lecture 4, part 1:
Boolean extensions of the incremental algorithm

Read: van Deemter (2002). “Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm” *Computational Linguistics* **28** (1), pp.37-52. March 2002

Plan of the lecture:

– Reference to sets

– Boolean incompleteness of earlier algorithms

– A direct extension of the incremental algorithm

– An alternative approach using *satellite sets*

– Problems (combinations with salience etc.)

- GRE is microcosm of NLG, involving

  Content Determination

  (Text planning)

  Lexicalization

  Sentence Aggregation

  Linguistic Realization

- Focus on *Content Determination*

- Radical assumption: CD can be done
  before everything else

- Properties = {Red, Alsacian,Owner=John}

  No distinction between *'She was molested by ...*

  a. *... John's redhaired alsacian'*

  b. *... a German shepherd dog; the red one, you know, which is owned by John'*

  (See lecture 5 for a different story)

What makes a GRE algorithm good?

Three criteria:

- Logical completeness of algorithm
- Computational tractability of algorithm
- Naturalness of its output

The Incremental Algorithm

Completeness: individuate an object using conjunctions of properties

Naturalness: for example,

Full Brevity (Dale 1989, 1992):

Use minimal number of properties

Dale & Reiter '95: Full Brevity implies

exponential complexity

Therefore: Approximate *Full Brevity* by incrementally combining properties until only the target satisfies them

The Incremental Algorithm

Target $= r$

$\mathcal{P} =$ list of properties in order of preference

Algorithm delivers set L

$L = \langle P_1, ..., P_n \rangle$

such that $P_1 \cap ... \cap P_n = \{r\}$.

- Start with most preferred property in $\mathcal{P}$
- If it's any good then include it
- Move on to next-most preferred property in $\mathcal{P}$
- and so on, until $C = \{r\}$

Incremental Algorithm (schematic)

$L := \oslash$

$C := Domain$

For each $P \in \mathcal{P}$ do

If $r \in [[P]]$ & $C \not\subseteq [[P]]$ then do

$L := L \cup \{P\}$

$C := C \cap [[P]]$

If $C = \{r\}$ then Return L

Return *Failure*

Some properties

– Hill climbing

– Heuristic search

– Fast (polynomial)

Some recent extensions

First extension: reference to sets
(van Deemter 2000)

Identify a *set* of objects, to say that
each of its elements has a certain property
(e.g., being dangerous)

dog $\boldsymbol{a}$ is dangerous and
dog $\boldsymbol{b}$ is dangerous and
dog $\boldsymbol{c}$ is dangerous

Target $\boldsymbol{S} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$

E.g., '*The chihuahuas* are dangerous'

N.B. Applicable whether or not each of $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$
can be identified separately
($\Rightarrow$ aggregation!)

Generate descriptions of sets:

$L := \varnothing$

$C := Domain$

For each $P \in \mathcal{P}$ do

If $S \subseteq [[P]]$ & $C \not\subseteq [[P]]$ then do

$L := L \cup \{P\}$

$C := C \cap [[P]]$

If $C = S$ then Return L

Return *Failure*

– Individual targets modeled as singletons $\{r\}$

– What if something holds of $\{a, b, c\}$ collectively, e.g., they are only dangerous as a group? (Stone 2000, van Deemter 2002)

  * Task: individuate sets of sets $\{S_1, .., S_n\}$

  * Special case: individuate one set $\{S\}$

  * Start out with powerset of Domain: $C := \mathcal{P}(\mathcal{D})$

  * Properties P hold of sets

    E.g., BEING A TEAM can be a property of a set of players

  * Add properties P; remove from C all those sets for which P does not hold

'Football teams all of whose members are British'

Second extension: beyond conjunction

N.B. This relates to **completeness**:

Is the algorithm able to individuate
every entity that *can* be individuated?

Example

Domain: $\{a, b, c, d, e, f\}$

TYPE: dog $\{a, b, c, d, e\}$, poodle $\{a, b\}$
COLOUR: black $\{a, b, c\}$, white $\{d, e, f\}$

How would you individuate c?

D&R does not produce a description of c. Yet,

$$\{c\} = black \cap \overline{poodle}$$

E.g., *'The black animal that's not a poodle'*,

N.B. Extension is even relevant for singletons

Disjunctions

Domain: $\{a, b, c, d, e, f\}$

TYPE: dog $\{a, b, c, d, e\}$, poodle $\{a, b\}$
COLOUR: black $\{a, b, c\}$, white $\{d, e, f\}$

Incremental algorithm does not describe $\{a, b, d, e\}$

Yet,

$$\{a, b, d, e\} = (white \cap dog) \cup poodle$$

'The white dogs and the poodles'

N.B. Confusion: and/or, logic/sets

One solution

Extension of D&R's incremental algorithm

– First accumulate properties from $\mathcal{P}$,
plus their negations

– then more and more complex properties ...

until `success` or `failure`

Consequences:

– More operations to play with

– More potential descriptions

– Failure occurs less often

– Embarrassment of riches: how do we choose?

