Cutting the Cost of Referring Expression Generation

Robert Dale
Robert.Dale@mq.edu.au

Work done jointly with Jette Viethen

Your Keys

An RFID Tag Key Ring

Available now from GAO RFID Asset Tracking: http://www.gaorfidassettracking.com

A Hypothetical Conversation with The Room

You:	Hi, Room – where did I drop my keys?
The Room:	Um ... I think you'll find they're under the light blue chair second from the left-hand end of the third row from the back of the auditorium.

A Hypothetical Conversation with The Room

You:	Hi, Room – where did I drop my keys?
The Room:	Um ... I think you'll find they're under the light blue chair second from the left-hand end of the third row from the back of the auditorium.

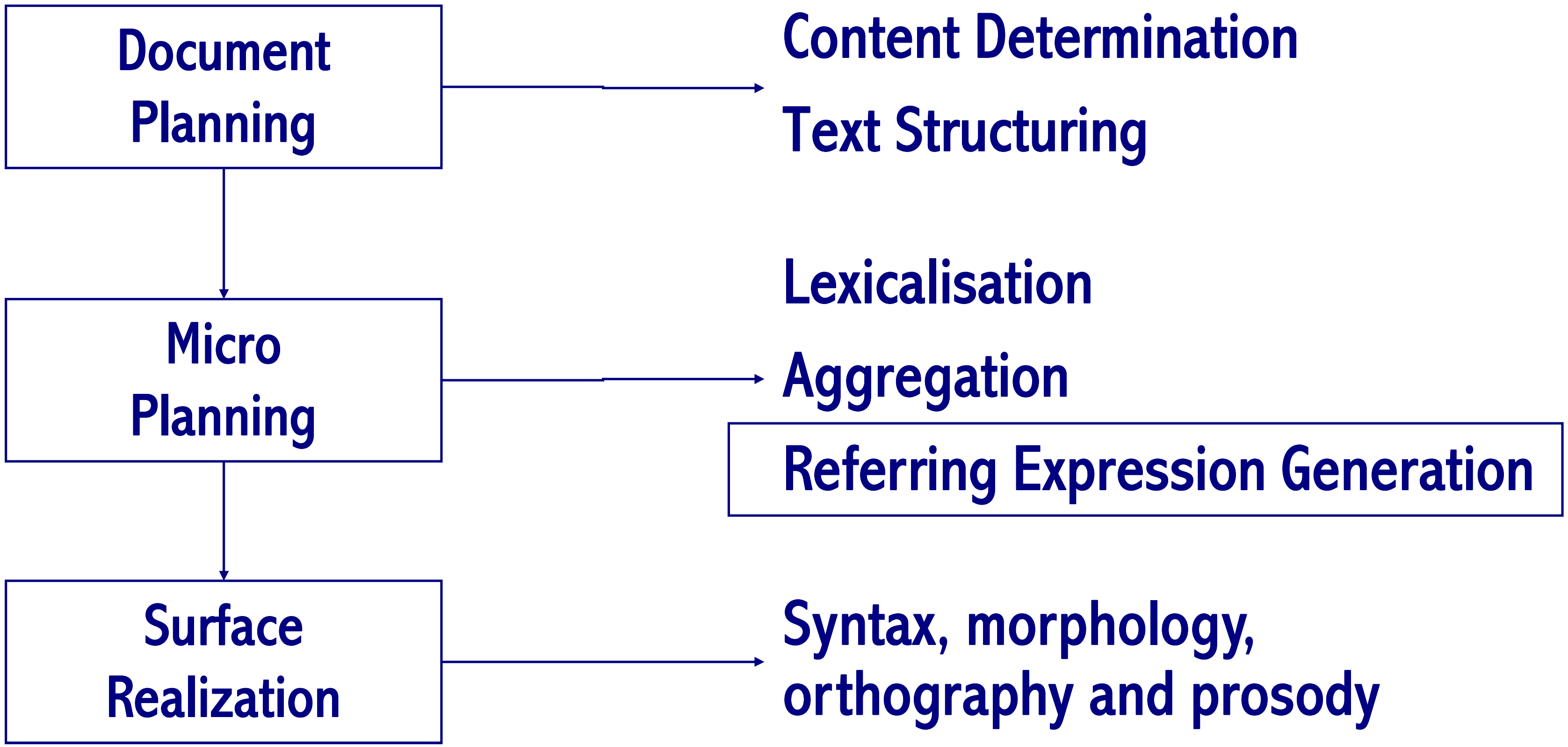
Outline

- The Context: Natural Language Generation
- Algorithms for Referring Expression Generation
- What People Do
- Towards a Better Computational Model
- Conclusions

