Trainable Approaches for Surface NLG*

Adwait Ratnaparkhi

WhizBang! Labs -- Research

*Funded by IBM TJ Watson Research Center

What is surface NL generation ?

- Module that produces grammatical NL phrase to describe an input semantic representation
- For our purposes
  - ***what information to say*** is determined elsewhere (deep generation)
  - ***how to say the information*** is determined by NLG systems (surface generation)

Existing Traditional Methods

- Canned Phrases & Templates
  - Simple to implement
  - Scalability is limited
- NLG Packages
  - FUF/SURGE (Columbia Univ.),ILEX (Edinburgh Univ.), PENMAN (ISI), REALPRO (CogenTex), ...
  - Advantages
    - Input: abstract semantic representation
      Output: NLG package turns it into English
  - Disadvantages
    - Requires many rules to map semantics to NL
    - Writing rules, as well as input representation requires linguistic expertise

Trainable NLG

- Motivation
  - Avoid manually writing rules mapping semantics to English
  - Data driven
    - Base NL generation on real data, instead of the preferences of grammar writer
    - Portability to other languages & domains
  - Solve *Lexical Choice* problem : if there are many correct ways to say the same thing, which is the best ?

Trainable NLG for air travel

- Generate noun phrase for a flight description
- Input to NLG: meaning of flight phrase
  - { $air = "USAIR", $city-fr = "Miami", $dep-time = "evening", $city-to = "Boston", $city-stp = "New York" }
- NLG produces: ***$air flight leaving $city-fr in the $dep-time and arriving in $city-to via $city-stp***
- **After substitution: *"USAIR flight leaving Miami in the evening and arriving in Boston via New York"***
- System learns to generate from corpus of (meaning, phrase) pairs, e.g.

Meaning	Phrase
$city-fr $city-to $air	***flight from $city-fr to $city-to on $air***

What is so difficult about generating flight descriptions ?

- Flight phrases are necessary in a dialog response
  - e.g., "There are 5 flights ... , which do you prefer ?"
- Combinatorial explosion of ways to present flight information, i.e., we use 26 attributes
  - Given n attributes, n! possible orderings
- NLG must solve:
  - What is the optimal ordering of attributes ?
  - What words do we use to "glue" together attributes, so that phrase is well-formed?
  - What is the optimal way to choose between multiple ways of saying the same flight, i.e., *lexical choice* ?

Three methods for trainable surface NLG

- NLG1: Baseline model
  - Find most common phrase to express attribute set
  - Surprisingly effective: over 80% accuracy
  - Cannot generate phrases for novel attribute sets
- NLG2: Consecutive n-gram model
  - predict words left-to-right
- NLG3: Dependency based model
  - predict words in dependency tree order (not necessarily left-to-right)

NLG2: n-gram based generation

- Predict sentence, one word at a time
  - Associate a probability with each word
  - Use information in previous 2 words & attributes
  - Simultaneously search many hypotheses
- Probability model for sentence:
  - A = initial attribute list
  - A_i = attributes remaining when predicting i^{th} word
  - $P(w_1 \ldots w_n | A) = \Pi_i \; P(w_i \mid w_{i-1}, w_{i-2}, A_i)$
- NLG2 outputs best sentence W*
  - $W^* = w_1^* \ldots w_n^* = \text{argmax}_{w1 \ldots wn} P(w_1 \ldots w_n \mid A)$

Combine local & non-local information to predict next word

- Implement information in context as features in maximum entropy framework
  - $f_j(w_i\ w_{i-1}\ w_{i-2}\ A_i) =$ 1 if $<w_i\ w_{i-1}\ w_{i-2}\ A_i>$ is *interesting*
    0 otherwise
  - Derive feature set by applying patterns to training data
  - E.g., $f_j(w_i\ w_{i-1}\ w_{i-2}\ A_i) = 1$ if w_i = "from", w_{i-1} = "flights", $\$city\text{-}fr \in A_i$,
    0 otherwise

- $P(w_i \mid w_{i-1}\ w_{i-2}\ A_i) = \Pi_{j=1\ldots k}\ \alpha_j^{f_j(w_i\ w_{i-1}\ w_{i-2}\ A_i)} / Z(w_{i-1}\ w_{i-2}\ A_i)$

- Each feature has a ***weight*** : $\alpha_j > 0$

NLG2 Sample output

- A = { $city-to = "Boston", $day-dep = "Tuesday", $airport-fr = "JFK", $time-depint = "morning" }
- NLG2 produces:

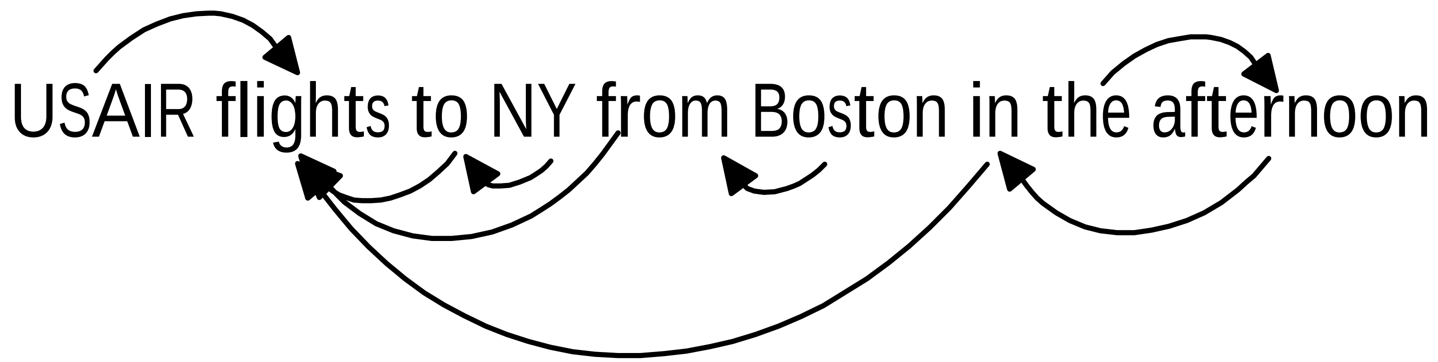
0.137	flights from JFK to Boston on Tuesday morning
0.084	flights from JFK to Boston Tuesday morning
0.023	flights from JFK to Boston leaving Tuesday morning
0.013	flights between JFK and Boston on Tuesday morning
0.002	flights from JFK to Boston Tuesday morning flights

NLG2 Summary

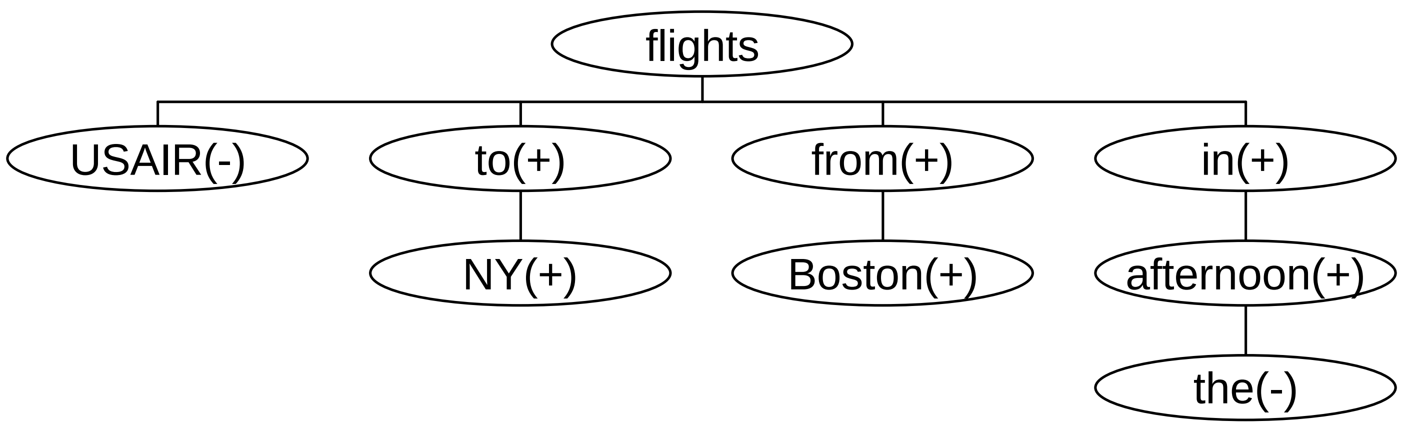
- Advantages
  - Automatic determination of attribute ordering, connecting English, and lexical choice
  - Minimally annotated data
  - 86-88% correct
- Disadvantages
  - Current word is dependent on only previous 2 words
    - May not scale to longer sentences with long distance dependencies
  - Difficult to implement number agreement

NLG3: Predict dependency tree

- Links indicate grammatical dependency



- Links form a tree (+/- indicate direction)



NLG3 Model for Dependency generation

- Testing: given attribute list (A), find most probable dependency tree T*
  - $T^* = \text{argmax}_t \; p(t \mid A)$
  - $p(t|A) = \prod_{child} p(\text{child} \mid \text{parent, grandparent, 2 siblings}, A_{child})$
  - Form of p(child| ...) is maximum entropy model
- Use beam-like search to find T*
- Assumption: easier to predict new words when conditioning on grammatically related words together with attributes

