Information extraction from text

Spring 2003, Part 2

Helena Ahonen-Myka

In this part

- 1. Some IE systems (sentence level phase)
  - FASTUS
  - CIRCUS
- 2. Learning of extraction rules
  - AutoSlog
  - AutoSlog-TS

1.1 FASTUS

- "Finite State Automaton Text Understanding System"
- SRI International (USA)
- MUC-4

FASTUS

- components:
  - dictionaries: part-of-speech for a word etc.
    - also inflected forms of the words
  - a set of domain patterns
  - a set of finite-state transducers

FASTUS: classification

- classification of documents into relevant and irrelevant
  - Is this document relevant?
    - For each sentence: is this sentence relevant?
    - if the document contains a relevant sentence, the document is (potentially) relevant
  - Is this sentence relevant?
    - A set of triggering words are selected from the domain patterns ("killed", "kidnapped", "dead"...)
  - Irrelevant sentences are removed

FASTUS: sentence analysis

- lexical analysis: for each word, pick up information from the dictionaries (is this a noun, verb...?)
- first set of finite transductors is used:
  - Name recognition (proper names, locations, etc.)
  - Noun group transductor (37 states)
  - Verb group transductor (18 states)

FASTUS: sentence analysis

- "A bomb was placed by a group of urban guerillas on the power tower."
  - a bomb (a-det bomb-noun): noun group
  - was placed: verb group
  - a group of urban guerillas: noun group
  - the power tower: noun group

7

FASTUS: domain pattern recognition

- a finite transducer is constructed for each pattern
  - state transitions are <head word, phrase type> pairs: bomb-nounGroup, placed-passiveVerbGroup
- pattern: bomb was placed by <Perpetrator> on <PhysicalTarget>
  - bomb-nounGroup placed-passiveVerbGroup by <Perpetrator> on <PhysicalTarget>
- would instantiate
  - Perpetrator = "a group of urban guerillas"
  - PhysicalTarget = "the power tower"

8

FASTUS

- In theory, many (most?) natural languages cannot be modelled using finite-state models (regular languages)
  - e.g. center embedding: "A mayor, who was kidnapped yesterday, was found dead today."
- In practice, arbitrarily deep structures do not exist -> finite-state models can be used
  - A mayor, who was kidnapped
  - A mayor was found dead today

9

FASTUS

- conceptually simple
- effective
- developed (originally) in three weeks

10

1.2 CIRCUS

- University of Massachusetts (USA)
- MUC-3 and MUC-4

11

Concept node definitions

- To extract information from text, CIRCUS relies on a domain-specific dictionary of **concept node definitions** (~domain patterns)
- Each concept node definition contains a set of slots to extract information from the surrounding context
  - e.g., slots for perpetrators, victims, …
  - each slot has
    - a syntactic expectation: where the filler is expected to be found in the linguistic context
    - a set of hard and soft constraints for its filler

12

Concept node definition for kidnapping verbs

- Concept node
  - name: $KIDNAP$
  - trigger word: kidnapped
  - slot-constraints:
    - class organization *Subject*
    - class terrorist *Subject*
    - class proper-name *Subject*
    - class human *Subject*
    - class human *DirectObject*
    - class proper-name *DirectObject*

13

Concept node definition for kidnapping verbs, cont.

- variable-slots:
  - Perpetrator *Subject*
  - Victim *DirectObject*
- constant-slots:
  - type kidnapping
- enabled-by:
  - active

14

Instantiated concept nodes

- each concept node definition has one or more **triggering words**
- given a sentence as input, CIRCUS
  - activates a concept node definition for each triggering word found in the sentence
  - generates a set of **instantiated concept nodes** as its output
- if multiple triggering words appear in sentence, then CIRCUS can generate multiple concept nodes for that sentence
- if no triggering words are found in the sentence, no output is generated

15

Instantiated concept nodes

- Given a sentence:
  - "Some guerillas kidnapped the diplomat."
- 'kidnapped' is found to be a triggering word for the concept node definition $kidnap$
- the following instantiated concept node is generated:
  - $kidnap$
    - Perpetrator: "some guerillas"
    - Victim: "the diplomat"

16

Knowledge needed for analysis

- for each word in the dictionary:
  - which parts-of-speech are associated with the word?
    - disambiguation routines to handle part-of-speech ambiguities
  - if the word is a triggering word: which concept node definition it triggers?
  - if the word is a noun or adjective, it has to be described in terms of one or more semantic features
    - e.g. for a noun: animate, human, terrorist
  - syntactic predictions: which words can follow?

