Natural Language Generation

ART on Dialogue Models and Dialogue Systems

François Mairesse
University of Sheffield
F.Mairesse@sheffield.ac.uk
http://www.dcs.shef.ac.uk/~francois

Overview

- NLG: what it is? what does it do?
- Template-based generation (canned text)
- Rule-based generation
- Trainable NLG

Some applications

- Simple report/letter writing
  - WeatherReporter: textual weather reports
  - STOP: personalised smoking-cessation letters
  - ModelExplainer: UML diagrams description for software development
- Question answering about knowledge bases
- Automated summarization of text
- Machine translation
- Dialogue systems

Inputs to a generator

- Content plan
  - Meaning representation of what to communicate
    - E.g. describe a particular restaurant
- Knowledge base
    - E.g. database of restaurants
- User model
  - Imposes constraints on output utterance
    - E.g. user wants short utterances
- Dialogue history
    - E.g. to avoid repetitions, referring expressions

Natural language generation objectives

- From a meaning representation of what to say
  - E.g. entities described by features in an ontology
  - E.g. has(WokThisWay, cuisine(bad))
- Output: a natural language string describing the input
  - E.g. "WokThisWay's food is *awful*"
- Desirable properties
  - Simple to use
  - Able to generate well-formed, human-like sentences
  - Trainable? Able to learn?
  - Variation in the output?

Template-based generation

- Most common technique in spoken language generation
- In simplest form, words fill in slots:

  "Flights from SRC to DEST on DATE. One moment please."
- Most common sort of NLG found in commercial systems
- Used in conjunction with concatenative TTS to make natural-sounding output

Template-based generation: Pros & Cons

- Pros
  - Conceptually simple
    - No specialized knowledge needed to develop
  - Tailored to the domain, so often good quality
- Cons
  - Lacks generality
    - Repeatedly encode linguistic rules (e.g., subject-verb agreement)
  - Little variation in style
  - Difficult to grow/maintain
    - Each new utterance must be added by hand

Enhance template generation

- Templates can be expanded/replaced to contain information needed to generate more complex utterances

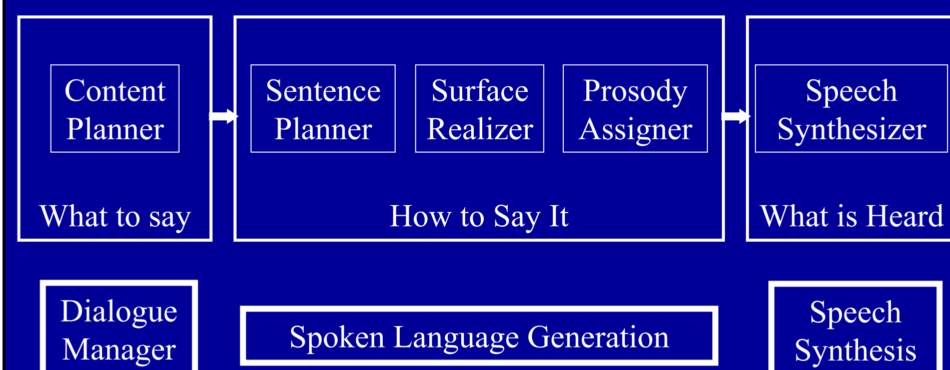
→ Need deeper utterance representations

→ Need linguistic rules to manipulate them

Components of a rule-based generator

- **Content planning**
  - What information must be communicated?
    - Content selection and ordering
- **Sentence planning**
  - What words and syntactic constructions will be used for describing the content?
    - Aggregation
      - What elements can be grouped together for more natural-sounding, succinct output?
    - Lexicalization
      - What words are used to express the various entities?
- **Realization**
  - How is it all combined into a sentence that is syntactically and morphologically correct?
- **Prosody assignment** (spoken language generation only)
  - How to produce appropriate speech based on the previous levels of representation?