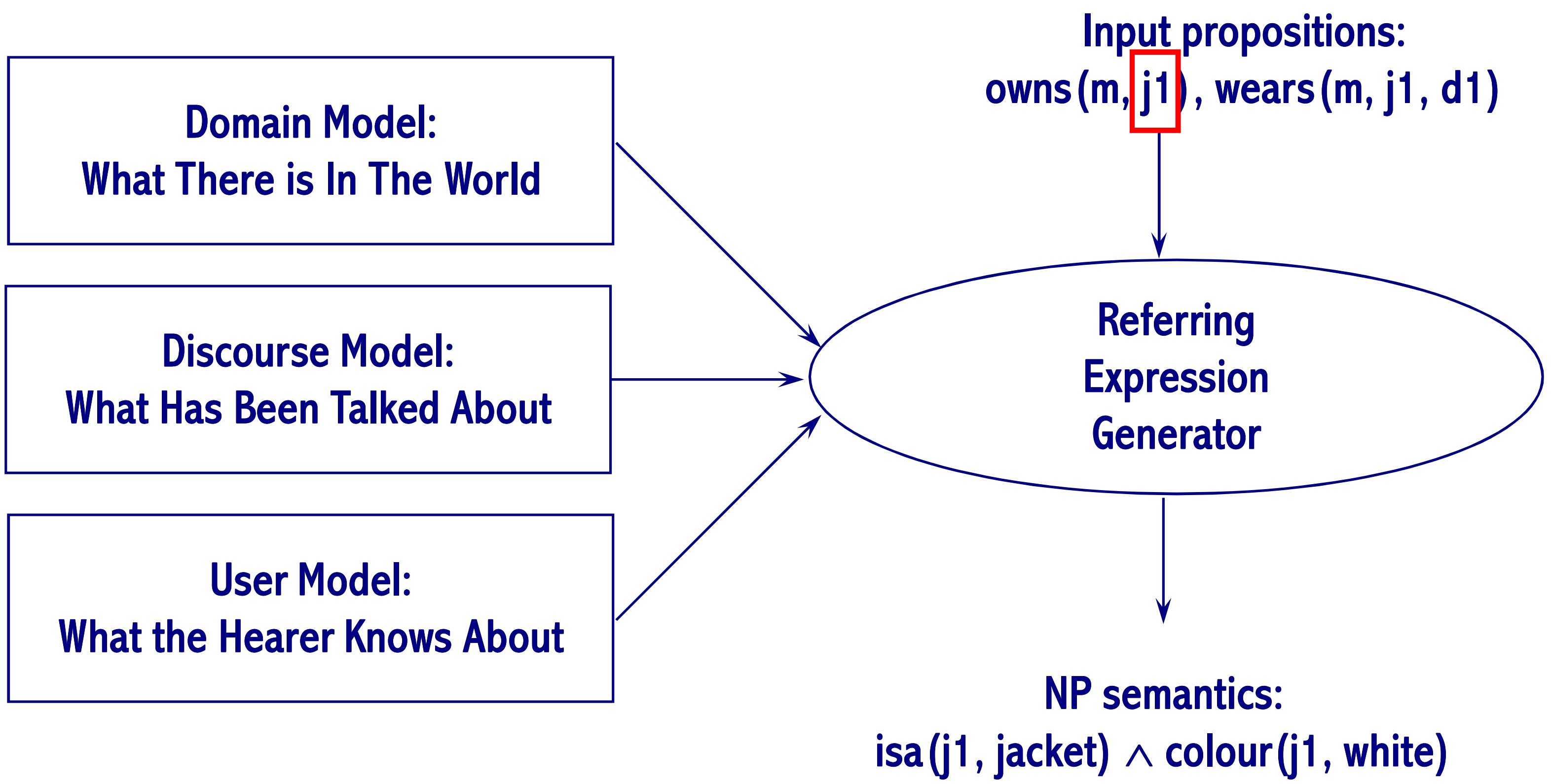
The Context

- Natural Language Generation is concerned with generating novel text from either (a) a non-linguistic base or (b) old text
- Important for applications:
  - any situation where it is not possible or practical to construct the full range of required outputs ahead of time
- Important for theory:
  - understanding what drives choice-making in language

A Standard Architecture for Generation



Referring Expression Generation



The Effect of Context on Reference

- Example 1:
  - owns(m, j1) → Matt owns a white jacket.
  - wears(m, j1, d) → He wears it on Sundays.
- Example 2:
  - owns(m, [j1+c1]) → Matt owns a white jacket and a white coat.
  - wears(m, j1, d) → He wears the jacket on Sundays.
- Example 3:
  - owns(m, [j1+j2]) → Matt owns a white jacket and a blue jacket.
  - wears(m, j1, d) → He wears the white one on Sundays.

Same

Different

Outline

- The Context: Natural Language Generation
- Algorithms for Referring Expression Generation
- What People Do
- Towards a Better Computational Model
- Conclusions

The Consensus Problem Statement

The goal:

Generate a distinguishing description

Given:

- an intended referent;
- a knowledge base of entities characterised by properties expressed as attribute–value pairs; and
- a context consisting of other entities that are salient;

Then:

- choose a set of attribute–value pairs that uniquely identify the intended referent

Guiding Principles

- Effectiveness
  - Say something that uniquely identifies the intended referent
- Efficiency
  - Say no more than is necessary
- Sensitivity
  - Say something the hearer understands

Computing Distinguishing Descriptions

Three steps which are repeated until a successful description has been constructed:

Start with a null description.

1. Check whether the description constructed so far is successful in picking out the intended referent from the context set. If so, quit.
2. If it's not sufficient, choose a property that will contribute to the description.
3. Extend the description with this property, and reduce the context set accordingly. Go to Step 1.

