Information extraction from text

Spring 2003, Part 4
Helena Ahonen-Myka

In this part

- active learning
- unsupervised learning
- multilingual IE
- closing of the course

Active learning

- the key bottleneck is obtaining the labeled training data
- the cost of labelling documents is usually considerably less than the cost of writing the wrapper's extraction rules by hand
- but: labeling documents can require domain expertise, and is tedious and error-prone
- active learning methods try to minimize the amount of training data required to achieve a a satisfactory level of generalization

Active learning

- basic idea:
  - start with a small amount of training data
  - run the learning algorithm
  - use the learned wrapper to predict which of the remaining unlabeled documents is most informative
    - informative – the example would help the learning algorithm generalize most
    - trivial example: if the set of examples contains duplicates, the learning algorithm should not suggest the user to annotate the same document twice

Active learning

- e.g. STALKER wrapper learning algorithm learns a sequence of landmarks for scanning from the beginning of the document to the start of the fragment to be extracted
- an alternative way to find the same position is to scan backwards from the end of the document for a different set of landmarks
- both learning tasks are solved in parallel on the available training data
- two resulting wrappers are then applied to all the unlabeled documents
- the system asks the user to label one of the documents for which the two wrappers give different answers

Active learning: other strategies

- compare
  - present for labeling the document that is textually least similar to the documents that have already been labeled
- unusual
  - choose the document with the most unusual proper nouns in it
- committee
  - invoke two different IE learning algorithms
  - select the document on which the learned rule sets most disagree

Unsupervised learning methods

- Brin: Extracting patterns and relations from the World Wide Web (DIPRE), 1998
- Yangarber et al: Automatic acquisition of domain knowledge for IE (ExDisco), 2000
- Crescenzi et al: ROADRUNNER: Towards automatic data extraction from large web sites, VLDB'2001

7

DIPRE

- in many cases we can obtain documents from multiple information sources, which will include descriptions of the same relation in different forms
- if several descriptions mention the same names as participants, there is a good chance that they are instances of the same relation

8

DIPRE

- e.g., finding (author, booktitle) pairs
  - 1. Start with a small seed set of (author, booktitle) pairs, like (Herman Melville, Moby Dick) , e.g. 5 pairs
  - 2. Find occurrences of all those books on the web
  - 3. From these occurrences, recognise patterns for the citations of books
  - 4. Search the web for these patterns and find new books
  - 5. Go to 2

9

DIPRE

- we can also start from a pattern
- suppose that we are seeking patterns corresponding to the relation HQ between a company C and the location L of its headquarters
- we are initially given one such pattern: "C, headquartered in L" => HQ(C,L)

10

DIPRE

- we can search for instances of this pattern in the corpus in order to collect pairs of invididuals in the relation HQ
  - for instance, "IBM, headquartered in Armonk" => HQ("IBM","Armonk")
- if we find other examples in the text which connect these pairs, e.g. "Armonk-based IBM", we might guess that the associated pattern "L-based C" is also indicator of HQ

11

DIPRE

- application
  - good for things that mean always the same (like authors and their books)?
  - does not work, e.g., with stock prices, because they are numbers?
  - the names etc. may not always have the same form (HP, Hewlett-Packard)

12

ExDisco

- Look for linguistic patterns which appear with a relatively high frequency in relevant documents
- the set of relevant documents is not known, they have to be found as part of the discovery process
  - one of the best indications of the relevance of the documents is the presence of good patterns -> circularity -> acquired in tandem

13

Preprocessing

- Name recognition marks all instances of names of people, companies, and locations -> replaced with the class name (C-Person, C-Company,...)
- a parser is used to extract all the clauses from each document
  - for each clause, a tuple is built, consisting of the basic syntactic constituents (subject, verb, object)
  - different clause structures (passive...) are normalized