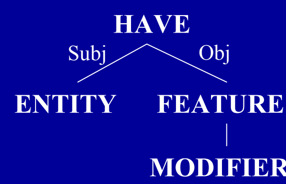
Spoken language generation: pipeline architecture



Example

- Output from dialogue manager
  - Two assertions

    has(WokThisWay, cuisine(bad))
    has(WokThisWay, decor(good))
- Content planning
  - Select information ordering
- Sentence planning
  - Choose syntactic templates →



  - Choose lexicon
    - bad → awful; cuisine → food quality
    - good → excellent; decor → décor
  - Aggregate the two proposition by merging objects
  - Generate referring expressions
    - ENTITY → this restaurant

Example (continued)

- Realization
  - Choose correct verb inflection: HAVE → has
  - No article needed for feature names
  - Convert sentence representation into a final string
  - Capitalize first letter and insert punctuation
- Prosody assignment
  - Standard pitch for an assertion
  - Emphasize user preference for food quality by increasing the voice intensity for modifier “awful”

→ “This restaurant has *awful* food quality but excellent decor.”

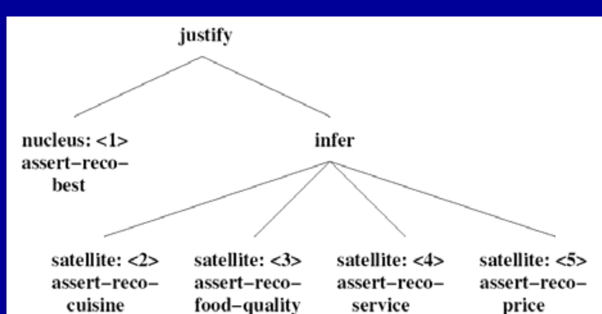
Content planning

- Typically look at spoken/textual data to characterize how information is
    - Selected
    - Ordered
    - Combined together

- A content planner will take a meaning representation and produce a content plan tree
    - Leaves are bits of information
    - Internal nodes are rhetorical relations (Mann & Thompson, 1988)
        - E.g. justification, contrast, inference, etc.

Example content plan tree

- For a restaurant recommendation
- Each leaf is associated with a syntactic template



Sentence planning

- Three main tasks
  - Lexicalisation
    - Many ways to express entities and rhetorical relations
      - E.g. Justify(X,Y) → "X because Y"
        → "X since Y"
    - Typically a domain lexeme database to avoid any misunderstanding
      - E.g. CUISINE → "food"
  - Aggregation
  - Referring expression generation

Sentence planning: aggregation

- Produces a shorter utterances and dialogues, but adds complexity
- Simple: combine two sentences using a conjunction
- Merge two sentences with same subject or same object
  - E.g. "The pizza is warm" + "The pizza is tasty"
  → "The pizza is warm and tasty"
  - E.g. "John bought a TV" "Sam bought a TV"
  → DOESN'T ALWAYS WORK!
- Syntactic embedding
  - E.g. "The pizza is warm" + "I'm eating the pizza"
  → "The pizza that I'm eating is warm"
  → "I'm eating the warm pizza"

Sentence planning: referring expressions

- How to refer to an entity?
  - "The Frog & Parrot", "The pub", "It", etc.
  - Need to know if initial reference → dialogue history
  - Pronominalization algorithm
    - Trade-off between missed pronouns and inappropriate pronouns
      - Pronominalize all entities previously mentioned? No! Need to check for ambiguities, if entity with same person, gender and number was mentioned

Pipeline architecture

- Advantages
  - Modularity
    - Helps managing complexity
    - Components can be improved independently
- However
  - Lower level components can't influence higher level generation decisions
    - E.g. if the utterance's length needs to be controlled
      - Content and sentence planning decisions need to be influenced by the realizer
  - Many other research systems, but harder to maintain and scale up
  - Do humans use a pipeline?

Question

- If you had to build a dialogue system, which approach would you choose for your NLG component (between templates and more complex linguistic rules) and why? Feel free to choose a particular domain to support your case.

Trainable NLG

Making NLG trainable

- What does it mean?
  - Produce better language *automatically* by looking at a collection of existing texts
- Why?
  - Make it less domain dependent
    - Different sources of data for different domain
  - Produce more complex utterances
    - Requires less linguistic expertise
    - Idioms can't be produced by rules
    - E.g. "This restaurant's food is to die for"
    - E.g. "The service will make you want to kill yourself"

Making NLG trainable

- How?
  - Overgenerate and rank
    - Produce various candidate utterances
      - Rule-based
    - Use a statistical model to rank them
      - Function assigning a score to utterances
      - Typically learned based on textual data
    - Pro
      - Initial generation can be imperfect
        - » Conflicts between generation choices
    - Cons
      - Usually high number of utterances to choose from
      - Can be hard to extract good model from data

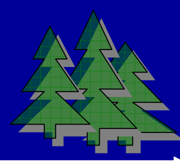
HALogen: combining rules with statistical language models