D&R$_{Boolean}$ in some detail:

D&R$_{Boolean}$

1. D&R phase using
all properties of the form $\boldsymbol{P}_{+/-}$
If SUCCESS then STOP, else go to **phase (2)**

2. D&R phase using
all properties of the form $\boldsymbol{P}_{+/-} \cup \boldsymbol{P}_{+/-}$
If SUCCESS then STOP, else go to **phase (3)**

3. D&R phase using
all properties of the form $\boldsymbol{P}_{+/-} \cup \boldsymbol{P}_{+/-} \cup \boldsymbol{P}_{+/-}$
If SUCCESS then STOP, else go to **phase (4)**

4. *Etcetera*

Example

Domain: $\{a, b, c, d, e\}$

Target: $S = \{a, b, d, e\}$

TYPE: dog $\{a, b, c, d, e\}$, poodle $\{a, b\}$
COLOUR: black $\{a, b, c\}$, white $\{d, e\}$

Phase 1: $\mathcal{P} =$

$$Dog, Poodle, Black, White,$$

$$\overline{Dog}, \overline{Poodle}, \overline{Black}, \overline{White}$$

No property selected

Recall: Target $= \{a, b, d, e\}$

Phase 2: $\mathcal{P} =$ all disjunctions $P_{+/-} \cup P_{+/-}$

For example,

a. $\overline{Poodle} \cup White = \{c, d, e\}$

b. $White \cup Poodle = \{a, b, d, e\}$

a. Too restrictive

b. Exactly right

'The poodles and the white animals'

Properties

– Incremental within each phase

– Incremental from phase to phase:

No backtracking to previous phases

– Exponential complexity

– Cut off algorithm after x phases

$\Rightarrow$ complexity becomes polynomial

E.g., $x = 2 \Rightarrow$ complexity is $O(n^3)$, but

'the poodle and the white dogs and the alsacians'

cannot be expressed.

– *Question:* how important is theoretical complexity?

– **Completeness:** Fine (Proof in CL paper)

– **Tractability:** OK

– **Naturalness:** Not so sure ...

∗ *Problem 1:* D&R$_{\boldsymbol{Boolean}}$ predicts that fewer ∪'s is always better. E.g., $\boldsymbol{a_1 \cap ... \cap a_{100}}$ is preferred over $\boldsymbol{b_1 \cup b_2}$

∗ *Problem 2:* D&R$_{\boldsymbol{Boolean}}$ may choose $\boldsymbol{a_1 \cap ... \cap a_{100} \cap (b_1 \cup b_2)}$ where $\boldsymbol{b_1 \cup b_2}$ would have been enough

So, finetuning of Boolean extension is necessary

Challenge: Reconcile

Naturalness (e.g., Brevity)
and **Tractability**.

Example of Problem 2

As before, with f added to the **Domain**:

Domain: $\{a, b, c, d, e, f\}$

Target: $S = \{a, b, d, e\}$

TYPE: dog $\{a, b, c, d, e\}$, poodle $\{a, b\}$
COLOUR: black $\{a, b, c\}$, white $\{d, e\}$

Phase 1: Dog removes f

Phase 2: $White \cup Poodle$ removes c

Note:

$White \cup Poodle$ is enough, since

$White \cup Poodle = \{a, b, d, e\} = S$

‘*The poodles and the white animals* that are dogs’

2. Alternative solution: Generate and optimize

(Joint work with Magnús M. Halldórsson)

Which objects *cannot* be separated? **First**, intersections

TYPE: dog $\{a, b, c, d, e\}$, poodle $\{a, b\}$
COLOUR: black $\{a, b, c\}$, white $\{d, e\}$

$$\text{Satellites}(x) =_{Def} \cap\{P : x \epsilon \mathcal{P}\}$$

$$\text{Satellites}(a) = dog \cap poodle \cap black = \{a, b\}$$
$$\text{Satellites}(b) = dog \cap poodle \cap black = \{a, b\}$$
$$\text{Satellites}(c) = dog \cap black = \{a, b, c\}$$
$$\text{Satellites}(d) = dog \cap white = \{d, e\}$$
$$\text{Satellites}(e) = dog \cap white = \{d, e\}$$

Observation 1:
No object can be separated from all others!

Observation 2:
Some *sets* can be separated from all others

Objective:

Exploit this idea for GRE

Second, add negations:

$$\overline{\mathcal{P}} =_{Def} \mathcal{P} \cup \{\overline{P} : P \in \mathcal{P}\}$$

$$\text{Satellites}(x) =_{Def} \cap \{P : P \in \overline{\mathcal{P}} \ \& \ x \in P\}$$

$\overline{\mathcal{P}} =$

$\boldsymbol{Dog}, \boldsymbol{Poodle}, \boldsymbol{Black}, \boldsymbol{White},$

$\overline{\boldsymbol{Dog}}, \overline{\boldsymbol{Poodle}}, \overline{\boldsymbol{Black}}, \overline{\boldsymbol{White}}$

TYPE:

$\boldsymbol{dog}(\{a, b, c, d, e\}), \boldsymbol{poodle}(\{a, b\})$
$\overline{\boldsymbol{dog}}(\phi), \overline{\boldsymbol{poodle}}(\{c, d, e\})$