17

Syntax processing in CIRCUS

- **stack**-oriented syntax analysis
- no parse tree is produced
- uses local syntactic knowledge to recognize noun phrases, prepositional phrases and verb phrases
- the constituents are stored in **global buffers** that track the subject, verb, direct object, indirect object and prepositional phrases of the sentence
  - *Subject*, *Verb*, *DirectObject*, ...

18

Syntax processing

- To process the sentence that begins
  - "John brought..."
- CIRCUS scans the sentence from left to right and
- uses syntactic predictions to assign words and phrases to syntactic constituents
- initially, the stack contains a single prediction: the hypothesis for a subject of a sentence

19

Syntax processing

- when CIRCUS sees the word "John", it
  - accesses its part-of-speech lexicon, finds that "John" is a proper noun
  - loads the standard set of syntactic predictions associated with proper nouns onto the stack
  - recognizes "John" as a noun phrase
  - because the presence of a NP satisfies the initial prediction for a subject, CIRCUS places "John" in the subject buffer (*Subject*) and pops the satisfied syntactic prediction from the stack

20

Syntax processing

- Next, CIRCUS processes the word "brought", finds that it is a verb, and assigns it to the verb buffer (*Verb*)
- in addition, the current stack contains the syntactic expectations associated with "brought": (the following constituent is...)
  - a direct object
  - a direct object followed by a "to" preposition phrase
  - a "to" preposition phrase followed by a direct object
  - an indirect object followed by a direct object

21

For instance,

- John brought a cake.
- John brought a cake to the party.
- John brought to the party a cake.
  - this is actually ungrammatical, but it has a meaning...
- John brought Mary a cake.

22

Syntactic expectations associated with "brought"

- 1. if NP is seen, NP is added to *DO*;
  - predict: if EndOfSentence, NIL -> *IO*
- 2. if NP, NP -> *DO*;
  - predict: if PP(to), PP -> *PP*, NIL -> *IO*
- 3. if PP(to), PP -> *PP*;
  - predict: if NP, NP -> *DO*
- 4. if NP, NP -> *IO*;
  - predict: if NP, NP -> *DO*

23

All alternatives are considered

- If the sentence continued: "John brought Mary"
  - "Mary" (NP) would be assigned to both *DirectObject* and *IndirectObject* buffers
- the syntactic expectations of (1) ,(2), and (3) above would be pushed to the stack
- depending on the words that follow "Mary", the contents of either *DirectObject* or *IndirectObject* are overwritten
  - "John brought Mary." ("Mary" = DO)
  - "John brought Mary to the party." ("Mary" = DO)
  - "John brought Mary a cake." ("Mary" = IO)

24

Filling template slots

- As soon as CIRCUS recognizes a syntactic constituent and places it in one of the global buffers, any active concept node that expects a slot filler from that buffer is examined
- the slot is filled if the constituent satisfies the slot's hard and soft semantic constraints
  - a hard constraint must be satisfied
  - a soft constraint defines a preference for a slot filler

25

Filling template slots

- "Some guerillas kidnapped the diplomat."
- analysis:
  - 1. "some guerillas" -> *Subject* buffer
  - 2. "kidnapped" -> triggers $kidnap$ concept node def
    - expects slot fillers from *Subject* and *DirectObject* buffers
  - 3. contents of *Subject* buffer -> Perpetrator
  - 4. "the diplomat" -> *DirectObject* buffer
  - 5. contents of *DirectObject* buffer -> Victim

26

Filling template slots

- A set of enabling conditions: describe the linguistic context in which the concept node should be triggered
  - $kidnap$ concept node should be triggered by "kidnap" only when the verb occurs in an active construction
  - a different concept node would be needed to handle a passive sentence construction

27

Hard and soft constraints

- soft constraints
  - Perpetrator should be an 'organization', 'terrorist', 'proper name', or 'human'
    - the dictionary may indicate that "guerilla" is a 'terrorist' or 'human'
  - Victim should be a 'human' or 'proper name'
    - "diplomat" is 'human'
- hard constraint
  - e.g. that some prepositional phrase filling a slot must begin with the preposition "to"

28

Filling template slots

- when a concept node satisfies certain instantiation criteria, it is freezed with its assigned slot fillers -> it becomes part of the semantic presentation of the sentence
- note: a concept node is not an entire answer template, just one part of it (representing information extracted from one clause)

29

Handling embedded clauses

- When sentences become more complicated, CIRCUS has to partition the stack processing in a way that recognizes embedded syntactic structures

30

Handling embedded clauses

- John asked Bill to eat the leftovers.
  - "Bill" is the subject of "eat"
- That's the gentleman that the woman invited to go to the show.
  - "gentleman" is the direct object of "invited" and the subject of "go"
- That's the gentleman that the woman declined to go to the show with.