(Langkilde-Geary, 2002)

Input

Symbolic Generator

- Mapping rules
- Dictionaries
- Morphology



Packed set of expressions

Statistical Ranker

- Ngram model based on 250 million words of newspaper text

Output

Example input format

```
(a1 / |conform,adapt|
    :AGENT (n1 / NONHUMAN-ANIMAL)
    :REASON (c1 / |alter>verbify|
                :GPI (e1 / |environs|)))
```

Example Input and Output

```
(a1 / |conform,adapt|
    :AGENT (n1 / NONHUMAN-ANIMAL)
    :REASON (c1 / |alter>verbify|
                :GPI (e1 / |environs|)))
```

Not-so-ideal:

- Beasts are adjusting because of a surround's alteration.
- Faunas conformed due to alteratia of environs.
- Because of changing of surroundings, creature adapts.

Ideal:

- The animals adapted because of environmental changes.

Recasting input for surface-level syntax

```
( / "serve"
  :voice passive
  :logical-object <cuisine>
  :logical-subject <venue>)
```

```
( / "serve"
  :voice passive
  :logical-object <cuisine>
  :postmod ( / <venue>
             :anchor by ))
```

IF top level contains ***logical-subject***, and also contains ***voice=passive***, **THEN** map ***logical-subject*** to ***postmod***, and add ***anchor=by.***

```
( / "serve"
  :voice passive
  :subject <cuisine>
  :postmod ( / <venue>
             :anchor by))
```

"<Cuisine> is served by <venue>."

Symbolic Generator

- Mapping rules (about 255 rules)
  1. Recast one input to another
  2. Add missing information to under-specified inputs
  3. Assign linear order to constituents
  4. Apply functions, such as morphological inflection
- Dictionaries
  A. Sensus dictionary, based on WordNet
     - (~100,000 words and concepts)
  B. Closed-class lexicon
  C. User-defined dictionary
- Morphology rules

Using statistical language model to prune choices

- How to select the best alternative?
  - Estimate the probability of occurrence based on a corpus: n-gram language models
    - Estimates the probability of a sentence, by counting words in a corpus

$$P(t) = P(w_1 w_2 \ldots w_n) = P(w_1)P(w_1|w_2)P(w_n|w_1,\ldots,w_{n-1}) = \prod_{i=1}^{n} P(w_i|w_1 \ldots w_{i-1})$$

    - Markov assumption: probability of a word does only depend on the n previous words

$$P(w_i|w_1 \ldots w_{i-1}) \approx P(w_i|w_{i-1}) = \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$$

N gram examples

- Bigram model (n = 2)

$$P(\text{I like drinking beer when I am not drunk}) \approx P(\text{I})P(\text{like}|\text{I})P(\text{drinking}|\text{like})P(\text{beer}|\text{drinking}) P(\text{when}|\text{beer})P(\text{am}|\text{I})P(\text{not}|\text{am})P(\text{drunk}|\text{not})$$

- Trigram (n = 3)

$$P(\text{I like drinking beer when I am not drunk}) \approx P(\text{I})P(\text{like}|\text{I})P(\text{drinking}|\text{I},\text{like})P(\text{beer}|\text{like},\text{drinking}) P(\text{when}|\text{drinking},\text{beer})P(\text{I}|\text{beer},\text{when})P(\text{am}|\text{when},\text{I}) P(\text{not}|\text{I},\text{am})P(\text{drunk}|\text{am},\text{not})$$

Computing N-grams

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Slightly more complicated to deal with zeros (interpolation)

How well do n-grams make linguistic decisions?

<u>Relative pronoun</u>
visitor who 9 visitors who 20
visitor which 0 visitors which 0
visitor that 9 visitors that 14

<u>Preposition (bigrams)</u>
in Japan 5413 to Japan 1196
came into 244 arrived into 0
came to 2443 arrived in 544
came in 1498 arrived to 35

<u>Preposition (trigrams)</u>
came to Japan 7 arrived to Japan 0
came into Japan 1 arrived into Japan 0
came in Japan 0 arrived in Japan 4

<u>Word Choice/Singular vs Plural</u>
reliance 567 reliances 0
trust 6100 trusts 1083

How well does HALogen work?