14

Preprocessing

- Because tuples may not repeat with sufficient frequency, each tuple is reduced to a set of pairs, e.g.
  - verb-object
  - subject-object
- each pair is used as a generalized pattern
- once relevant pairs have been identified, they can be used to gather the set of words for the missing roles
  - e.g. verbs that occur with a relevant subject-object pair: "company {hire/fire/expel/...} person"

15

Discovery procedure

- Unsupervised procedure
  - the training corpus does not need to be annotated, not even classified
  - the user must provide a small set of seed patterns regarding the scenario
- starting with this seed, the system automatically performs a repeated, automatic expansion of the pattern set

16

Discovery procedure

- 1. The pattern set is used to divide the corpus U into a set of relevant documents, R, and a set of non-relevant documents U – R
  - a document is relevant, if it contains at least one instance of one of the patterns
- 2. Search for new candidate patterns:
  - automatically convert each document in the corpus into a set of candidate patterns, one for each clause
  - rank patterns by the degree to which their distribution is correlated with document relevance

17

Discovery procedure

- 3. Add the highest ranking pattern to the pattern set
  - optionally present the pattern to the user for review
- 4. Use the new pattern set to induce a new split of the corpus into relevant and non-relevant documents.
- 5. Repeat the procedure (from step 1) until some iteration limit is reached

18

Example

- Management succession scenario
- two initial seed patterns
  - C-Company C-Appoint C-Person
  - C-Person C-Resign
- C-Company, C-Person: semantic classes
- C-Appoint = {appoint, elect, promote, name, nominate}
- C-Resign = {resign, depart, quit}

19

ExDisco: conclusion

- Resources needed:
  - unannotated, unclassified corpus
  - a set of seed patterns
- produces complete, multi-slot patterns

20

ROADRUNNER

- the system does not rely on user-specified examples
  - also does not require any interaction with the user during the wrapper induction process
- the wrapper induction system does not know the structure (schema) of the page content
  - schema will be inferred during the induction process
  - system can handle arbitrarily nested structures

21

ROADRUNNER

- basic idea:
  - the system works with two HTML pages at a time
  - discovery is based on the study of similarities and dissimilarities between these pages
    - mismatches are used to identify relevant structures and content to be extracted

22

Algorithm

- HTML documents are first pre-processed by a lexical analyzer -> a list of tokens
  - each token is either an HTML tag or a string
- algorithm works on two objects at a time
  - sample
  - wrapper
- the wrapper is represented by a union-free regular expression (= a regular expression without union (or choice: |)

23

Algorithm

- the process starts with two documents
  - one is chosen to be an initial version of the wrapper
- the wrapper is progressively refined
  - the algorithm tries to find a common regular expression for the wrapper and the sample
    - by solving mismatches between the wrapper and the sample

24

Algorithm

- the sample is parsed using the wrapper
- a mismatch happens, when some token in the sample does not comply to the grammar specified by the wrapper
- when a mismatch is found, the algorithm tries to solve it by generalizing the wrapper
- the algorithm succeeds if a common wrapper can be generated by solving all mismatches

25

Mismatches

- two kinds of mismatches:
  - **string mismatches**: mismatches that happen when different strings occur in corresponding positions of the wrapper and sample
  - **tag mismatches**: mismatches between
    - different tags on the wrapper and sample, or
    - between one tag and one string

26

Mismatches

- string mismatches: discovering fields
  - If two pages belong to the same class, string mismatches may be due only to different values of a field
  - -> discover fields
  - to solve a string mismatch (e.g. 'John Smith' vs. 'Paul Jones'), the wrapper is generalized to mark the newly discovered field (e.g. 'John Smith' -> Text)
  - constant strings do not originate fields in the wrapper

27

Mismatches

- Tag mismatches: discovering optionals and iterators
  - strategy: first look for repeated patterns, then, if this attempt fails, try to identify an optional pattern
    - otherwise iterations may be missed
    - e.g. two books vs three books would be interpreted as "two books are required + an optional book is possible"