Computing Distinguishing Descriptions: The Greedy Algorithm [1989]

Initial Conditions:

$C_r = \langle$all entities$\rangle$; $P_r = \langle$all properties true of r$\rangle$; $L_r = \{\}$

1. Check Success

   if $|C_r| = 1$ then return L_r as a distinguishing description

   elseif $P_r = 0$ then return L_r as a non-dd

   else goto Step 2.

2. Choose Property

   for each $p_i \in P_r$ do: $C_{r_i} \leftarrow C_r \cap \{x \mid p_i(x)\}$

   Chosen property is p_j, where C_{r_j} is smallest set.

   goto Step 3.

3. Extend Description (wrt the chosen p_j)

   $L_r \leftarrow L_r \cup \{p_j\}$; $C_r \leftarrow C_{r_j}$; $P_r \leftarrow P_r - \{p_j\}$; goto Step 1.

Problems

- The algorithm is computationally expensive
- It does not guarantee to find a minimal distinguishing description
- It doesn't take account of the user

A Response: The Incremental Algorithm [1995]

Initial Conditions:

- $C_r = \langle$all entities$\rangle$; $P = \langle$preferred attributes$\rangle$; $L_r = \{\}$

1. Check Success
   - if $|C_r| = 1$ then return L_r as a distinguishing description
   - elseif $P = 0$ then return L_r as a non-dd
   - else goto Step 2.
2. Evaluate Next Property
   - get next $p_i \in P$ such that userknows$(p_i(r))$
   - if $|\{x \in C_r \mid p_i(x)\}| < |C_r|$ then goto Step 3
   - else goto Step 2.
3. Extend Description (wrt the chosen p_j)
   - $L_r \leftarrow L_r \cup \{p_j\}$; $C_r \leftarrow C_{rj}$; goto Step 1.

Key Properties of the Incremental Algorithm

- Important distinction between:
  - the way choices are made (domain independent)
  - the choices available (domain dependent)
- Computationally cheaper than the Greedy Algorithm

Why Is This Not a Good Model of What People Do?

1. People often produce redundant descriptions
2. People don't always produce distinguishing descriptions
3. The 'add a property, check how we're doing' model seems too computationally expensive to be plausible
4. Different people produce different descriptions in the same situation