31

Handling embedded clauses

- the stack of syntactic predictions is viewed as a single control kernel whose expectations change in response to specific lexical items as the analysis moves through the sentence
- when the analysis comes to a subordinate clause, the top-level kernel creates a subkernel that takes over to process the inferior clause -> a new parsing environment

32

Concept node classes

- Concept node definitions can be categorized into the following taxonomy of concept node types
  - verb-triggered (active, passive, active-or-passive)
  - noun-triggered
  - adjective-triggered
  - gerund-triggered
  - threat and attempt concept nodes

33

Active-verb triggered concept nodes

- A concept node triggered by a specific verb in an active voice
- typically a prediction for finding the Perpetrator in *Subject* and the Victim or PhysicalTarget in *DirectObject*
- for all verbs important to the domain
  - kidnap, kill, murder, bomb, detonate, massacre, ...

34

Concept node definition for kidnapping verbs

- Concept node
  - name: $KIDNAP$
  - slot-constraints:
    - class organization *Subject*
    - class terrorist *Subject*
    - class proper-name *Subject*
    - class human *Subject*
    - class human *DirectObject*
    - class proper-name *DirectObject*

35

Concept node definition for kidnapping verbs, cont.

- variable-slots
  - Perpetrator *Subject*
  - Victim *DirectObject*
- constant-slots:
  - type kidnapping
- enabled-by:
  - active
  - not in reduced-relative

36

Is the verb active?

- Function **active** tests
  - the verb is in past tense
  - any auxiliary preceding the verb is of the correct form (indicating active, not passive)
  - the verb is not in the infinitive form
  - the verb is not preceded by "being"
  - the sentence is not describing threat or attempt
  - no negation, no future

Passive verb-triggered concept nodes

- Almost every verb that has a concept node definition for its active form should also have a concept node definition for its passive form
- these typically predict for finding the Perpetrator in a by-*PrepPhrase* and the Victim or PhysicalTarget in *Subject*

Concept node definition for killing verbs in passive

- Concept node
  - name $KILL-PASS-1$
  - slot-constraints:
    - class organization *PrepPhrase*
    - class terrorist *PrepPhrase*
    - class proper-name *PrepPhrase*
    - class human *PrepPhrase*
    - class human *Subject*
    - class proper-name *Subject*

Concept node definition for killing verbs in passive

- variable-slots:
  - Perpetrator *PrepPhrase* is-preposition "by"?
  - Victim *Subject*
- constant-slots:
  - type murder
- enabled-by:
  - passive
  - subject is not "no one"

Fillers for several slots

- "Castellar was killed by ELN guerillas with a knife"
- a separate concept node for each PrepPhrase
- Concept node
  - name $KILL-PASS-2$
  - slot-constraints:
    - class human *Subject*
    - class proper-name *Subject*
    - class weapon *PrepPhrase*

Fillers for several slots

- variable-slots:
  - Instrument *PrepPhrase* is-preposition "by" and "with"?
  - Victim *Subject*
- constant-slots:
  - type murder
- enabled-by:
  - passive
  - subject is not "no one"

Noun-triggered concept nodes

- The following concept node definition is triggered by nouns
  - massacre, murder, death, murderer, assassination, killing, and burial
- looks for the Victim in an of-PrepPhrase

43

Concept node definition for murder nouns

- Concept node
  - name $MURDER$
  - slot-constraints:
    - class human *PrepPhrase*
    - class proper-name *PrepPhrase*
  - variable-slots:
    - Victim *PrepPhrase*, preposition "of" follows triggering word?
  - constant-slots: type murder
  - enabled-by: noun-triggered, not-threat

44

Adjective-triggered concept nodes

- Sometimes a verb is too general to make a good trigger
  - "Castellar was found dead."
- it may be easier to use an adjective to trigger a concept node and check for the presence of specific verbs (in EnabledBy)

45

Other concept nodes

- Gerund-triggered concept nodes
  - for important gerunds
    - killing, destroying, damaging,...
- Threat and attempt concept nodes
  - require enabling conditions that check both the specific event (e.g. murder, attack, kidnapping) and indications that the event is a threat or attempt
  - "The terrorists intended to storm the embassy."

