing a customization tool is the form and level of the information to be obtained from the user
- goal: the customization is performed directly by the user (rather than by an expert system developer)

Portability of information extraction systems

- if we are using a pattern matching system, most work will probably be focused on the development of the set of patterns
- also changes
  - to the dictionaries
  - to the semantic hierarchy
  - to the set of inference rules
  - to the rules for creating the output templates

Portability of information extraction systems

- we cannot expect the user to have experience with writing patterns (regular expressions with associated actions) and familiarity with formal syntactic structure
- one possibility is to provide a graphical representation of the patterns but still too many details of the patterns are shown
- possible solution: learning from examples

Learning from examples

- learning of patterns
  - information is obtained from examples of sentences of interest and the information to be extracted
- for instance, in a system ”AutoSlog” patterns are created semiautomatically from the templates of the training corpus

AutoSlog

- Ellen Riloff, University of Massachusetts
  - Automatically constructing a dictionary for information extraction tasks, 1993
- idea:
  - given a template slot which is filled with words from the text (e.g. a name), the program searches for these words in the text and hypothesizes a pattern based on the immediate context of these words
  - the patterns are presented to a system developer, who can accept or reject the pattern
  - if the training corpus is representative of the target texts, the patterns should work also with new texts

Domain-specific knowledge

- the UMASS/MUC4 system used 2 dictionaries
  - a part-of-speech lexicon: 5436 lexical definitions, including semantic features for domain-specific words
  - a dictionary of 389 extraction patterns (= concept node definitions)
- for MUC4, the set of extraction patterns was manually constructed by 2 graduate students: 1500 person-hours

Two observations

- two central observations:
  - the most important facts about a news event are typically reported during the initial event description
    - the first reference to a targeted piece of information (e.g. a victim) is most likely where the relationship between that information and the event is made explicit

Two observations

- the immediate linguistic context surrounding the targeted information usually contains the words or phrases that describe its role in the event
  - e.g. ”A U.S. diplomat was kidnapped by FMLN guerillas”
  - the word ’kidnapped’ is the key word that relates the victim (A U.S. diplomat) and the perpetrator (FMLN guerillas) to the kidnapping event
  - ’kidnapped’ is the triggering word

Algorithm

- given a set of training texts and their associated answer keys
  - AutoSlog proposes a set of patterns that are capable of extracting the information in the answer keys from the texts
- given a string from an answer key template (= targeted string)
  - AutoSlog finds the first sentence in the text that contains the string
  - the sentence is given to a syntactic analysis component which generates an analysis of the sentence
  - using the analysis, AutoSlog identifies the first clause in the sentence that contains the string

Algorithm

- a set of heuristic rules are applied to the clause to suggest a good triggering word for an extraction pattern
- if none of the heuristic rules is satisfied then AutoSlog searches for the next sentence in the text and process is repeated

13

Heuristic rules

- each heuristic rule looks for a specific linguistic pattern in the clause surrounding the targeted string
- if a heuristic identifies its linguistic pattern in the clause then it generates
  - a triggering word
  - a set of enabling conditions

14

Heuristic rules

- suppose
  - the clause ”the diplomat was kidnapped”
  - the targeted string ”the diplomat”
- the targeted string appears as the subject and is followed by a passive verb ’kidnapped’
- a heuristic that recognizes the linguistic pattern **<subject> passive-verb** is satisfied
  - returns the word ’kidnapped’ as the triggering word, and
  - as enabling condition: a passive construction

15

Heuristic rule / extraction pattern

• <subj> passive-verb	• <victim> was murdered
• <subj> active-verb	• <perpetrator> bombed
• <subj> verb infinitive	• <perpetrator> attempted to kill
• <subj> aux noun	• <victim> was victim
• passive-verb <dobj>	• killed <victim>
• active-verb <dobj>	• bombed <target>
• infinitive <dobj>	• to kill <victim>

16

Heuristic rule / extraction pattern

• verb infinitive <dobj>	• threatened to attack <target>
• gerund <dobj>	• killing <victim>
• noun aux <dobj>	• fatality was <victim>
• noun prep <np>	• bomb against <target>
• active-verb prep <np>	• killed with <instrument>
• passive-verb prep <np>	• was aimed at <target>