28

Discovering optionals

- Either on the wrapper or on the sample we have a piece of HTML code that is not present on the other side
  - by skipping this piece of code, we should be able to resume the parsing
- 1. optional pattern is located by cross-search
  - "look forward" and decide if the optional part is in the wrapper or in the sample
- 2. wrapper generalization
  - if <x y="..."/> is the optional pattern, a new pattern (<x y="..."/>)? is added to the wrapper

29

Discovering iterators

- e.g. one author has two books (wrapper) and the other has three books (sample)
- 1. repeated pattern is located by terminal-tag search
  - both the wrapper and the sample contain at least one occurrence of the repeated pattern
  - last token of the repeated pattern can be found immediately before the mismatch position
  - this token is called **terminal tag**
  - one of the mismatching tokens is the **initial tag** of the repeated pattern
    - the one which is followed by the terminal tag

30

Discovering iterators

- 2. repeated pattern matching
  - the candidate occurrence of the repeated pattern is matched against some upward portion of the sample
  - search succeeds, if we manage to find a match for the whole pattern
- 3. wrapper generalization
  - The contiguous repeated occurrences of the pattern s (around the mismatch region) are replaced by (s)+

31

Solving mismatches

- once a mismatch has been solved the parsing can be resumed
- if the parsing can completed (= it reaches the ends of the pages), we have generated a common wrapper for the two pages
- more complex cases:
  - **recursion:** new mismatches can be generated when trying to solve a mismatch
  - **backtracking:** algorithm may choose some alternative which does not lead to a correct solution -> choices can be backtracked and the next alternative tried

32

Evaluation

- cannot extract disjunctive structures
- as the names of the fields (attributes) are not known, the fields have to be named manually
  - also automatic post-processing might be possible?

33

Multilingual IE

- Assume we have documents in two languages (English/French), and the user requires templates to be filled in one of the languages (English) from documents in either language
  - "Gianluigi Ferrero a assisté à la réunion annuelle de Vercom Corp à Londres."
  - "Gianluigi Ferrero attended the annual meeting of Vercom Corp in London."

34

Both texts should produce the same template fill:

- <meeting-event-01> :=
  - organisation: 'Vercom Corp'
  - location: 'London'
  - type: 'annual meeting'
  - present: <person-01>
- <person-01> :=
  - name: 'Gianluigi Ferrero'
  - organisation: UNCLEAR

35

Multilingual IE: three ways of addressing the problem

- 1. solution
  - A full French-English machine translation (MT) system translates all the French texts to English
  - an English IE system then processes both the translated and the English texts to extract English template structures
  - the solution requires a separate full IE system for each target language (here: for English) and a full MT system for each language pair

36

Multilingual IE: three ways of addressing the problem

- 2. solution
  - Separate IE systems process the French and English texts, producing templates in the original source language
  - a 'mini' French-English MT system then translates the lexical items occurring in the French templates
  - the solution requires a separate full IE system for each language and a mini-MT system for each language pair

37

Multilingual IE: three ways of addressing the problem

- 3. solution
  - a general IE system, with separate French and English front ends
  - the IE system uses a language-independent domain model in which 'concepts' are related via bi-directional mappings to lexical items in multiple language-specific lexicons
  - this domain model is used to produce a language-independent representation of the input text – a discourse model

38

Multilingual IE: three ways of addressing the problem

- 3. solution continues…
  - the required information is extracted from the discourse model and the mappings from concepts to the English lexicon are used to produce templates with English lexical items
  - the solution requires a separate syntactic/semantic analyser for each language, and the construction of mappings between the domain model and a lexicon for each language

39

Multilingual IE

- Which parts of the IE process/systems are language-specific?
- Which parts of the IE process are domain-specific?

40

Closing

- What did we study:
  - stages of an IE process
  - learning domain-specific knowledge (extraction rules, semantic classes)
  - IE from (semi-)structured text

41

Closing

- exam: next week on Friday 28.3. at 16-20 (Auditorio)
  - alternative: in May
- last exercises: deadline on Tuesday 25.3.
- remember Course feedback / Kurssikysely!

42