COLOUR:

$\boldsymbol{black}(\{a, b, c\}), \boldsymbol{white}(\{d, e\})$
$\overline{\boldsymbol{black}}(\{d, e\}), \overline{\boldsymbol{white}}(\{a, b, c\})$

Satellites$(a) = \{a, b\}$
Satellites$(b) = \{a, b\}$
Satellites$(c) = \{c\}$ (Was: $\{a, b, c\}$)
Satellites$(d) = \{d, e\}$
Satellites$(e) = \{d, e\}$

Thus, $\boldsymbol{c} =$

$\boldsymbol{dog} \cap \overline{\boldsymbol{poodle}} \cap \boldsymbol{black} \cap \overline{\boldsymbol{white}}$

Third: **Disjunctions** can also be covered,

if we allow set **unions** between Satellite sets

Target: Set S of objects

Step 1: Compute Satellite set of each object

Step 2: Unify these Satellite sets

Example: Target $S = \{a, b, d, e\}$

Unify Satellite sets of a, b, d, e:

$\text{Satellites}(a) = \{a, b\} \cup$
$\text{Satellites}(b) = \{a, b\} \cup$
$\text{Satellites}(d) = \{d, e\} \cup$
$\text{Satellites}(e) = \{d, e\}$

General form: $Sat \cup \ldots \cup Sat$

where $Sat = P_{+/-} \cap \ldots \cap P_{+/-}$

Result:

$$\text{Satellites} = (a)\ (dog \cap poodle \cap black) \cup$$
$$\text{Satellites} = (b)\ (dog \cap poodle \cap black) \cup$$
$$\text{Satellites} = (d)\ (dog \cap white) \cup$$
$$\text{Satellites} = (e)\ (dog \cap white)$$

$$= \{a, b, d, e\}$$

'The objects that are
either *black poodle dogs*
or *black poodle dogs*
or *white dogs*
or *white dogs'*

Problem: This is hardly *brief* ...

Advantage: The Satellite set algorithm is fast!

$n_a =$ number of properties in $\mathcal{P}$

$n_s =$ number of elements in S

Computing Satellites(x) takes n_a calculations

This is done n_s times

Complexity is $O(n_s.n_a)$

N.B. Satellites(x) is calculated only once!

Satellite sets have two uses:

1. Determine whether a description is possible:

$$\text{Target} \neq \text{Union of satellite sets}$$

$$\Rightarrow$$

Target cannot be described

2. *Optimize* Union of satellite sets.

First optimization:

Construct Satellites(x) incrementally

Example:

Satellites$(a) =$

$dog = \{a, b, c, d, e\} \cap$
$poodle = \{a, b\} \cap$
$(black = \{a, b, c\})$

$= \{a, b\}$

N.B.: Incrementality does not shorten the calculation, but only the outcome!

Second optimization:

Construct **Union of Satellites** incrementally

Example:

$\text{Satellites}(a)\ (dog, poodle) \cup$
$(\text{Satellites}(b)\ (dog, poodle) \cup)$
$\text{Satellites}(d)\ (dog, white) \cup$
$(\text{Satellites}(e)\ (dog, white))$

$= \{a, b, d, e\}$

Result:

$(dog \cap poodle) \cup (dog \cap white)$

‘The poodle dogs and the white dogs’

Third optimization: Boolean simplification

Simplification of Boolean expressions in chips design

(E.g., **Quine/McCluskey** algorithm)

Example:

$$(\boldsymbol{dog} \cap \boldsymbol{poodle}) \cup (\boldsymbol{dog} \cap \boldsymbol{white}) =$$

$$\boldsymbol{dog} \cap (\boldsymbol{poodle} \cup \boldsymbol{white})$$

'The dogs that are either poodles or white'

N.B. Optimizations are work in progress

Full Brevity remains intractable!

Problems when extensions are integrated

1. Different formalisations

– different approaches to ling. realization

– often not implemented

2. Booleans + Relations

∗ Even more patterns to compare

∗ Incremental approach becomes
even more doubtfull

∗ What's more natural?:

'The dogs that sleep in a doghouse' (**1** x R)
'The dogs that are not Alsacians' (**1** x Neg)
'The poodles and the chihuahuas' (**1** x Union)

3. Sets + Salience

> *'A family of five labradors have taken refuge in our shed. Two of them are really sweet, and we're likely to keep them.* The animals ...'
>
> = the two most salient labradors?
>
> = the five most salient labradors?
>
> = all the animals in the world?

For example,

$\boldsymbol{Sal}$(five labradors)$= \mathbf{10}$
$\boldsymbol{Sal}$(two labradors)$= \mathbf{20}$
$\boldsymbol{Sal}$(we)$= \mathbf{20}$

Cause of ambiguity: Unless hearer knows who $\boldsymbol{r}$ is, she has no way of determining $\boldsymbol{C}$:

$$\boldsymbol{C := \{x \epsilon Domain : Sal(x) \geq Sal(r)\}}$$

This concludes our discussion of ways in which the *expressive power* of GRE algorithms can be extended.

Now: a discussion of the problem of **Logical Form Equivalence** (Appelt, Shieber, etc.)