Data-oriented Generation: Statistical Approaches

Presentation by Bettina Fromkorth

In Data Oriented Processing and Generation

SS 2005
Overview

- Introduction
- Ratnaparkhi (2000):
  - Trainable Surface Natural Language Generation (NLG) Systems (generation steps)
  - Three systems (considering Probability Models; Feature Pattern and Selection; Search Procedures)
- Summary of First Paper
- Ratnaparkhi (2001):
  - Hybrid statistical and grammar-based approach
  - Modelling information novelty; application
- Summary and Outlook on Future Work
Introduction

- **Deep NLG:** process of deciding what information to convey (what to say)
- **Surface NLG:** process of rendering that information in natural language (how to say it)
- Easiest way to implement surface NLG: templates: e.g. flight departing from $city-fr at $time-dep and arriving in $city-to at $time-arr
Ratnaparkhi (2000)

- Semantic representation: attribute-value pairs e.g:
  \[
  \{ \textit{city-fr} = \text{New York City}, \textit{city-to} = \text{Seattle}, \textit{time-dep} = 6 \text{ a.m.}, \textit{date-dep} = \text{Wednesday} \}\n  \]

- Requirements on corpora:
  - Corpus marked with domain-specific semantic attributes for NLG1 and NLG2
  - Corpus with semantic attributes and syntactic dependency information
Trainable Surface NLG Systems from Annotated Corpora

Goal: learn the mapping from semantics to words for a text to be generated

• Learn optimal attribute ordering (e.g. “flights to New York in the evening“ or “flights in the evening to New York“)

• Learn optimal lexical choice (e.g. “flights departing to New York“ or “flights leaving to New York“)

• Hereby reflect the observed usage of language in a corpus
Generation Steps

- Training Data (Sample):
  Flights on $air from $city-fr to $city-to the $time-depint of $date-dep
  $trip flights on $air from $city-fr to $city-to leaving after $time-depaft on $date-dep

- Two steps for generation (all three systems):
  - Produce a sequence of words intermixed with attributes
  - Replace the corresponding attributes in the phrase by concrete values
Two steps of NLG process

- **Input step 1**: \{city-fr, city-to, time-dep, date-dep\}
- **Output step 1**: “a flight to $city-to that departs from $city-to at $time-dep on $date-dep“
- **Input step 2**: “a flight to $city-to that departs from $city-to at $time-dep on $date-dep“, \{city-fr = New York City, city-to = Seattle, time-dep = 6 a.m., date-dep = Wednesday\}
- **Output step 2**: “a flight to New York City that departs from $Seattle at 6 a.m on Wednesday“
NLG 1 (baseline)

Chooses most frequent template in training data corresponding to a given set of attributes

$$
\text{nlg}_1(A) \begin{cases} 
\text{argmax}_{\text{phrase} \in T_A} C(\text{phrase}, A) & T_A \neq \emptyset \\
[\text{empty string}] & T_A = \emptyset 
\end{cases}
$$

$nlg_1(A)$: returns phrase corresponding to attribute set $A$

$T_A$: phrases that occurred with $A$ in the training data

$C(\text{phrase}, A)$: training data frequency of NL phrase $\text{phrase}$ and set of Attributes $A$
NLG 2 (n-gram model)

- Individually generating each word in the phrase
- Express given attribute-value set via word sequence with highest probability (all input attributes mentioned exactly once)
- Search Procedure: Search from left to right

\[
p(w_i | w_{i-1}, w_{i-2}, attr_i) = \frac{\prod_{j=1}^{k} \alpha_j f_j(w_i, w_{i-1}, w_{i-2}, attr_i)}{Z(w_i - 1, w_i - 2, attr_i)}
\]

\[
Z(w_i - 1, w_i - 2, attr_i) = \sum_{w'} \prod_{j=1}^{k} \alpha_j f_j(w_i, w_{i-1}, w_{i-2}, attr_i)
\]
# Feature Patterns for NLG 2

<table>
<thead>
<tr>
<th>Description</th>
<th>Feature $f(w_i, w_{i-1}, w_{i-2},\text{ attr}_i) = \ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Attributes remaining</td>
<td>1 if $w_i = ?$ and $\text{attr}_i = {}$, 0 otherwise</td>
</tr>
<tr>
<td>Word bi-gram with attribute</td>
<td>1 if $w_i = ?$ and $w_{i-1} = ?$ and $? \in \text{ attr}_i$, 0 otherwise</td>
</tr>
<tr>
<td>Word tri-gram with attribute</td>
<td>1 if $w_i = ?$ and $w_{i-1} w_{i-2} = ?$ and $? \in \text{ attr}_i$, 0 otherwise</td>
</tr>
</tbody>
</table>
NLG 3: dependency information

- Previous two words are not necessarily the best informantes when predicting the next word
- NLG 3 assumes conditioning on syntactically related words
- Search procedure creates dependency trees from top-to-bottom
## Feature Patterns for NLG 3

