Outline of today's lecture

Classification: from last time

Overview of Natural Language Generation

An extended example: cricket reports

Learning from existing text

Referring expressions

Lecture 10: Natural Language Generation

Classification in NLP

- Also sentiment classification, word sense disambiguation and many others. POS tagging (sequences).
- Feature sets vary in complexity and processing needed to obtain features. Statistical classifier allows some robustness to imperfect feature determination.
- Acquiring training data is expensive.
- Few hard rules for selecting a classifier: e.g., Naive Bayes often works even when independence assumption is clearly wrong (as with pronouns). Experimentation, e.g., with WEKA toolkit.

Lecture 10: Natural Language Generation

Generation

Generation from what?! (Yorick Wilks)

- Logical form: inverse of (deep) parsing.
  aka realisation (but realisation often from a syntax tree).
  Special case: in an MT system using semantic transfer.
- Formally-defined data: databases, knowledge bases, semantic web ontologies, etc.
- Semi-structured data: tables, graphs etc.
- Numerical data: e.g., weather reports.
- User input (plus other data sources) in assistive communication.

Generating from data often requires domain experts.

Regeneration: transforming text

- Text from partially ordered bag of words: statistical MT.
- Paraphrase
- Summarization (single- or multi- document)
- Wikipedia article construction from text fragments
- Text simplification

Also: mixed generation and regeneration systems, MT.

Components of a generation system

Content determination deciding what information to convey

Discourse structuring overall ordering, sub-headings etc

Aggregation deciding how to split information into sentence-sized chunks

Referring expression generation deciding when to use pronouns, which modifiers to use etc

Lexical choice which lexical items convey a given concept (or predicate choice)

Realization mapping from a meaning representation (or syntax tree) to a string (or speech)

Fluency ranking

Approaches to generation

- Classical (limited domain): hand-written rules for first five steps, grammar for realization, grammar small enough that no need for fluency ranking (or hand-written rules).
- Templates: most practical systems. Fixed text with slots, fixed rules for content determination.
- Statistical (limited domain): components as above, but use machine learning (supervised or non-supervised).
- Regeneration: symbolic or statistical or mixed.

Lecture 10: Natural Language Generation

Input: cricket scorecard

Result India won by 63 runs						
India innings (50 overs maximum)	R	M	B	4s	6s	SR
SC Ganguly run out (Silva/Sangakarra)	9	37	19	2	0	47.36
V Sehwag run out (Fernando)	39	61	40	6	0	97.50
D Mongia b Samaraweera	48	91	63	6	0	76.19
SR Tendulkar c Chandana b Vaas	113	141	102	12	1	110.78
. . .						
Extras (lb 6, w 12, nb 7) 25						
Total (all out; 50 overs; 223 mins) 304						

Output: match report

India beat Sri Lanka by 63 runs. Tendulkar made 113 off 102 balls with 12 fours and a six. . . .

Actual report:

The highlight of a meaningless match was a sublime innings from Tendulkar, . . . he drove with elan to make 113 off just 102 balls with 12 fours and a six.

Output: match report

India beat Sri Lanka by 63 runs. Tendulkar made 113 off 102 balls with 12 fours and a six. . . .

Actual report:

The highlight of a meaningless match was a sublime innings from Tendulkar, . . . he drove with elan to make 113 off just 102 balls with 12 fours and a six.

Representing the data

- Granularity: we need to be able to consider individual (minimal?) information chunks (cf factoids in summarisation).
- Abstraction: generalize over instances.
- Faithfulness to source versus closeness to natural language?
- Inferences over data (e.g., amalgamation of scores)?
- Formalism.

e.g., name(team1/player4, Tendulkar),
balls-faced(team1/player4, 102)

Content selection

There are thousands of factoids in each scorecard: we need to select the most important.

name(team1, India), total(team1, 304),
name(team2, Sri Lanka), result(win, team1, 63),
name(team1/player4, Tendulkar),
runs(team1/player4, 113),
balls-faced(team1/player4, 102),
fours(team1/player4, 12),
sixes(team1/player4, 1)

Discourse structure and (first stage) aggregation

Distribute data into sections and decide on overall ordering:

Title: name(team1, India), name(team2, Sri Lanka), result(win,team1,63)

First sentence: name(team1/player4, Tendulkar), runs(team1/player4, 113), fours(team1/player4, 12), sixes(team1/player4, 1), balls-faced(team1/player4, 102)

Reports often state the highlights and then describe events in chronological order.

Predicate choice (lexical selection)

Mapping rules from the initial scorecard predicates:

result(win,t1,n) $\mapsto$ *_beat_v(e,t1,t2), _by_p(e,r), _run_n(r), card(r,n)*

name(t,C) $\mapsto$ *named(t,C)*

This gives:

name(team1, India), name(team2, Sri Lanka), result(win,team1,63) $\mapsto$
named(t1,'India'), named(t2, 'Sri Lanka'), _beat_v(e,t1,t2), _by_p(e,r), _run_n(r), card(r,'63')

Realistic systems would have multiple mapping rules.
This process may require refinement of aggregation.