17

Building extraction patterns

- e.g. <victim> was kidnapped
- triggering word (’kidnapped’) and enabling conditions (verb in passive) as above
- a slot to extract the information
  - the name of the slot comes from the answer key template
    - ”the diplomat” is Victim -> slot: Victim
  - the syntactic constituent comes from the linguistic pattern, e.g. the filler is the subject of the clause
    - ”the diplomat” is subject -> slot: Victim *Subject*

18

Building extraction patterns

- hard and soft constraints for the slot
  - e.g. constraints to specify a legitimate victim ('human',…)
- a type
  - e.g. the type of the event (bombing, kidnapping) from the answer key template

19

Example

…, public buildings were bombed and a car-bomb was…

Filler of the slot 'Phys_Target' in the answer key template:"public buildings"

Pattern (concept node definition):

Name: target-subject-passive-verb-bombed
Trigger: bombed
Slot: Phys_Target *Subject*
Slot-constraints: class phys-target *Subject*
Constant-slots: type bombing
Enabled-by: passive

20

A bad pattern

"they took 2-year-old gilberto molasco, son of patricio rodriguez, .."

Pattern (concept node definition):

Name: victim-active-verb-dobj-took
Trigger: took
Slot: victim *DirectObject*
Slot-constraints: class victim *DirectObject*
Constant-slots: type kidnapping
Enabled-by: active

21

A bad pattern

- a pattern is triggered by the word "took" as an active verb
- this pattern is appropriate for this sentence, but in general we don't want to generate a kidnapping node every time we see the word "took"

22

Bad patterns

- AutoSlog generates bad patterns for many reasons
  - a sentence contains the targeted string but does not describe the event
  - a heuristic proposes a wrong triggering word
  - syntactic analysis works incorrectly
- solution: human-in-the-loop

23

Empirical results

- training data: 1500 texts (MUC-4) and their associated answer keys
  - 6 slots were chosen
  - 1258 answer keys contained 4780 string fillers
- result:
  - 1237 extraction patterns

24

Empirical results

- human-in-the-loop:
  - 450 definitions were kept
  - time spent: 5 hours (compare: 1500 hours for a hand-crafted set of patterns)
- the resulting set of extraction patterns was compared with a hand-crafted set within the UMass/MUC-4 system
  - precision, recall, F-measure almost the same

25

AutoSlog-TS

- Riloff (University of Utah): Automatically generating extraction patterns from untagged text, 1996

26

Extracting patterns from untagged text

- AutoSlog needs manually tagged or annotated information to be able to extract patterns
- manual annotation is expensive, particularly for domain-specific applications like IE
  - may also need skilled people
  - ~8 hours to annotate 160 texts (AutoSlog)

27

AutoSlog-TS

- needs only a preclassified corpus of relevant and irrelevant texts
  - much easier to generate
  - relevant texts are available online for many applications
- generates an extraction pattern for every noun phrase in the training corpus
- the patterns are evaluated by processing the corpus and generating relevance statistics for each pattern

28

Process

- Stage 1:
  - the sentence analyzer produces a syntactic analysis for each sentence and identifies the noun phrases
  - for each noun phrase, the heuristic (AutoSlog) rules generate a pattern (a concept node) to extract the noun phrase
    - if more than one rule matches the context, multiple extraction patterns are generated
    - <subj> bombed, <subj> bombed embassy

29

Process

- Stage 2:
  - the training corpus is processed a second time using the new extraction patterns
  - the sentence analyzer activates (and counts) all patterns that are applicable in each sentence
  - relevance statistics are computed for each pattern
  - the patterns are ranked in order of importance to the domain

30

Relevance statistics

- relevance rate: $R_i = F_i / N_i$
  - F_i : the number of instances of pattern i that were activated in the relevant texts
  - N_i: the total number of instances of pattern i in the training corpus
- domain-specific expressions appear substantially more often in relevant texts than in irrelevant texts

31

Ranking of patterns

- the extraction patterns are ranked according to the formula:
  - $score_i = R_i * \log (F_i)$
  - or zero, if $R_i < 0.5$
    - in this case, the pattern is negatively correlated with the domain (assuming the corpus is 50% relevant)
- the formula promotes patterns that are
  - highly relevant or highly frequent