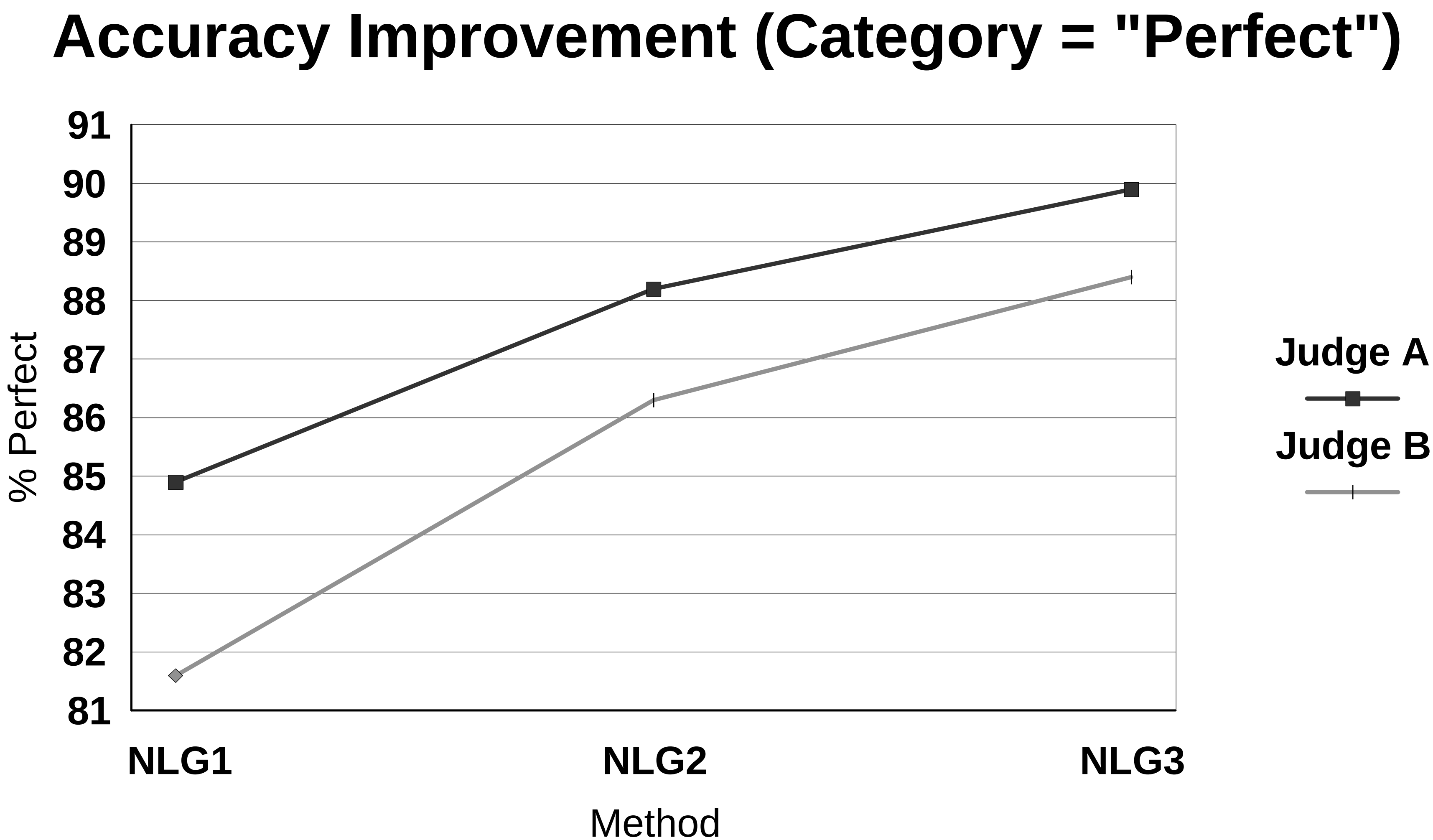
NLG3 Summary

- Automatic determination of attribute ordering, connecting English, and lexical choice
- Annotated data semi-automatically derived from NLU training data
- Easier to implement number agreement
- Should scale to longer sentences with long-distance dependencies
- 88-90% correct on test sentences