Generating referring expressions

named(t1p4, 'Tendulkar'), _made_v(e,t1p4,r), card(r,'113'), run(r), _off_p(e,b), ball(b), card(b,'102'), _with_(e,f), card(f,'12'), _four_n(f), _with_(e,s), card(s,'1'), _six_n(s)
$\rightarrow$ *Tendulkar made 113 runs off 102 balls with 12 fours with 1 six.*

This is not grammatical. So convert:

with(e,f), card(f,'12'), _four_n(f), _with_(e,s), card(s,'1'), _six_n(s)

into:

with(e,c), _and(c,f,s), card(f,'12'), _four_n(f), card(s,'1'), _six_n(s)

Also: '113 runs' to '113'

Realisation

Produce grammatical strings in ranked order:

Tendulkar made 113 off 102 balls with 12 fours and one six.
Tendulkar made 113 with 12 fours and one six off 102 balls.
. . .
113 off 102 balls was made by Tendulkar with 12 fours and one six.

Lecture 10: Natural Language Generation

Learning from aligned scorecards and reports

Result India won by 63 runs						
India innings (50 overs maximum)	R	M	B	4s	6s	SR
SC Ganguly run out (Silva/Sangakarra)	9	37	19	2	0	47.36
V Sehwag run out (Fernando)	39	61	40	6	0	97.50
D Mongia b Samaraweera	48	91	63	6	0	76.19
SR Tendulkar c Chandana b Vaas	113	141	102	12	1	110.78
. . .						
Extras (lb 6, w 12, nb 7) 25						
Total (all out; 50 overs; 223 mins) 304						

The highlight of a meaningless match was a sublime innings from Tendulkar, . . . he drove with elan to make 113 off just 102 balls with 12 fours and a six.

Learning from aligned scorecards and reports

Annotate reports with corresponding data structures:

> *The highlight of a meaningless match was a sublime innings from Tendulkar (team1 player4), . . . and this time he drove with elan to make 113 (team1 player4 R) off just 102 (team1 player4 B) balls with 12 (team1 player4 4s) fours and a (team1 player4 6s) six.*

Learning from aligned scorecards and reports

Produce a dataset with large numbers of annotated reports:

- Either: annotate training set by hand (boring!)
- Or: write rules to create training set automatically, using numbers and proper names as links. (Parse the reports?)

Statistical content selection and discourse structuring

Content selection:

- Treat as a classification problem: derive all possible factoids from the data source and decide whether each is in or out, based on training data. Kelly et al (2009) using cricket data.
- Categorise factoids into classes, group factoids.
- Problem: avoiding 'meaningless' factoids, e.g. player names with no additional information about their performance.

Discourse structuring: generalising over reports to see where particular information types are presented (cf Wikipedia article generation).

Lecture 10: Natural Language Generation

Referring expressions

Given some information about an entity, how do we choose to refer to it?

- Pronouns/proper names/definite expressions etc (generate and test using anaphora resolution).
- Ellipsis and coordination (as in cricket example)
- Attribute selection: need to include enough modifiers to distinguish the expression from possible distractors.
  e.g., *the dog*, *the big dog*, *the big dog in the basket*.

Entities and referring expressions

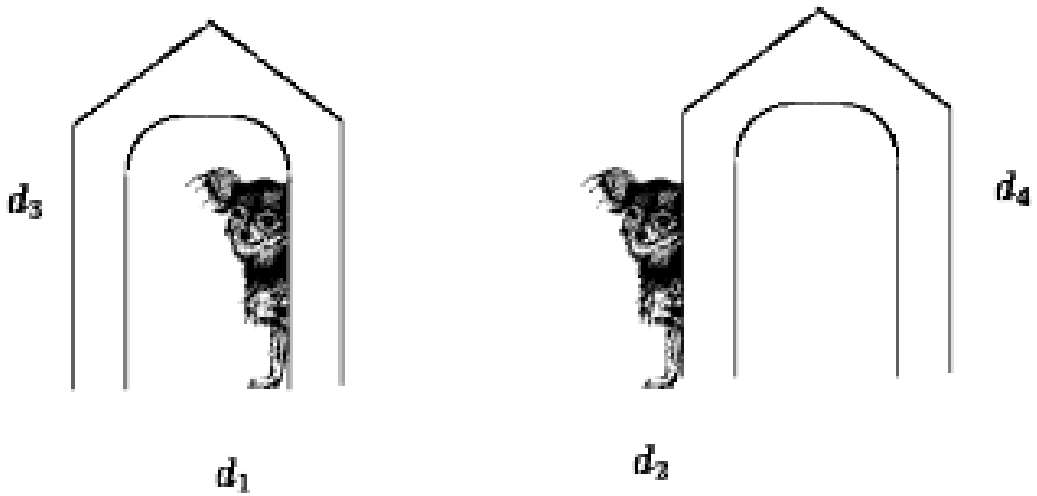

Figure 2
Another scene: Two dogs and two doghouses (from Krahmer and Theune [2002]).