<table>
<thead>
<tr>
<th>Description</th>
<th>Feature $f (ch_i(w), ch_{i-1}(w), ch_{i-2}(w), par(w), dir, attr_{w,i}) = ...$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siblings</td>
<td>1 if $ch_i(w) = ?$ and $ch_{i-1}(w) = ?$ and $ch_{i-2}(w) = ?$ and $dir = ?$ and $\notin attr_{w,i}$, 0 otherwise</td>
</tr>
<tr>
<td>Parent + Sibling</td>
<td>1 if $ch_i(w) = ?$ and $ch_{i-1}(w) = ?$ and $w = ?$ and $dir = ?$ and $\notin attr_{w,i}$, 0 otherwise</td>
</tr>
<tr>
<td>Parent + Grandparent</td>
<td>1 if $ch_i(w) = ?$ and $w = ?$ and $par(w) = ?$ and $dir = ?$ and $\notin attr_{w,i}$, 0 otherwise</td>
</tr>
</tbody>
</table>
Search Procedure
(advancing only top N trees)

1. Predict the next left child (call it $x_l$)
2. If it is *stop*, jump to (4)
3. Recursively predict children of $x_l$. Resume from (1)
4. Predict the next right child (call it $x_r$)
5. If it is *stop*, we are done predicting children for current head
6. Recursively predict children of $x_r$. Resume from (4)
## Weighted Evaluations

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>% Correct</th>
<th>% OK</th>
<th>% Bad</th>
<th>% No output</th>
<th>% error reduction from NLG1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLG1</td>
<td>-</td>
<td>84.9</td>
<td>4.9</td>
<td>7.2</td>
<td>3.0</td>
<td>-</td>
</tr>
<tr>
<td>NLG2</td>
<td>N=10, M=30, K=3</td>
<td>88.2</td>
<td>4.7</td>
<td>6.4</td>
<td>0.7</td>
<td>22</td>
</tr>
<tr>
<td>NLG3</td>
<td>N=5, M=30, K=10</td>
<td>89.9</td>
<td>4.4</td>
<td>5.5</td>
<td>0.2</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 5: Weighted evaluation of trainable surface generation systems by judge A

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>% Correct</th>
<th>% OK</th>
<th>% Bad</th>
<th>% No output</th>
<th>% error reduction from NLG1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLG1</td>
<td>-</td>
<td>81.6</td>
<td>8.4</td>
<td>7.0</td>
<td>3.0</td>
<td>-</td>
</tr>
<tr>
<td>NLG2</td>
<td>N=10, M=30, K=3</td>
<td>86.3</td>
<td>5.8</td>
<td>7.2</td>
<td>0.7</td>
<td>26</td>
</tr>
<tr>
<td>NLG3</td>
<td>N=5, M=30, K=10</td>
<td>88.4</td>
<td>4.0</td>
<td>7.4</td>
<td>0.2</td>
<td>37</td>
</tr>
</tbody>
</table>

Table 6: Weighted evaluation of trainable surface generation systems by judge B
Discussion: Ratnarparkhi (2000)

- NLG 2 and NLG 3 systems automatically attempt to generalize from knowledge inherent in training corpus of templates
- Templates for novel attribute sets can be generated
- Grammar-based approaches need to be manually encoded
Discussion: Ratnarparthi (2000)

(2)

- Trainable approaches avoid expense of crafting a grammar to determine attribute ordering and lexical choice

- But: they are less accurate than grammar-based approaches (even though feature patterns, search parameters and training data can be tuned)
Discussion: Ratnarparkhi (2000) limitations of statistical systems

• Simple complexity works (length of phrase fragment describing attribute short; not too many ambiguities) → CPU time and memory costs not too high

• Semantic annotation scheme rich enough to accurately represent meaning in domain; simple enough to contain useful corpus statistics
Ratnarparkhi (2001)

- One system: hybrid statistical and grammar based system using grammar rules, conditions on the rules, and corpus statistics to determine word order
- Prototype conversational system uses surface NLG module to express new information differently than old one based on run-time dialog state (variations of word order)
Hybrid Statistical and Grammar Based System

- Surface NLG module given in dependency-like grammar (similar to NLG 3)
- Many word sequences consistent with grammar rules and rule conditions are created
- Module uses corpus statistics to find word sequence that most resembles real utterances of people
System Procedure

- **Dialog manager** decides that something needs to be uttered to the user
- Input to NLG module extracted from **dialog state**:
  - Mandatory set of attribute-value pairs, $A_1$
  - Optional set of attribute-value pairs, $A_2$
- Deep generation model determines what is contained in $A_1$ and $A_2$
- Attributes as described in other paper
Grammar