46

CIRCUS

- shallow, local syntactic analysis is fast
- system was also effective: one of the best in MUC-3 and MUC-4
- manual construction of the dictionary of concept node definitions is a problem
  - for MUC-4, 2 graduate students worked 1500 hours
  - -> system is not portable

47

2. Learning of extraction rules

- IE systems depend on a domain-specific knowledge
  - acquiring and formulating the knowledge may require many person-hours of highly skilled people (usually both domain and the IE system expertize is needed)
  - the systems cannot be easily scaled up or ported to new domains
  - automating the dictionary construction is needed

48

Learning of extraction rules

- AutoSlog
- AutoSlog-TS

49

2.1 AutoSlog

- Ellen Riloff, University of Massachusetts
  - Automatically constructing a dictionary for information extraction tasks, 1993
- continues the work with CIRCUS

50

AutoSlog

- Automatically constructs a domain-specific dictionary for IE
- given a training corpus, AutoSlog proposes a set of dictionary entries that are capable of extracting the desired information from the training texts
- if the training corpus is representative of the target texts, the dictionary should work also with new texts

51

Concept node dictionary

- 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 concept node definitions
- For MUC4, the concept node dictionary was manually constructed by 2 graduate students: 1500 person-hours

52

AutoSlog

- Two central observations:
  - the most important facts about a news event are typically reported during the initial event description
    - the first reference to a major component of an event (e.g. a victim or perpetrator) usually occurs in a sentence that describes the event
    - the first reference to a targeted piece of information is most likely where the relationship between that information and the event is made explicit

53

AutoSlog

- 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

54

Algorithm

- Given a set of training texts and their associated answer keys, AutoSlog proposes a set of concept node definitions that are capable of extracting the information in the answer keys from the texts

55

Algorithm

- Given a string from an answer key template
  - AutoSlog finds the first sentence in the text that contains the string
  - the sentence is handed over to CIRCUS which generates a conceptual analysis of the sentence
  - using the analysis, AutoSlog identifies the first clause in the sentence that contains the string

56

Algorithm

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

57

Heuristic rules

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

58

Conceptual anchor point heuristics

- 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 pattern **<subject> passive-verb** is satisfied
  - returns the word 'kidnapped' as the triggering word, and
  - as enabling condition: a passive construction

59

Linguistic patterns

<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>

60

Linguistic patterns

- verb infinitive <dobj>
- gerund <dobj>
- noun aux <dobj>
- noun prep <np>
- active-verb prep <np>
- passive-verb prep <np>

- threatened to attack <target>
- killing <victim>
- fatality was <victim>
- bomb against <target>
- killed with <instrument>
- was aimed at <target>

61

Building concept node definitions

- a slot to extract the information
  - a name of the slot comes from the answer key template
    - "the diplomat" is Victim -> Variable-slot: Victim
  - the syntactic constituent from the linguistic pattern, e.g. the filler is the subject of the clause
    - "the diplomat" is subject -> Variable-slot: Victim *Subject*

62

Building concept node definitions

- hard and soft constraints for the slot
  - e.g. constraints to specify a legitimate victim
- a type
  - e.g. the type of the event (bombing, kidnapping) from the answer key template
  - uses domain-specific mapping from template slots to the concept node types
    - not always the same: a concept node is only a part of the representation

63

Example

…, <u>public buildings</u> were bombed and a car-bomb was…

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

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

64

A bad definition

"they took 2-year-old <u>gilberto molasco</u>, son of patricio rodriguez, .."

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

65

A bad definition

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

66

Bad definitions

- AutoSlog generates bad definitions for many reasons
  - a sentence contains the targeted string but does not describe the event
  - a heuristic proposes a wrong triggering word
  - CIRCUS analyzes the sentence incorrectly
- Solution: human-in-the-loop

67

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 concept node definitions

68

Empirical results

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

69

2.2 AutoSlog-TS

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

70

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)

71

Extracting patterns from untagged text

- The annotation task is complex
- e.g. for AutoSlog the user must annotate relevant noun phrases
  - What constitutes a relevant noun phrase?
  - Should modifiers be included or just a head noun?
  - All modifiers or just the relevant modifiers?
  - Determiners? Appositives?

72

Extracting patterns from untagged text

- The meaning of simple NP's may change substantially when a prepositional phrase is attached
  - "the Bank of Boston" vs. "river bank"
  - Which references to tag?
  - Should the user tag all references to a person?

73

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

74

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

75

Process

- Stage 2:
  - the training corpus is processed a second time using the new extraction patterns
  - the sentence analyzer activates 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

76

Relevance statistics

- relevance rate: Pr (relevant text | text contains pattern i) = rfreq_i / totfreq_i
  - rfreq_i : the number of instances of pattern i that were activated in the relevant texts
  - totfreq_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

77

Ranking of patterns

- The extraction patterns are ranked according to the formula:
  - relevance rate * log (frequency)
  - or zero, if relevance rate < 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

78

The top 25 extraction patterns

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

79

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>

80

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

81

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

82

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

83