Linguistic aggregation

The issue

Search techniques for aggregation

WHAT IS AGGREGATION

Term "Aggregation" coined by Mann and Moore (1980)

Structural aggregation **– Several logical assertions share information**

- **NL utterance with coordinated or omitted parts**
- **"Foxes and wolves are animals"** ***Conjunction reduction***
- **"Foxes are larger than birds and smaller than wolves"** ***Ellipsis / Gapping***

Conceptual aggregation

- **NL utterance combining the meaning of several ones**
- **"John hits Peter" and "Peter hits John" → "John and Peter fight"**

The role of aggregation

- **Structural aggregation is the dominating factor**
- **Aggregation relevant for virtually every application of NL generation**
- **Not aggregating information is likely to produce awkward and redundant texts**

AGGREGATION

Characterization

Relevant for all phases of generation

Integration in the overall process according to demands of application

Methods – in "historic" order

1980:	**(Mann, Moore)**	**Arbitrary rewrite rules covering a variety of issues**
		No control mechanism offered
1990:	**(Dalianis, Hovy)**	**Opportunistically applied rules for coordination issues**
		Discourse motivated ordering
2000:	**(Shaw)**	**Systematic procedure, mostly for syntactic aggregation**
		For applications with substantial aggregation needs
		Linear complexity, sacrificing optimality

AGGREGATION IN INCREMENTAL GENERATION

Filtering

Relevance and recency filtering (e.g., featuring warnings)

Selection of anaphoric reference of the basis of recency

Building fully specified logical form

Incemental aggregation

Generation starts prior to content specification completion

“Retro-aggregation” - on the basis of the previous utterance

Example:

“I have cancelled flying to the school.

And the tower. And landing at the base.”

SHAW'S AGGREGATION ALGORITHM

Motivation

A set of propositions can be aggregated in a huge number of ways

Some aggregated linguistic expressions may be preferable to others

Cognitive limitations on the complexity of aggregated expressions

Compromise between quality and effort, geared by language properties

Parameterizing aggregation

Obeying coherence requirements - not aggregating across rhetorical relations

Taking into account linguistic idiosyncracies of the target language

Controling the complexity of expressions

BASICS OF SHAW'S ALGORITHM

Aggregation operations and examples

Propositions that differ in *one* argument

“Alice installed *Excel and Latex* for John on Monday”

Propositions that differ in *multiple* arguments

recurring elements deleted in forward or backward direction

“Cindy removed Access [for John] on Monday,
and Bob [removed] Latex for John on Tuesday”

SHAW'S ALGORITHM

4 stages

1. **Group propositions and order them according to similarities**

   Based on the number of distinct values for each argument.

2. **Determine recurring elements in the ordered propositions that will be combined**

   Done incrementally, starting with the first two propositions, etc.

3. **Create a sentence boundary when the combined clause reaches a given threshold**

4. **Determine which recurring elements are redundant and should be deleted**

   Requires grammatical knowledge about the target language

FUNCTIONALITY AND RESULTS (1)

Initial representation

p(install,Alice,Excel,John,Monday)

p(remove,Bob,Word,John,Tuesday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

FUNCTIONALITY AND RESULTS (2)

Initial representation

p(install,Alice,Excel,John,Monday)

p(remove,Bob,Word,John,Tuesday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

Representation after ordering (stage 1)

p(install,Alice,Excel,John,Monday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

p(remove,Bob,Word,John,Tuesday)

FUNCTIONALITY AND RESULTS (3)

Initial representation

p(install,Alice,Excel,John,Monday)

p(remove,Bob,Word,John,Tuesday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

Representation after ordering (stage 1)

p(install,Alice,Excel,John,Monday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

p(remove,Bob,Word,John,Tuesday)

Recurrence markings performed (stage 2 & 3)

p(install^{1},Alice^{1},Excel,John^{1,2},Monday^{1,2})

p(install^{1},Alice^{1},Latex,John^{1,2},Monday^{1,2})

p(remove,Cindy,Access,John^{2},Monday^{2})

p(remove,Bob,Word,John,Tuesday)

FUNCTIONALITY AND RESULTS (4)

Initial representation

p(install,Alice,Excel,John,Monday)

p(remove,Bob,Word,John,Tuesday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