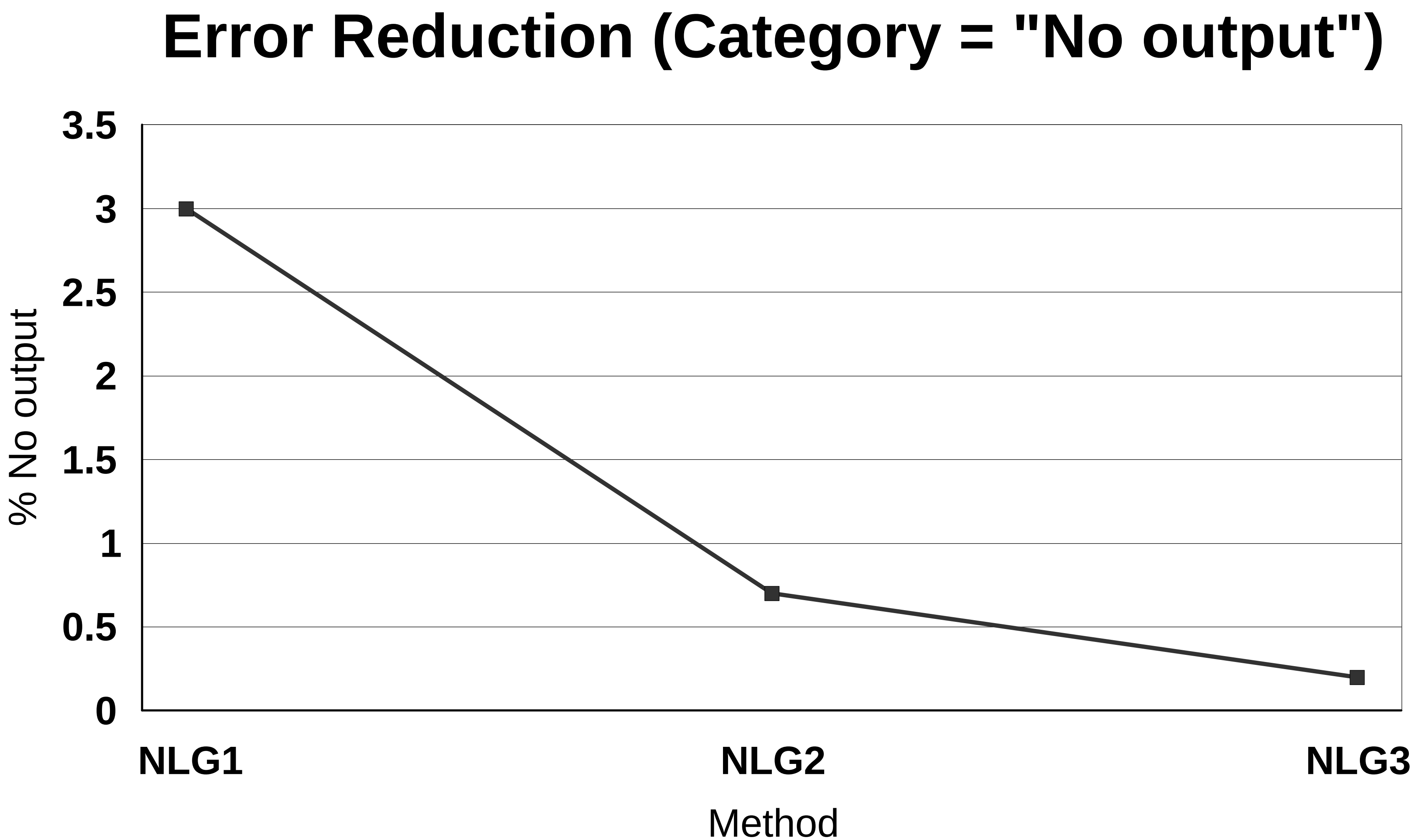
Evaluation

- Training: 6k flight phrases
  - NLG1, NLG2 : train from text only
  - NLG3 : train from text & grammatical dependencies
- Testing: 2k flight phrases
  - test data consists of 190 unique attribute sets
- Evaluate NLG output by hand (2 judges)
  - 1 = perfectly acceptable [Perfect]
  - 2 = acceptable except for tense or agreement [OK]
  - 3 = not acceptable (extra or missing words) [Bad]
  - 4 = no output from NLG [Nothing]

Accuracy improves with more sophisticated methods



Fewer cases of no output with more sophisticated models



Conclusions

- Learning reduces error from baseline system by 33% - 37%
  - attribute ordering,
  - connecting English,
  - lexical choice
- (Langkilde & Knight, 1998) uses corpus statistics to rerank output of hand-written grammar
  - NLG3 can be viewed as inducing a probabilistic dependency grammar
- (Berger et al, 1996) does statistical MT (and hence generation) straight from source text
  - Our systems use a statistical approach with an "interlingua" (attribute-value pairs)