- Grammar rules define possible dependency trees the NLG module may generate in the context of the current dialog state.
- Relationship between parent, one or more children:
  - **Parent**: usually linguistic “head“ of phrase
  - **Direction**: - (left) or + (right) intended word order of children relative to parent
  - **Children**: One or more words (children to parent)
  - **Condition**: Code fragment evaluating to true or false depending on current state of dialog system
Example Grammar

<table>
<thead>
<tr>
<th>Par</th>
<th>Dir</th>
<th>Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>+</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>+</td>
<td>c d</td>
</tr>
<tr>
<td>c</td>
<td>-</td>
<td>f</td>
</tr>
</tbody>
</table>

- No ordering constraint between children of different rules with the same head
- Siblings (specified in the same rule) cannot be re-ordered or broken up with respect to each other
- Children are recursively expanded
Allowing for Conjunction and Commas

- Constructs "&" and "|
- e.g. Grammar generates:
  - a b;
  - a b and C;
  - ab, c, and d

<table>
<thead>
<tr>
<th>Par</th>
<th>Dir</th>
<th>Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>+&amp;</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>+&amp;</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>+&amp;</td>
<td>d</td>
</tr>
</tbody>
</table>

- Comma notation is dependent on application
- Generation of morphology not treated explicitly (generic token rewriting facility)
Assigning Scores

\[ P (w_1 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1} \ldots w_n) \]

\[ P(w_i \mid w_{i-1} \ldots w_n) = \sum_{j=1}^{4} \alpha_j P_j (w_i \mid w_{i-1} \ldots w_n) \]

- Probability models \( P \) are computed from statistics derived from about 8000 utterances
- \( P_1, P_2, \) and \( P_3 \) are derived from trigram, bigram, and unigram; \( P_4 \) is the uniform model
Searching Procedure (top N trees)

- If t is A-complete, remove it from consideration. If N trees are A-completed, terminate search.
- Else (t not A-complete), system determines active parent starting at root, recursively checking left children, right children, parent itself, for the first tree node that is not completed.
Searching Procedure

(2)

- Tree node completed if:
  - It is left-complete: all its left children have been generated, and
  - It is right-complete
- If no active parent is found: discard t
- If p is the active parent, system first works in left direction, if p is not left-complete else it works in right direction
- Either apply rule or mark tree (following slides)
Applying a Rule in a Given Direction

• Apply a rule \( r \) if
  – Parent specified in is equal to active parent \( p \)
  – Condition of \( r \) evaluates to true
  – \( r \) has not been previously used to generate children for parent \( p \)
  – Attributes mentioned in children have not been mentioned elsewhere in tree

• If rule applicable, add children in \( r \) to active parent
• Use new tree \( t' \) for consideration in next iteration
Marking of tree

• If one doesn’t apply rule: mark tree
  – Left-complete, if we were adding in left direction
  – Right-complete, if were adding in right direction
  – Use new tree for consideration in next iteration
Information Novelty

• Utterances have an information structure: one part refers to existing information in discourse, other part refers to newly introduced information (theme vs. rheme; topic vs. comment; presupposed vs. focus)

• In air domain if both types of information are contained old information (theme) usually preceds new information (rheme)
Information Novelty (2)

- Here: model informational novelty correlating roughly with theme (old information) vs. Rheme (new information) distinction;
- No difference in pitch is treated here;
- Old information must be grounded (here implicitly by stating all specified values as theme)
Application

- Telephony conversational system for air travel
- Most utterances are generated using existing template-based approach, a certain class of utterances are generated with NLG system described here
- Conversational System collects information from user and then conducts search on flight database
- In our system word order depends on: Grammar, Statistics, Attribute, Novelty
Application (2)

- Deep generation component marks attributes as old or new
- Surface NLG module allows us to detect the novelty of an attribute in any of the rule condition of the grammar
- Grammar for summary sentence is written to produce sentences having structure: There are N flights [old information] that [new information]
- Until now: truely novel information is not treated only novelty introtuced by user (previous turn)
Summary

- System for surface NLG using grammar rules, rule conditions and statistical information to decide word order at run time
- System takes template fragments and attempts to paste them together consistent with grammar and optimal with respect to scoring function.
- System integrated in conversational system for air travel
- Attempt to model Information Structure aspect (novelty)
Summary (2)

- Hybrid surface NLG module not a general purpose generation package
- Work within limited (small) domain: not much linguistic expertise required making this approach more practical (grammar rules easy to write)
Outlook on Future Work

• Plans to extend the framework with facilities for conversational system: passing semantic “features“ from parent to child, interface with morphological database (agreement, inflection)

• Evaluation in context of entire conversational system
Thank you for your attention!
References


Points for Discussion

NLG 3 had dependency information and semantic annotation in Corpus while in the second paper we had dependency information via grammar

• Where is the big difference between having the dependency annotations in a Corpus and specified as a grammar?

• If Information Structure Information would be available in Corpus would the system be able to learn them as well?