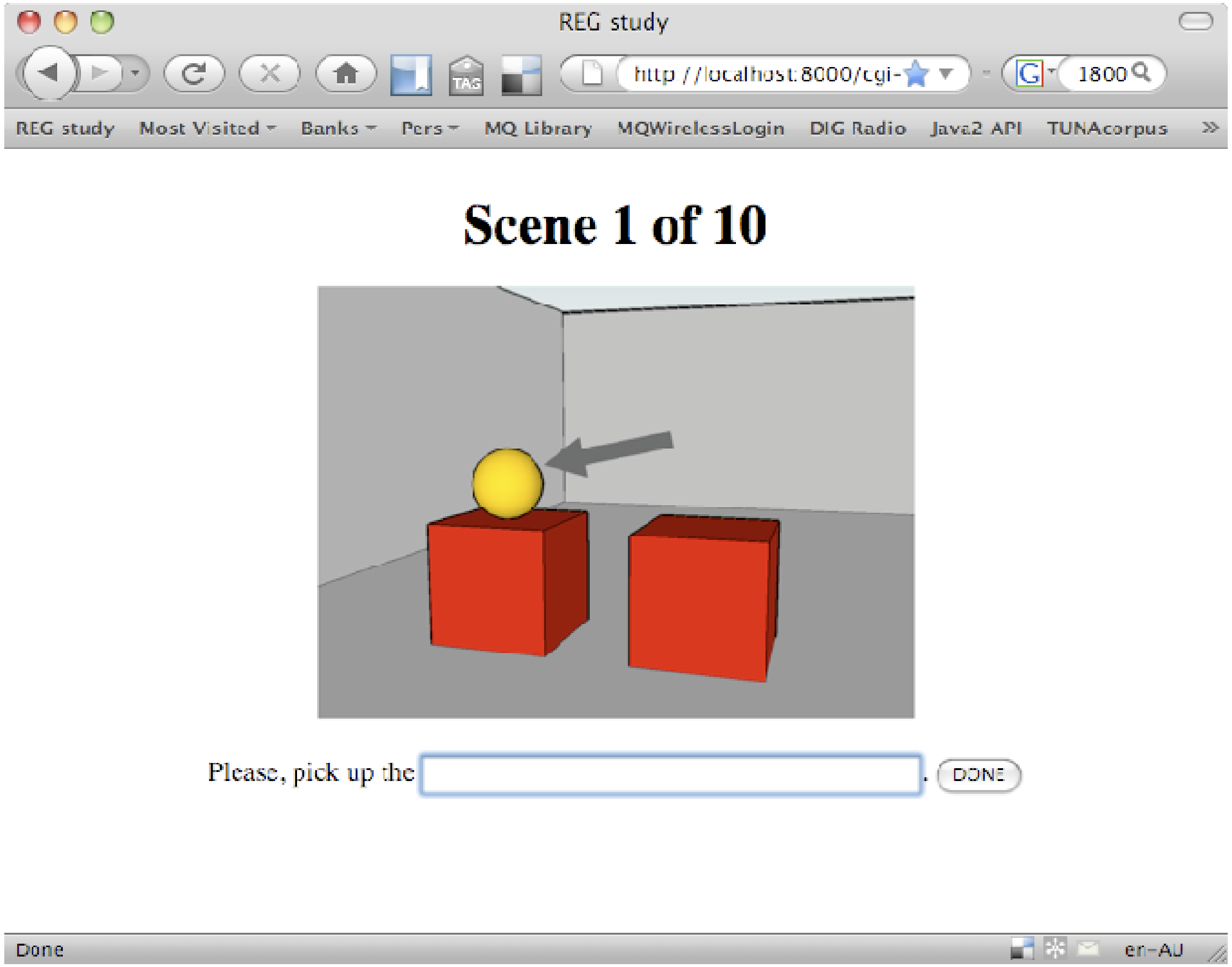
Outline

- The Context: Natural Language Generation
- Algorithms for Referring Expression Generation
- What People Do
- Towards a Better Computational Model
- Conclusions

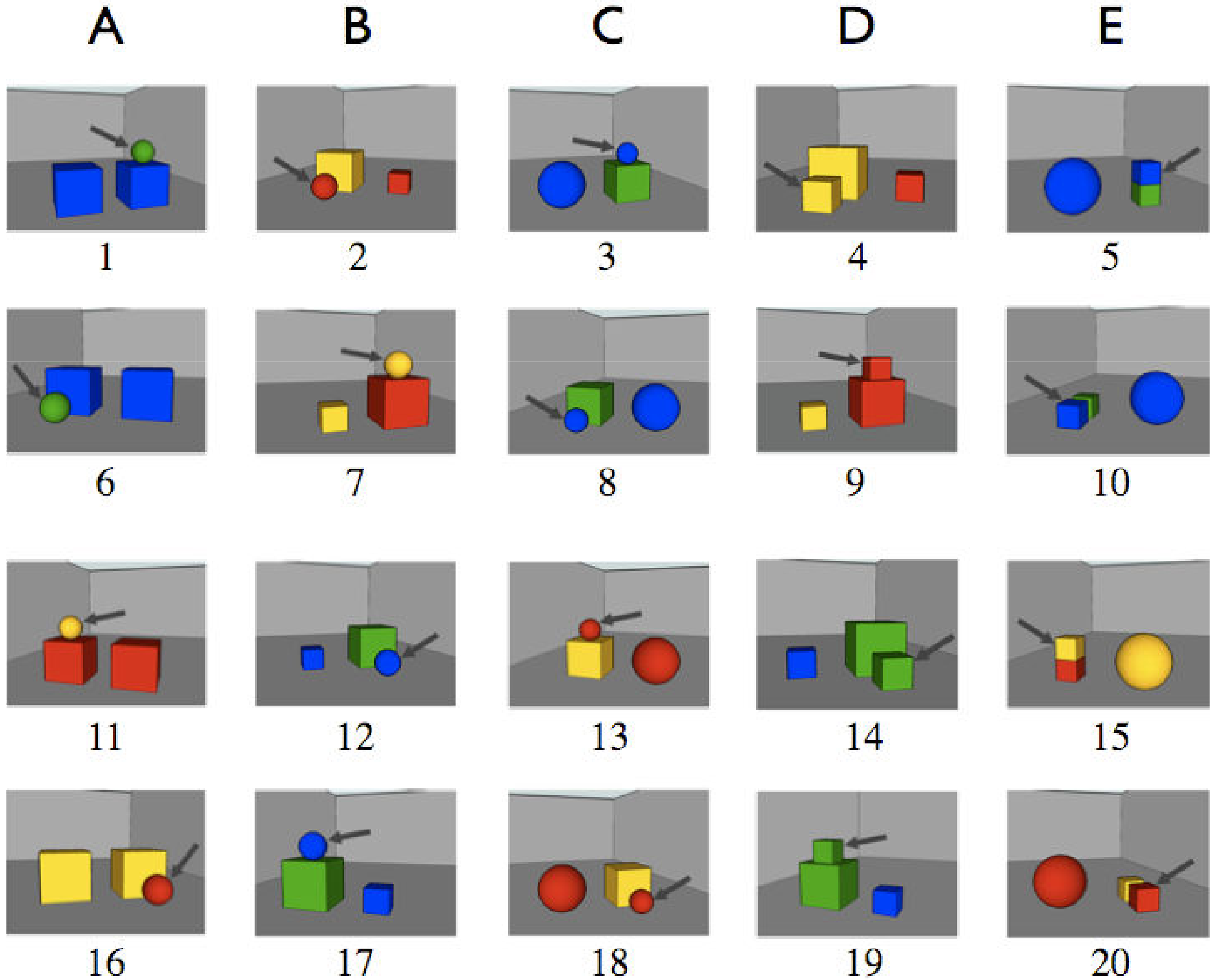
Human-Produced Data Sets

- The TUNA Corpus [van Deemter et al 2006]
  – 900 descriptions of furniture
  – 900 descriptions of people
- The GRE3D3 Corpus [Viethen and Dale 2008]
  – 630 descriptions of coloured blocks

