

# Outline of today's lecture

Overview of Natural Language Generation

Components of Natural Language Generation systems

Referring expressions

Summarization

## Overview of Natural Language Generation

Components of Natural Language Generation systems

Referring expressions

Summarization

# Generation

## Generation from what?! (Yorick Wilks)

- ▶ Logical form: inverse of (deep) parsing.  
aka **realisation** (but realisation often from a syntax tree).
- ▶ 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.

## Example: Feedback on bumblebee identification

- ▶ Citizen scientists send in photos of bumblebees with their attempted identification (based on web interface): expert decides on actual species.
- ▶ Problem: expert has insufficient time to explain the errors.
- ▶ NLG system input: location data, attempted identification, expert identification, features of both species.
- ▶ NLG system output: coherent text explaining error or confirming identification and giving additional information.
  - ▶ Better identification training.
  - ▶ Expansion from 200 records a year to over 600 a month.

Blake et al (2012)

[homepages.abdn.ac.uk/advaith/pages/Coling2012.pdf](http://homepages.abdn.ac.uk/advaith/pages/Coling2012.pdf)

[Home](#) [Upload Photos](#) [Give us feedback](#) [Logout \(test\)](#)

Identify the bumblebee species in your photos. [How can I do it?](#) Use:  Common name  Scientific name

If no photo is displayed, please refresh the page (press F5)

Thorax:   
 Abdomen:   
 Antennae:   
 Face:   
 Wings:   
 Pollen basket:   
 Tail:

**22 matches**

Use the scrollbar to see all 22 species of bumblebees

**Common bumblebees**

**Less common bumblebees**

**Rare bum**

A montane species, found on moorland, in open forest and also visiting lowland meadows. Feeds predominantly on bilberry, heaths, ling and white clover.

**Queen/Worker:** Easily recognised by the extensive orange-red on the abdomen. Two yellow bands on the thorax.

**Male:** Same as female.

**Very rare**

**Similar species:**

Red-tailed bumblebee and Heath bumblebee. None of these has the orange tail extending over anything like as much of the abdomen.

## Example: Feedback on bumblebee identification

Our expert identified the bee as a Heath bumblebee rather than a Broken-belted bumblebee. . . . The Heath bumblebee's thorax is black with two yellow to golden bands whereas the Broken-belted bumblebee's thorax is black with one yellow to golden band. The Heath bumblebee's abdomen is black with one yellow band near the top of it and a white tip whereas the Broken-belted bumblebee's abdomen is black with one yellow band around the middle of it and a white to buff tip.

## 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.

## 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 systems: usually statistical, including neural sequence-to-sequence models.



Overview of Natural Language Generation

Components of Natural Language Generation systems

Referring expressions

Summarization

## Components of a classical 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**

# 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:

$$\begin{aligned} \text{result}(\text{win}, t1, n) &\mapsto \_beat\_v(e, t1, t2), \_by\_p(e, r), \\ &\_run\_n(r), \text{card}(r, n) \\ \text{name}(t, C) &\mapsto \text{named}(t, C) \end{aligned}$$

This gives:

$$\begin{aligned} \text{name}(\text{team1}, \text{India}), \text{name}(\text{team2}, \text{Sri Lanka}), \\ \text{result}(\text{win}, \text{team1}, 63) &\mapsto \\ \text{named}(t1, \text{'India'}), \text{named}(t2, \text{'Sri Lanka'}), \\ \_beat\_v(e, t1, t2), \_by\_p(e, r), \_run\_n(r), \text{card}(r, \text{'63'}) \end{aligned}$$

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)  
 → 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.*

## Content selection: 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.*

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).