- <u>Minimally specified input frame (bigram model):</u>
  It would sell its fleet age of Boeing Co. 707s because of maintenance costs increase the company announced earlier.
- <u>Minimally specified input frame (trigram model):</u>
  The company earlier announced it would sell its fleet age of Boeing Co. 707s because of the increase maintenance costs.
- <u>Almost fully specified input frame:</u>
  Earlier the company announced it would sell its aging fleet of Boeing Co. 707s because of increased maintenance costs.

N-gram modeling limitations

→ Higher n produces better results, but less data to estimate probability correctly!

→ Highly dependent on the source of text (newspaper articles)
- Spoken language?

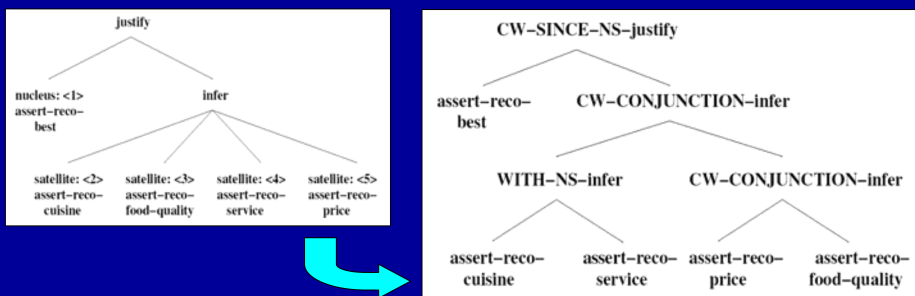
→ N gram will never model deep relations in a sentence, like correct pronouns or distant subject-verb agreement
- E.g. *The restaurant which … has …*

SPoT/SParKY: A trainable generator with deeper linguistic features

(Walker et al. 2002)

Stochastic generation

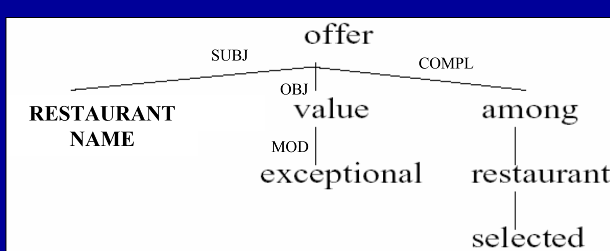
- Randomly generate sentence plan trees from a content plan tree
  - Map rhetorical relations to clause combining operations
    E.g. justification → *since*, *because*
    inference → conjunction, period, merge
  - Nodes are ordered



Generating Sentence Plans

- How to express each information?

→ Database of **Deep Syntactic Structures** (DSyntS, similar to parse trees)



- Operations combine DSyntS's into larger DSyntS's

Stochastic generation

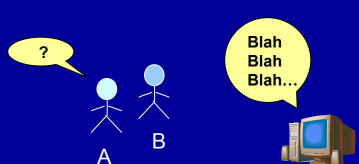
- Last step: realization of each alternative
  - RealPro (Lavoie and Rambow 97)
    - Combines the syntactic structure (DSyntS) into a surface sentence, using rules of English (e.g. agreement)

"WokThisWay has the best overall quality among the selected restaurants since it is a Chinese restaurant, with good service, its price is 24 pounds, and it has good food quality."

"WokThisWay is a Chinese restaurant, with good food quality. It has good service. Its price is 24 pounds. It has the best overall quality among the selected restaurants."

Trainable sentence ranking

- Training the ranker
  - Training data: user ratings of sentences

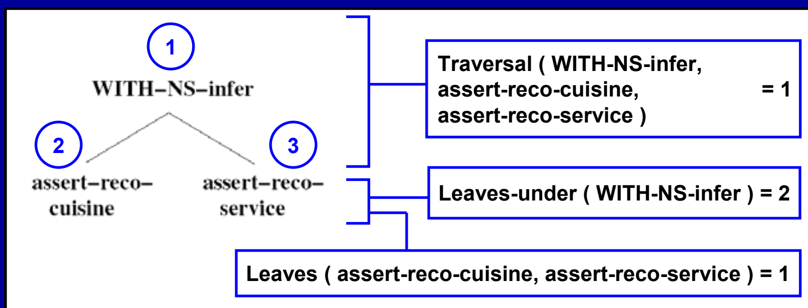


	User A	User B
Sentence 1	4/5	5/5
Sentence 2	3/5	1/5
Sentence 3	2/5	3/5
…		

  - Learning algorithm: RankBoost (Freund et al. 98)
    - Non linear function approximation algorithm, in which the function ranks its arguments
  - Generalizes user ratings for any new sentence
    - Compute ranking score