32

The top 25 extraction patterns (MUC-4)

- <subj> exploded
- murder of <np>
- assassination of <np>
- <subj> was killed
- <subj> was kidnapped
- attack on <np>
- <subj> was injured
- exploded in <np>

33

The top 25 extraction patterns, continues

- death of <np>
- <subj> took place
- caused <dobj>
- claimed <dobj>
- <subj> was wounded
- <subj> occurred
- <subj> was located
- took_place on <np>

34

The top 25 extraction patterns, continues

- responsibility for <np>
- occurred on <np>
- was wounded in <np>
- destroyed <dobj>
- <subj> was murdered
- one of <np>
- <subj> kidnapped
- exploded on <np>
- <subj> died

35

Human-in-the-loop

- the ranked extraction patterns were presented to a user for manual review
- the user had to
  - decide whether a pattern should be accepted or rejected
  - label the accepted patterns
    - murder of <np> -> <np> means the victim

36

AutoSlog-TS: conclusion

- empirical results comparable to AutoSlog
  - recall slightly worse, precision better
- the user needs to
  - provide sample texts (relevant and irrelevant)
  - spend some time filtering and labeling the resulting extraction patterns

37

Multi-level bootstrapping

- Riloff (Utah), Jones(CMU): Learning Dictionaries for Information Extraction by Multi-level Bootstrapping, 1999

38

Multi-level bootstrapping

- an algorithm that generates simultaneously
  - a semantic lexicon for several categories
  - extraction patterns for lexicon entries in each category
- input: unannotated training texts and a few seed words for each category of interest (e.g. location)

39

Mutual bootstrapping

- observation:
  - extraction patterns can generate new examples of a semantic category
  - new examples in turn can be used to identify new extraction patterns

40

Mutual bootstrapping

- process begins with a text corpus and a few predefined seed words for a semantic category
  - text corpus: e.g. terrorist events texts, web pages
  - semantic category : (e.g.) location, weapon, company

41

Mutual bootstrapping

- AutoSlog is used in an exhaustive manner to generate extraction patterns for every noun phrase in the corpus
- the extraction patterns are applied to the corpus and the extractions are recorded
  - for each pattern it is recorded which NPs it extracted

42

Mutual bootstrapping

- input for the next stage:
  - a set of extraction patterns, and for each pattern, the NPs it can extract from the training corpus
  - this set can be reduced by pruning the patterns that extract one NP only
    - general (enough) linguistic expressions are preferred

43

Mutual bootstrapping

- using the data, the extraction pattern is identified that is most useful for extracting known category members
  - known category members in the beginning = the seed words
  - e.g. in the example, 10 seed words were used for the location category (in terrorist texts): bolivia, city, colombia, district, guatemala, honduras, neighborhood, nicaragua, region, town

44

Mutual bootstrapping

- the best extraction pattern found is then used to propose new NPs that belong to the category (= should be added to the semantic lexicon)
- in the following algorithm:
  - SemLex = semantic lexicon for the category
  - Cat_EPlist = the extraction patterns chosen for the category so far

45

Algorithm

- Generate all candidate extraction patterns from the training corpus using AutoSlog
- Apply the candidate extraction patterns to the training corpus and save the patterns with their extractions to EPdata
- SemLex = {seed_words}
- Cat_EPlist = {}

46

Algorithm, continues

- Mutual Bootstrapping Loop
  - 1. Score all extraction patterns in EPdata
  - 2. best_EP = the highest scoring extraction pattern not already in Cat_EPlist
  - 3. Add best_EP to Cat_EPlist
  - 4. Add best_EP's extractions to SemLex
  - 5. Go to step 1

47

Mutual bootstrapping

- at each iteration, the algorithm saves the best extraction pattern for the category to Cat_EPlist
- all of the extractions of this pattern are assumed to be category members and are added to the semantic lexicon

48

Mutual bootstrapping

- in the next iteration, the best pattern that is not already in Cat_EPlist is identified
  - based on both the original seed words + the new words that have been added to the lexicon
- the process repeats until some end condition is reached