Recurrence markings performed (stage 2 & 3)

p(install^{1},Alice^{1},Excel,John^{1,2},Monday^{1,2})

p(install^{1},Alice^{1},Latex,John^{1,2},Monday^{1,2})

p(remove,Cindy,Access,John^{2},Monday^{2})

p(remove,Bob,Word,John,Tuesday)

Representation after ordering (stage 1)

p(install,Alice,Excel,John,Monday)

p(install,Alice,Latex,John,Monday)

p(remove,Cindy,Access,John,Monday)

p(remove,Bob,Word,John,Tuesday)

Deletions and coordination done (stage 4)

{p(install,Alice,<Excel,

Latex>, – ,Monday),

p(remove,Cindy,Access,John, –)}

p(remove,Bob,Word,John,Tuesday)

FUNCTIONALITY AND RESULTS (5)

Initial representation

p(install,Alice,Excel,John,Monday)
p(remove,Bob,Word,John,Tuesday)
p(install,Alice,Latex,John,Monday)
p(remove,Cindy,Access,John,Monday)

Representation after ordering (stage 1)

p(install,Alice,Excel,John,Monday)
p(install,Alice,Latex,John,Monday)
p(remove,Cindy,Access,John,Monday)
p(remove,Bob,Word,John,Tuesday)

Recurrence markings performed (stage 2 & 3)

p(install^{1},Alice^{1},Excel,John^{1,2},Monday^{1,2})
p(install^{1},Alice^{1},Latex,John^{1,2},Monday^{1,2})
p(remove,Cindy,Access,John^{2},Monday^{2})
p(remove,Bob,Word,John,Tuesday)

Deletions and coordination done (stage 4)

{p(install,Alice,<Excel,
Latex>, – ,Monday),
p(remove,Cindy,Access,John, –)}
p(remove,Bob,Word,John,Tuesday)

"On Monday, Alice installed Excel and Latex and Cindy removed Access for John. Bob removed Word for John on Tuesday"

EXTENSION TO GENERATING SEMANTICALLY COMPLEX OPERATORS

Common characteristics of approaches to date

Typically treated as an *opportunistic* optimization measure

Mostly operating on some local level

Favoring aggregations with multiple joint parts

Demands from results produced by formal reasoning systems

Regularly occurring* commonalities, combinations of *alternatives

Coordination in a *cross product fashion* rather than in a *pairwise fashion*

Making use of semantically complex NL operators

(“each”, “vice-versa”, “remaining”, “distinct”)

SETS OF SOLUTIONS IN MODEL GENERATION

Assignment problem **– jobs to persons**

4 persons for 8 jobs, each person takes 2 jobs, with additional constraints

16 solutions, 128 assertions

be(Pete,boxer,<2,4,5,7,9-11,13-16>)

be(Pete,guard,<1,3,6,8,12,15,16>)

be(Pete,operator,<1-14>)

be(Roberta,actor,<1,2,6,7,11,12,14,16>)

be(Roberta,guard,<4,10,11,13>)

be(Roberta,officer,<1-5,8,9,12-16>)

be(Roberta,teacher,<3,5-10,15>)

be(Steve,actor,<3,5,13>)

be(Steve,boxer,<8,12>)

be(Steve,guard,<9,14>)

be(Steve,nurse,<1-16>)

be(Steve,officer,<6,7,10,11>)

be(Steve,operator,<15,16>)

be(Steve,teacher,<1,2,4>)

be(Thelma,actor,<4,8-10,15>)

be(Thelma,boxer,<1,3,6>)

be(Thelma,chef,<1-16>)

be(Thelma,guard,<2,5,7>)

be(Thelma,teacher,<11-14,16>)

CONSTRUCTS EXPRESSIBLE BY NL OPERATORS

The Permut construct

Comprises all combinations of the values of two slots (a single exception possible)

"Each of Roberta, Steve and Thelma is an actor, a boxer, and a teacher, (but Roberta is not a boxer)."