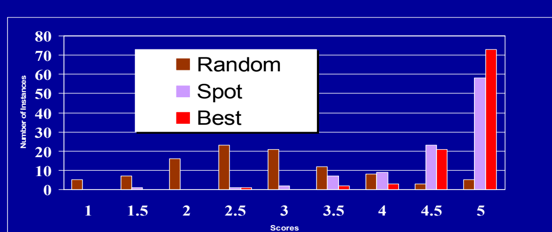
Trainable sentence ranking

- Want to learn user preferences, but how to represent each sentence plan tree in a finite way?
- Associate features to each alternative tree
  - Node counts of sentence plan tree



Evaluation Goals

- Major problem: not clear that quality is good enough for real systems
- Training evaluation: shows that the learning algorithm (RankBoost) did a good job learning from judges' feedback
  - Compare the human score of the highest ranked alternative with the best alternative chosen by the judges

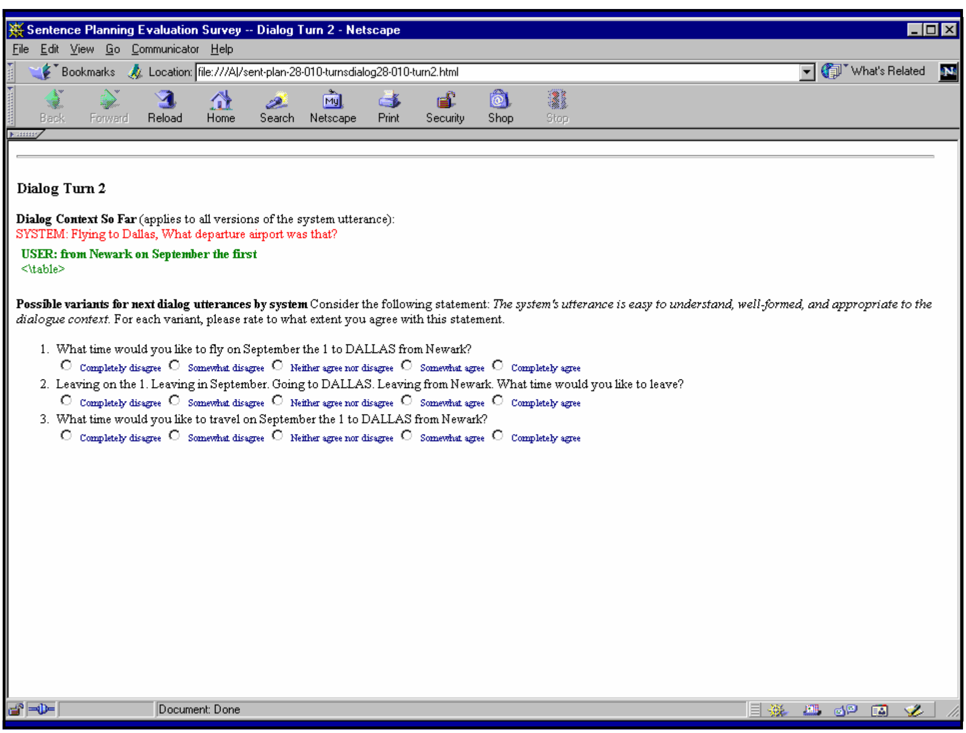


- But doesn't show
  - That the output quality is good (for real people)
  - How the output quality compares to rule-based approaches or template approaches