49

Scoring

- based on how many different lexicon entries a pattern extracts
- the metric rewards generality
  - a pattern that extracts a variety of category members will be scored higher than a pattern that extracts only one or two different category members, no matter how often

50

Scoring

- head phrase matching:
  - X matches Y if X is the rightmost substring of Y
  - ”New Zealand” matches ”eastern New Zealand” and ”the modern day New Zealand”
  - … but not ”the New Zealand coast” or ”Zealand”
  - important for generality
- each NP was stripped of leading articles, common modifiers (”his”, ”other”,…) and numbers before being saved to the lexicon

51

Scoring

- the same metric was used as in AutoSlog-TS
  - score(pattern$_i$) = R_i * log(F_i)
- F_i: the number of unique lexicon entries among the extractions produced by pattern i
- N_i: the total number of unique NPs that pattern i extracted
- $R_i = F_i / N_i$

52

Example

- 10 seed words were used for the location category (terrorist texts):
  - bolivia, city, colombia, district, guatemala, honduras, neighborhood, nicaragua, region, town
- the first five iterations...

53

Example

Best pattern	”headquartered in <x> (F=3, N=4)
Known locations	nicaragua
New locations	san miguel, chapare region, san miguel city
Best pattern	”gripped <x>” (F=2, N=2)
Known locations	colombia, guatemala
New locations	none

54

Example

Best pattern ”downed in <x>” (F=4, N=6)
Known locations nicaragua, san miguel*, city
New locations area, usulutan region, soyapango

Best pattern ”to occupy <x>” (F=4, N=6)
Known locations nicaragua, town
New locations small country, this northern area, san sebastian neighborhood, private property

55

Example

Best pattern ”shot in <x>” (F=5, N=12)
Known locations city, soyapango*
New locations jauja, central square, head, clash, back, central mountain region, air, villa el_salvador district, northwestern guatemala, left side

56

Strengths and weaknesses

- the extraction patterns have identified several new location phrases
  - jauja, san miguel, soyapango, this northern area
- but several non-location phrases have also been generated
  - private property, head, clash, back, air, left side
  - most mistakes due to ”shot in <x>”
- many of these patterns occur infrequently in the corpus

57

Multi-level bootstrapping

- the mutual bootstrapping algorithm works well but its performance can deteriorate rapidly when non-category words enter the semantic lexicon
- once an extraction pattern is chosen for the dictionary, all of its extractions are immediately added to the lexicon
  - few bad entries can quickly infect the dictionary

58

Multi-level bootstrapping

- for example, if a pattern extracts dates as well as locations, then the dates are added to the lexicon and subsequent patterns are rewarded for extracting these dates
- to make the algorithm more robust, a second level of bootstrapping is used

59

Multi-level bootstrapping

- the outer bootstrapping mechanism (”meta-bootstrapping”)
  - compiles the results from the inner (mutual) bootstrapping process
  - identifies the five most reliable lexicon entries
  - these five NPs are retained for the permanent semantic lexicon
  - the entire mutual bootstrapping process is then restarted from scratch (with new lexicon)

60

Multi-level bootstrapping

- number of iterations: 50 (for instance)
- output:
  – extraction patterns generated by the last iteration
    - extraction patterns from the previous iterations are thrown away

  – permanent semantic lexicon

61

Scoring for reliability

- to determine which NPs are most reliable, each NP is scored based on the number of different category patterns that extracted it (N_i):

$$score(NP_i) = \sum_{k=1}^{N_i} 1 + (.01 * score(pattern_k))$$

  – intuition: a NP extracted by e.g. three different category patterns is more likely to belong to the category than a NP extracted by only one pattern
  – additionally: a small factor to account for the strength of the patterns that extracted the NP

62

Multi-level bootstrapping

- the main advantage of meta-bootstrapping comes from re-evaluating the extraction patterns after each mutual bootstrapping process
- in practice, the ordering of patterns changes: more general patterns float to the top as the semantic lexicon grows

63

Multi-level bootstrapping: conclusion

- both a semantic lexicon and a dictionary of extraction patterns are acquired simultaneously
- resources needed:
  – corpus of (unannotated) training texts
  – a small set of words for a category
  – manual check of the lexicon entries (fast?)

64