A meta-algorithm for generating referring expressions

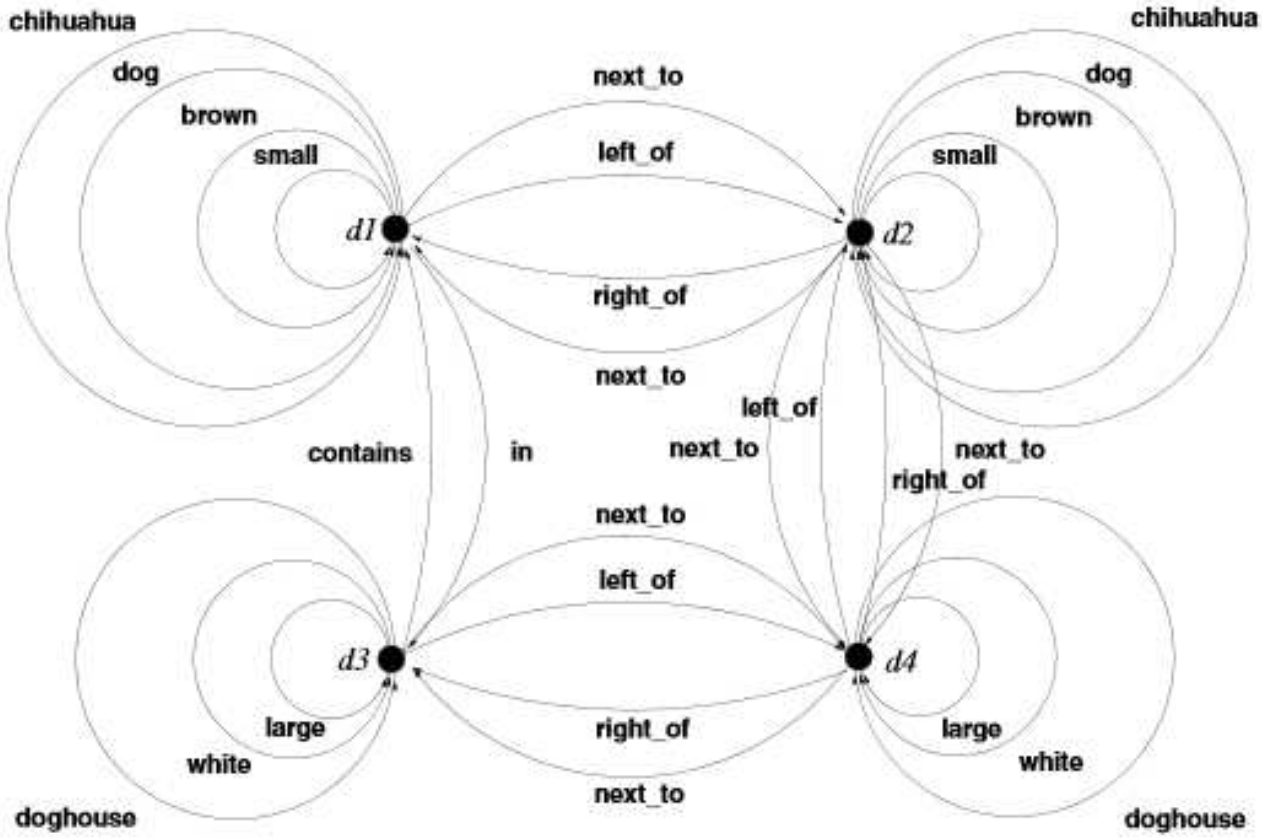

Figure 3
A graph representation of the scene in Figure 2.

A meta-algorithm for generating referring expressions

- Predicates in the KB are arcs on a graph, with nodes corresponding to entities.
- A description is a graph with unlabelled nodes: it matches the KB graph if it can be 'placed over' it (subgraph isomorphism).
- A distinguishing graph is one that refers to only one entity (i.e., it can only be placed over the KB graph in one way).
- If description refers to entities other than the one we want, the others are distractors.
- Aim: lowest cost distinguishing graph.

Algorithm

1. Start from node we want to describe (e.g., d2)
2. Expand graph by adding adjacent edges.
3. Cost function associated with each edge: e.g., full brevity — edge cost is 1.
4. Explore search space, only retaining graphs cheaper than best solution.
5. n^K where K is upper bound on number of edges.

Some issues

- Humans often use redundant expressions.
- Verbosity may be politer, easier to understand, convey emphasis etc
- Require knowledge of syntax: not just predicates. e.g., *earlier* and *before*.
- Limited domain: corpus-based method may be preferable.