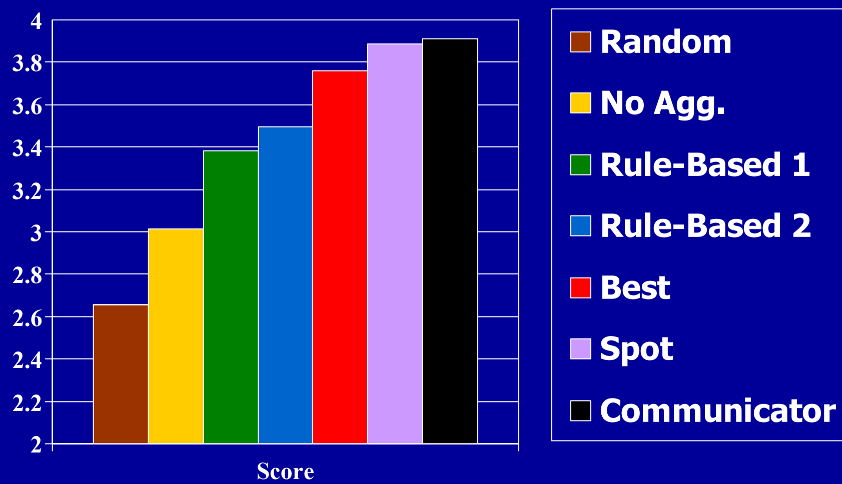
Evaluation Experiment

- 60 subjects compared
  - 7 generators
  - On outputs for 20 text plans
  - Provided subjective rating on 1..5 scale

- Communicator: Template based generator
- SPoT: Trainable sentence planner
- Two Rule-based
- Two Baseline: No Aggregation, Random
- Best: human selection from Random



Results of Evaluation (60 Subjects)



Results of Evaluation

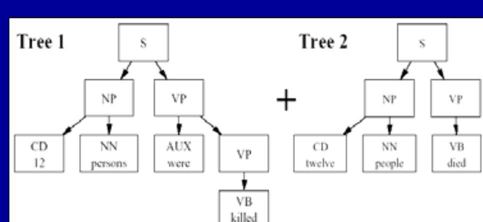
- Random worst, no aggregation second worst
- Rule-based systems scored in medium-range
- SPoT and template-based score equally well
- But SPoT was trained for this domain in days, template-based developed over ~ 2 years!

Linguistic Variation

- Use different models for ranking
  - E.g. n-gram models computed on texts with different style
    - Problem favor ‘average’ style
- A lot of variation is *idiomatic*
  - *E.g. breaking the ice, beating around the bush*
- Stored in human memory?
- Paraphrasing problem
  - Map a meaning representation to multiple realizations
  - Major problem: not much data available!

Paraphrase acquisition

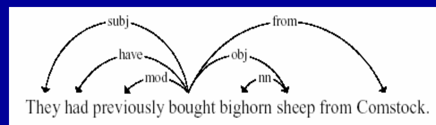
- With a sentence-aligned corpus
  - Merge nodes of parse trees together recursively (Pang et al., 03)



  - Many false paraphrases
  - Sentence-aligned corpus hard to obtain, even more for spoken dialogues

Paraphrase acquisition

- Without aligned corpus: DIRT (Lin & Pantel, 2001)
  - Compute paths in parse trees
    - Leaves are arguments
    - E.g   N:subj:V←buy→V:from:N
      X buys something from Y



  - Based on the **Distributional Hypothesis**:
    If two paths tend to occur in similar contexts,
    the meanings of the paths tend to be similar
  - For each pair of paths, compute a similarity measure based on the number of occurrences with identical arguments
  - Resulting paraphrases are very noisy, produces antonym phrases

→ **Still lot of work to be done!**

Conclusion

- Complex dialogue needs NLG
- Template are simple to implement and produce good results for a very small domain and inflexible dialogues
- Rule-based NLG allows you to produce richer utterances, but still highly domain dependent
- Machine-learning viable alternative to hand-crafting in NLG, probably only option for systems with large generation capabilities

Annotated Bibliography Feedback

- Biggest concern: papers that didn't seem relevant
  - A SDS has many different components—no need to discuss all in focused paper
  - Topics have accepted definitions—reinforcement learning, for example, is a specific type of machine learning
- Another concern: topic range too broad
  - This is a relatively short paper—topics need to be focused

Sample paper outline

- Provide overview of field
  - Why is it important?
    - E.g., automatic learning of dialogue strategies makes it possible to develop systems more quickly
  - Why are you interested in it?
    - Frustration with using a particular system makes you realize how user modelling could help
- Mention focus of paper
  - Specific aspect of topic
    - User modelling in dialogue systems
    - Reinforcement learning to optimize user satisfaction

Sample paper outline (cont'd.)

- Mention specific implementations
  - Reinforcement learning using simulated users vs. user feedback
  - Dynamic user modelling/user modelling based on multi-attribute decision theory
- Evaluation metrics applied to topic
  - How they are applied
    - Automatic/user feedback/none
  - What they measure
    - Automatically derived correlate to satisfaction/user satisfaction
  - Sample results

Sample paper outline

- Conclusion
  - What you think of topic
    - Importance
    - Relevance
    - Potential for use in real-world SDS