The Experimental Setup



The Stimulus Scenes



Data Filtering and Normalisation

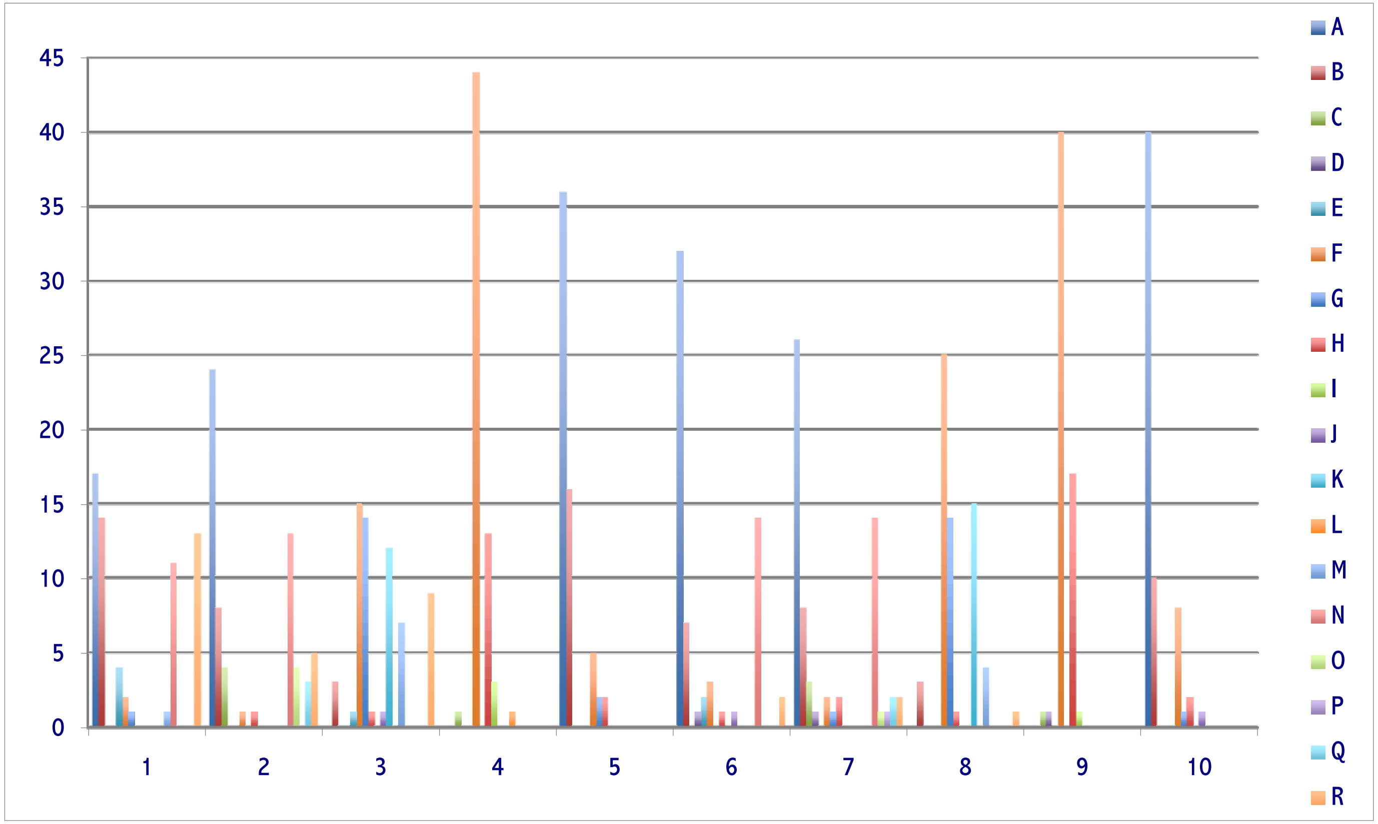
- 74 participants:
  - One asked for data to be discarded; one reported as being colour blind; one used very long referring expressions referring to the onlooker; eight participants only used type in their descriptions
- Normalisation:
  - Spelling mistakes corrected; colour names and head nouns normalised; complex syntactic structures simplified

→ 623 scene descriptions

Description Patterns

Label	Pattern	Example
A	⟨tg_col, tg_type⟩	*the blue cube*
B	⟨tg_col, tg_type, rel, lm_col, lm_type⟩	*the blue cube in front of the red ball*
C	⟨tg_col, tg_type, rel, lm_size, lm_col, lm_type⟩	*the blue cube in front of the large red ball*
D	⟨tg_col, tg_type, rel, lm_size, lm_type⟩	*the blue cube in front of the large ball*
E	⟨tg_col, tg_type, rel, lm_type⟩	*the blue cube in front of the ball*
F	⟨tg_size, tg_col, tg_type⟩	*the large blue cube*
G	⟨tg_size, tg_col, tg_type, rel, lm_col, lm_type⟩	*the large blue cube in front of the red ball*
H	⟨tg_size, tg_col, tg_type, rel, lm_size, lm_col, lm_type⟩	*the large blue cube in front of the large red ball*
I	⟨tg_size, tg_col, tg_type, rel, lm_size, lm_type⟩	*the large blue cube in front of the large ball*
J	⟨tg_size, tg_col, tg_type, rel, lm_type⟩	*the large blue cube in front of the ball*
K	⟨tg_size, tg_type⟩	*the large cube*
L	⟨tg_size, tg_type, rel, lm_size, lm_type⟩	*the large cube in front of the large ball*
M	⟨tg_size, tg_type, rel, lm_type⟩	*the large cube in front of the ball*
N	⟨tg_type⟩	*the cube*
O	⟨tg_type, rel, lm_col, lm_type⟩	*the cube in front of the red ball*
P	⟨tg_type, rel, lm_size, lm_col, lm_type⟩	*the cube in front of the large red ball*
Q	⟨tg_type, rel, lm_size, lm_type⟩	*the cube in front of the large ball*
R	⟨tg_type, rel, lm_type⟩	*the cube in front of the ball*