Overview of Natural Language Generation

Components of Natural Language Generation systems

**Referring expressions**

Summarization

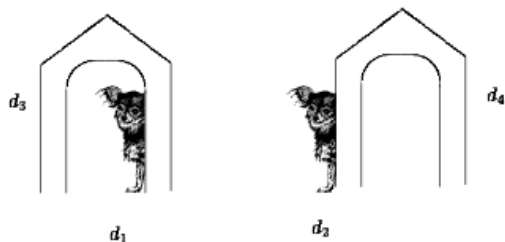
## Referring expressions

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

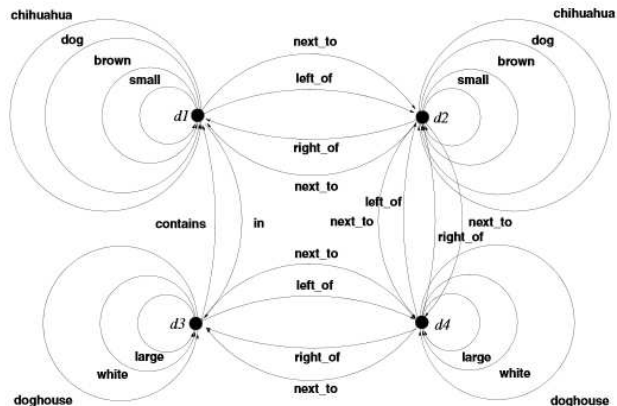


Figure 3  
A graph representation of the scene in Figure 2.

## 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: sensible if generating from a knowledge-base, otherwise corpus-based methods are needed.

Overview of Natural Language Generation

Components of Natural Language Generation systems

Referring expressions

**Summarization**

## Overview of summarization

- ▶ Pure form of task: reduce the length of a document.
- ▶ Most used for search results, question answering etc: different scenarios have different requirements.
- ▶ Multidocument summarization: e.g., bringing together information from different news reports.
- ▶ Two main system types:
  - ▶ Extractive: select sentences from a document. Possibly compress selected sentences.
  - ▶ Abstractive: use partial analysis of the text to build a summary.  
Approach here: 'proposition' based (Fang and Teufel, 2014, 2016; Fang et al, 2016).

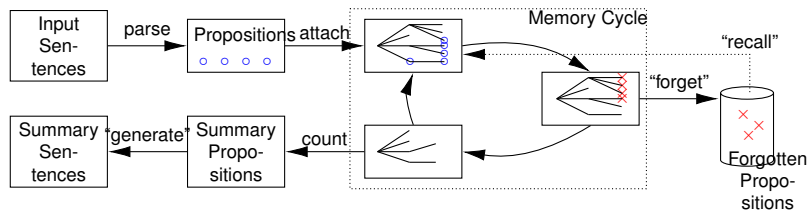
## Proposition-based summarizer

1. Parse each input sentence, create a set of propositions (e.g.: REVOLUTIONISE (DISCOVERY, FIRE-LIGHTING))
2. Attach each proposition to a **coherence tree**, using argument overlap.
3. Provisionally 'forget' some of the existing propositions.
4. Carry on to the next sentence.
5. If no coherence, recover temporarily forgotten propositions to bridge.

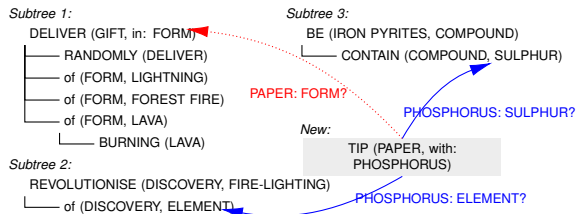
Finally: rank propositions by the number of cycles survived, realize summary from the highest ranked.



## Architecture (from Fang and Teufel (2016))



## Possible attachments of a new proposition

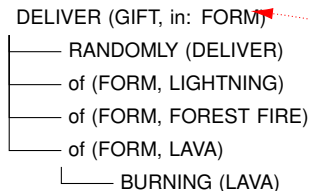


1. [fire was] a gift randomly delivered in the form of lightning, forest fire or burning lava
2. fire-lighting was revolutionised by the discovery of the element
3. iron pyrites, a compound that contains sulphur

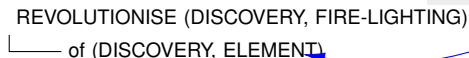
New proposition: paper tipped with phosphorus

## Possible attachments of a new proposition

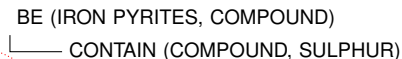
Subtree 1:



Subtree 2:



Subtree 3:



PAPER: FORM?

PHOSPHORUS: SULPHUR?

New:

TIP (PAPER, with:  
PHOSPHORUS)

PHOSPHORUS: ELEMENT?

from Fang and Teufel (2016)