The Choice and Except constructs

Choice **expresses the assignment of one slot value to several values of another slot (as alternatives) –** ***Except*** **expresses the complement to** ***Choice***

"Thelma holds one of the positions actor, guard, and teacher (and Roberta holds the remaining positions)"

The Assign construct

All bijective functions between two sets of individuals (no repetitions, alternatively)

"Roberta, Steve, and Thelma each holds one distinct job out of the set actor, guard, and teacher"

COORDINATION WITH ONE-SLOT DIFFERENCES

Coord- and Choice constructs

Composing simple propositions into one-slot distinct constructs

***Coord-constructs* (ordinary aggregation)**

1 **be(Thelma,chef)**
be(Thelma,guard)
be(Thelma,boxer)

2 ***Coord*(be,Thelma,<chef,guard>)**
be(Thelma,boxer)

3 ***Coord*(be,Thelma,**
<chef,guard,boxer>)

"Thelma is the chef,
the guard, and the boxer."

***Choice-constructs* (with alteratives)**

1 **be(Thelma,chef,1)**
be(Thelma,guard,2)
be(Thelma,boxer,3)

2 ***Choice*(be,Thelma,<chef,guard>,<1,2>)**
be(Thelma,boxer,3)

3 ***Choice*(be,Thelma,**
<chef,guard,boxer>,<1,2,3>)

"Thelma holds one of the positions
chef, guard, and boxer."

COORDINATION WITH TWO-SLOT DIFFERENCES (2)

Permut-construct

Composing *Coord-constructs* with identical value lists and another distinct slot

At most one value combination may be missing

1 ***Coord*(be,<Roberta>,<actor,guard,teacher>)**
***Coord*(be,<Steve>,<actor,guard,teacher>)**
***Coord*(be,<Thelma>,<guard,teacher>)**

2 ***Permut*(be,<Roberta,Steve>,<actor,guard,teacher>)**
***Coord*(be,<Thelma>,<guard,teacher>)**

3 ***Permut*(be,<Roberta,Steve,Thelma/Thelma>,<actor,guard,teacher/actor>)**

"Each of Roberta, Steve and Thelma is an actor and a guard, and a teacher, but Thelma is not an actor."

COORDINATION WITH DISJUNCTIONS (1)

***Assign-construct* (simple form)**

Composing two *Choice-constructs* with complementing value lists in same variants

***Choice*(be,Roberta,<actor,teacher>,<16,15>)**

***Choice*(be,Thelma,<actor,teacher>,<15,16>)**

***Assign*(be,<Roberta,Thelma>,<actor,teacher>,<15,16>)**

“Roberta is the actor and Thelma the teacher, or vice-versa.”

THE AGGREGATION PROCEDURE

Inspired by Shaw's procedure

1 **Building coordinations with a single difference**

2 **Attempting to express all variants in disjunctions unambiguously**

3 **Split sets of variants, repeat step 2 for each partitioning**

4 **Building coordinations with two differences**

5 **Performing linguistic realization**

Sensitive to ordering

Mostly linear, multiple passes; may not be optimal

Aims at building cross-product-type coordinations, well-suited for few slots

EXTENSIONS TO FORMULAS

Measures meeting particularities of formulas

Ordering criteria based on nesting, number of operators and variables

Permut operators for more than 2 slots

Multiple passes with different orderings

Application – Categorization of sets of residue classes and opertions as algebraic structures

Operations of residue classes modulo 5 (the quasi-groups)

x-y	2*x-y	2*x-(y+1)	3*x-(y+1)	2*x+(y+1)	3*x+(y+1)	(x-y)+1	(2*x)+(3*y)
	3*x-y	2*x-(y+2)	3*x-(y+2)	2*x+(y+2)	3*x+(y+2)	(x-y)+2	(3*x)+(2*y)
	2*x+y	2*x-(y+3)	3*x-(y+3)	2*x+(y+3)	3*x+(y+3)	(x-y)+3	2*(x-y)
	3*x+y	2*x-(y+4)	3*x-(y+4)	2*x+(y+4)	3*x+(y+4)	(x-y)+4	3*(x-y)

Condensed formal expressions and formulas for the above set of operators

Coord(<2,3>,*,x,+,<3,2,*,y)	i*x+(5-i)*y,i=2,3
Coord(x,-,y,+,<0,1,2,3,4>)	x-y[+i], i=1,…,4
Permut(<2,3>,*,x,<+,->,y)	i*(x±y), i=2,3
Permut(<2,3>,*,x,<+,->,y,+,<0,1,2,3,4>)	i*x±y[+j], i=2,3, j=1…4