Distribution of Patterns Across Scenes



Distribution of Patterns Across Scenes

Pattern		Scene # 1	2	3	4	5	6	7	8	9	10
A	tg_col, tg_type	17	24			36	32	26			40
B	tg_col, tg_type, rel, lm_col, lm_type	14	8	3		16	7	8	3		10
C	tg_col, tg_type, rel, lm_size, lm_col, lm_type		4		1			3		1	
D	tg_col, tg_type, rel, lm_size, lm_type						1	1		1	
E	tg_col, tg_type, rel, lm_type	4		1			2				
F	tg_size, tg_col, tg_type	2	1	15	44	5	3	2	25	40	8
G	tg_size, tg_col, tg_type, rel, lm_col, lm_type	1		14		2		1	14		1
H	tg_size, tg_col, tg_type, rel, lm_size, lm_col, lm_type		1	1	13	2	1	2	1	17	2
I	tg_size, tg_col, tg_type, rel, lm_size, lm_type				3					1	
J	tg_size, tg_col, tg_type, rel, lm_type			1			1				1
K	tg_size, tg_type			12					15		
L	tg_size, tg_type, rel, lm_size, lm_type				1						
M	tg_size, tg_type, rel, lm_type	1		7					4		
N	tg_type	11	13				14	14			
O	tg_type, rel, lm_col, lm_type		4					1			
P	tg_type, rel, lm_size, lm_col, lm_type							1			
Q	tg_type, rel, lm_size, lm_type		3					2			
R	tg_type, rel, lm_type	13	5	9			2	2	1		

Some Questions

- What exactly are we trying to model – an ideal speaker?
- What is an ideal speaker?
- How do we account for the variation amongst real speakers?

Outline

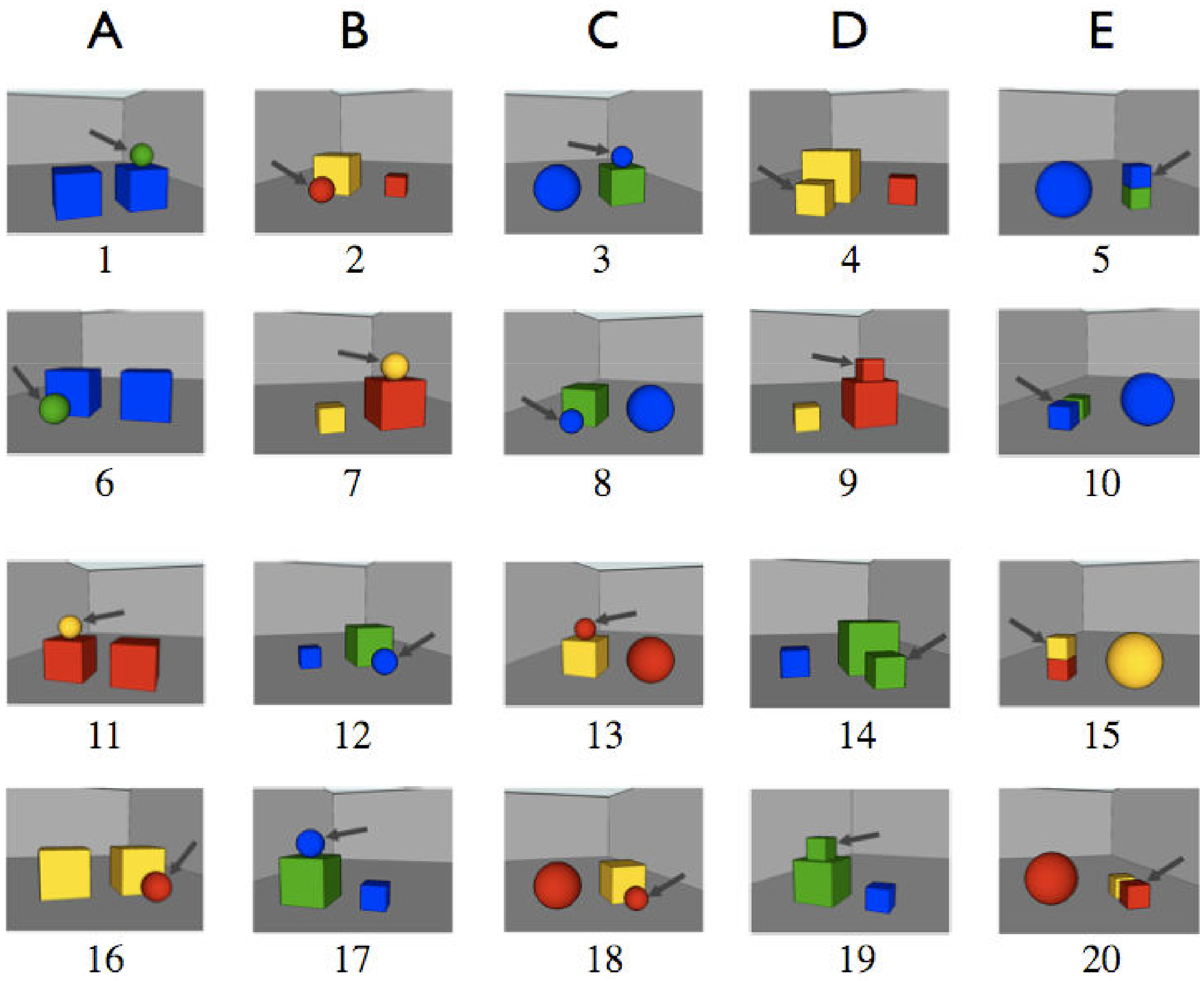
- The Context: Natural Language Generation
- Algorithms for Referring Expression Generation
- What People Do
- Towards a Better Computational Model
- Conclusions

A Machine Learning Experiment

Can we use human data to learn how to refer?

1. Identify relevant characteristics of scenes
2. See if these can be correlated with description patterns via a machine learner

The Scenes



Characteristics of Scenes

Label	Attribute	Values
tg_type = lm_type	Target and Landmark share Type	TRUE, FALSE
tg_type = dr_type	Target and Distractor share Type	TRUE, FALSE
lm_type = dr_type	Landmark and Distractor share Type	TRUE, FALSE
tg_col = lm_col	Target and Landmark share Colour	TRUE, FALSE
tg_col = dr_col	Target and Distractor share Colour	TRUE, FALSE
lm_col = dr_col	Landmark and Distractor share Colour	TRUE, FALSE
tg_size = lm_size	Target and Landmark share Size	TRUE, FALSE
tg_size = dr_size	Target and Distractor share Size	TRUE, FALSE
lm_size = dr_size	Landmark and Distractor share Size	TRUE, FALSE
rel	Relation between Target and Landmark	on top of, in front of

Description Patterns

Label	Pattern	Example
A	⟨tg_col, tg_type⟩	*the blue cube*
B	⟨tg_col, tg_type, rel, lm_col, lm_type⟩	*the blue cube in front of the red ball*
C	⟨tg_col, tg_type, rel, lm_size, lm_col, lm_type⟩	*the blue cube in front of the large red ball*
D	⟨tg_col, tg_type, rel, lm_size, lm_type⟩	*the blue cube in front of the large ball*
E	⟨tg_col, tg_type, rel, lm_type⟩	*the blue cube in front of the ball*
F	⟨tg_size, tg_col, tg_type⟩	*the large blue cube*
G	⟨tg_size, tg_col, tg_type, rel, lm_col, lm_type⟩	*the large blue cube in front of the red ball*
H	⟨tg_size, tg_col, tg_type, rel, lm_size, lm_col, lm_type⟩	*the large blue cube in front of the large red ball*
I	⟨tg_size, tg_col, tg_type, rel, lm_size, lm_type⟩	*the large blue cube in front of the large ball*
J	⟨tg_size, tg_col, tg_type, rel, lm_type⟩	*the large blue cube in front of the ball*
K	⟨tg_size, tg_type⟩	*the large cube*
L	⟨tg_size, tg_type, rel, lm_size, lm_type⟩	*the large cube in front of the large ball*
M	⟨tg_size, tg_type, rel, lm_type⟩	*the large cube in front of the ball*
N	⟨tg_type⟩	*the cube*
O	⟨tg_type, rel, lm_col, lm_type⟩	*the cube in front of the red ball*
P	⟨tg_type, rel, lm_size, lm_col, lm_type⟩	*the cube in front of the large red ball*
Q	⟨tg_type, rel, lm_size, lm_type⟩	*the cube in front of the large ball*
R	⟨tg_type, rel, lm_type⟩	*the cube in front of the ball*

Results

- Weka J48 pruned decision tree classifier
- Predicts actual form of reference in 48% of cases under 10-fold cross validation
- The rule learned:

```
if target-type = distractor-type
    then use pattern F (⟨tg_size, tg_col, tg_type⟩)
    else use pattern A (⟨ tg_col, tg_type⟩)
endif
```

Distribution of Patterns Across Scenes

	Scene #									
Pattern	1	2	3	4	5	6	7	8	9	10
A tg_col, tg_type	17	24			36	32	26			40
B tg_col, tg_type, rel, lm_col, lm_type	14	8	3		16	7	8	3		10
C tg_col, tg_type, rel, lm_size, lm_col, lm_type		4		1			3		1	
D tg_col, tg_type, rel, lm_size, lm_type						1	1		1	
E tg_col, tg_type, rel, lm_type	4		1			2				
F tg_size, tg_col, tg_type	2	1	15	44	5	3	2	25	40	8
G tg_size, tg_col, tg_type, rel, lm_col, lm_type	1		14		2		1	14		1
H tg_size, tg_col, tg_type, rel, lm_size, lm_col, lm_type		1	1	13	2	1	2	1	17	2
I tg_size, tg_col, tg_type, rel, lm_size, lm_type				3					1	
J tg_size, tg_col, tg_type, rel, lm_type			1			1				1
K tg_size, tg_type			12					15		
L tg_size, tg_type, rel, lm_size, lm_type				1						
M tg_size, tg_type, rel, lm_type	1		7					4		
N tg_type	11	13				14	14			
O tg_type, rel, lm_col, lm_type		4					1			
P tg_type, rel, lm_size, lm_col, lm_type							1			
Q tg_type, rel, lm_size, lm_type		3					2			
R tg_type, rel, lm_type	13	5	9			2	2	1		

What About Speaker Difference?

- As well as the characteristics of scenes, add participant ID as a feature
- Description pattern prediction increases to 57.62%
- So: it may be possible to learn individual differences from the data

Interim Conclusions

- We can learn a ‘correct answer’ for every scene
- We can’t explain the diversity in forms of reference

An Alternative Approach

- People build different descriptions for the same intended referent in the same scene
- Are we looking for commonality in the wrong place?
  - Maybe the decision processes around each specific attribute are less varied

Learning the Presence or Absence of Individual Properties

Attribute to Include	Baseline (0-R)
Target Colour	78.33%
Target Size	57.46%
Relation	64.04%
Landmark Colour	74.80%
Landmark Size	88.92%

Example: Heuristics for Target Colour Inclusion

1. Always use colour [37 participants]
2. If the target and the landmark are of the same type, use colour [all the rest]
3. If the target and the landmark are not of the same type then:
   i. Exclude colour [19 participants]
   ii. Use colour if target and distractor are the same size [4]
   iii. Use colour if target and distractor share size and the target is on top of the landmark [2]
   iv. Use colour if target and distractor share colour [1]

What Does This Mean?

- Everybody's different, but we often have some things in common:
  - A speaker profile consists of a collection of attribute-specific heuristics
  - Speaker profiles can vary significantly but be based on a set of commonly used attribute-specific heuristics
- The heuristics a particular speaker uses in a given situation may depend on a variety of contextual and personal-history factors

Speaker Profiles

#	tg_col	tg_size	tg_size	rel	lm_size
13	TgCol-T	TgSize-1	Rel-F	n/a	n/a
10	TgCol-T	TgSize-1	Rel-T	LmCol-T	LmSize-1
9	TgCol-1	TgSize-1	Rel-F	n/a	n/a
2	TgCol-3	TgSize-1	Rel-4	LmCol-F	LmSize-1
2	TgCol-T	TgSize-1	Rel-2	LmCol-T	LmSize-1
2	TgCol-1	TgSize-1	Rel-T	LmCol-1	LmSize-1

- TgCol-T = always include tg colour
- TgSize-1 = include target size if target and distractor share type
- Rel-F = never use a relation

Implications for Algorithm Development

- Each property is different: reduction to a single metric of value (such as discriminatory power) is too simplistic
- Properties may be included independently of other properties
- An alternative to the ‘add one then check’ model:
  - A ‘read off the scene’ model: gestalt analysis of a scene results in several properties being chosen in parallel
  - Properties are selected on the basis of simple heuristics, not on the basis of reflection as to whether they truly make a difference

Cost Reduction in Referring Expression Generation

- First proposals:
  - 'full brevity', high computational complexity: carefully evaluate all the alternatives
- Second generation:
  - use a precomputed preference-order over properties
- Third generation:
  - independently pick properties that look promising on the basis of past experience

What About Subsequent Reference?

- In dialog, people converge (align) to the same descriptions
- Observation:
  - Most references are to entities which have already been referred to, in contexts which have not changed since the last reference
- Consequence:
  - Why compute? Just copy the last reference!

Before

- If this is an initial reference
  - Choose a perspective [$?]
  - Produce a minimal distinguishing description for the intended referent [$$$]
- If this is a subsequent reference
  - Produce a minimal distinguishing description for the intended referent [$$$]

After

- If this is an initial reference
  - Choose a perspective [$?]
  - Take a guess at a form of reference that might work [$$]
- If this is a subsequent reference
  - Unless something in the context has changed, just copy the last reference [$]

Outline

- The Context: Natural Language Generation
- Algorithms for Referring Expression Generation
- What People Do
- Towards a Better Computational Model
- Conclusions

Is This The Whole Story?

- No. Sometimes we do reflect on the referring expression constructed so far, and add more:
  - Uhm, I'm gonna transfer to the phone on the table by the red chair . . . [points in the direction of the phone] the . . . the red chair, against the wall, uh the little table, with the lamp on it, the lamp that we moved from the corner? . . . the black phone, not the brown phone . . .

    [Lucy from 'Twin Peaks']

New Questions

- What properties of a scene just ‘jump out’?
- How do we decide if the first cut is good enough? How and when do more reflective reasoning processes kick in?
- How are speaker profiles modified dynamically through alignment and learned success?

Conclusions

- Existing algorithms, based on a cycle of ‘add a carefully-considered property then check how we’re doing’, don’t acknowledge ‘bounded rationality’
- A better model: different speakers use different heuristics for property inclusion in different circumstances
- Heuristics are simple, and likely based on individual history and other factors
- There is no gold standard (so evaluation is a challenge!)

Some Lessons Learned

- Don't look for complex solutions that cover all cases when a simpler solution works most of the time
- Acknowledge that human language use is characterised by bounded rationality and risk-taking, so perhaps our algorithms should be too