NPFL099 - Statistical dialogue systems

User simulation

Filip Jurčíček

Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic

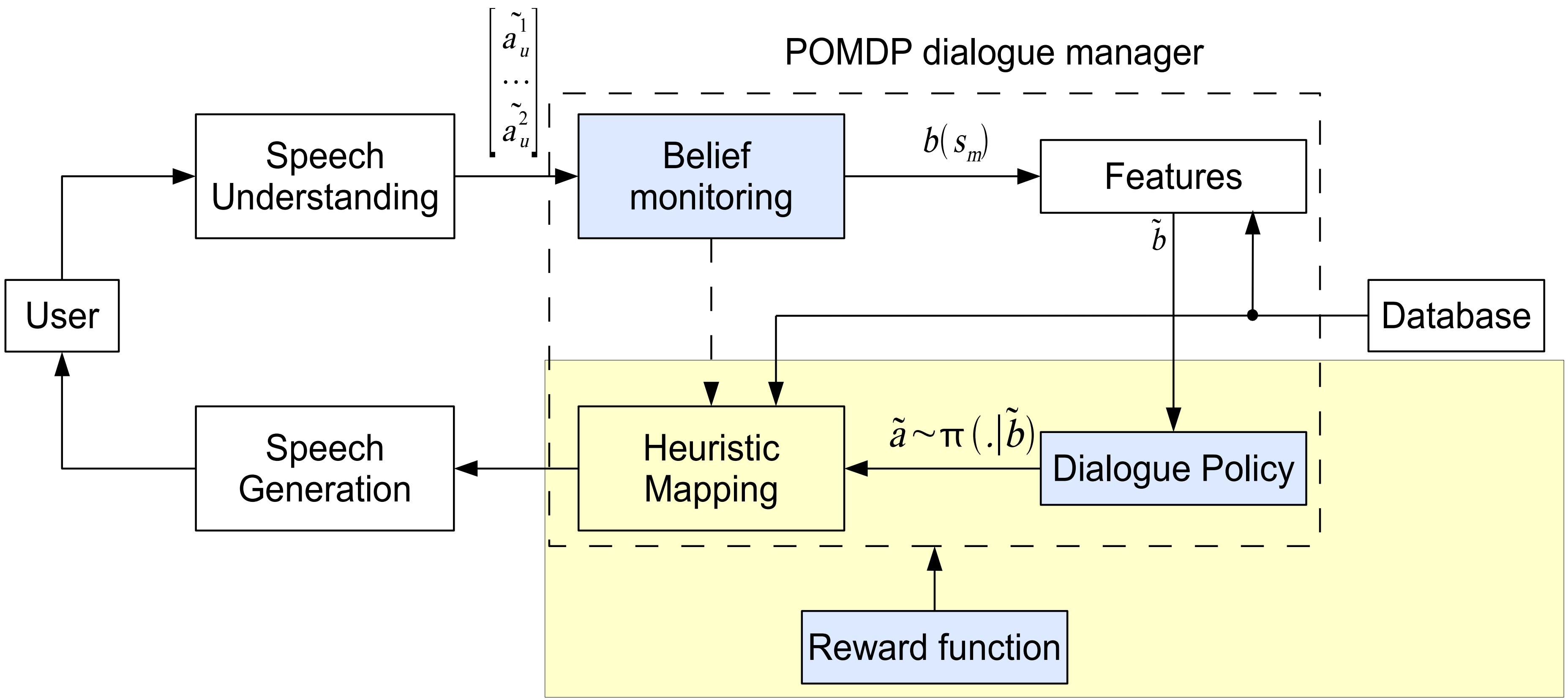
Home page: http://ufal.mff.cuni.cz/~jurcicek

Version: 14/05/2013

Outline

- User simulation
- Dialogue act level simulations
  - N-gram dialogue act model
  - Agenda based simulation
  - ISU based approach
  - Bayesian user simulation
- Word level simulation
- Speech level simulation

POMDP Dialogue systems



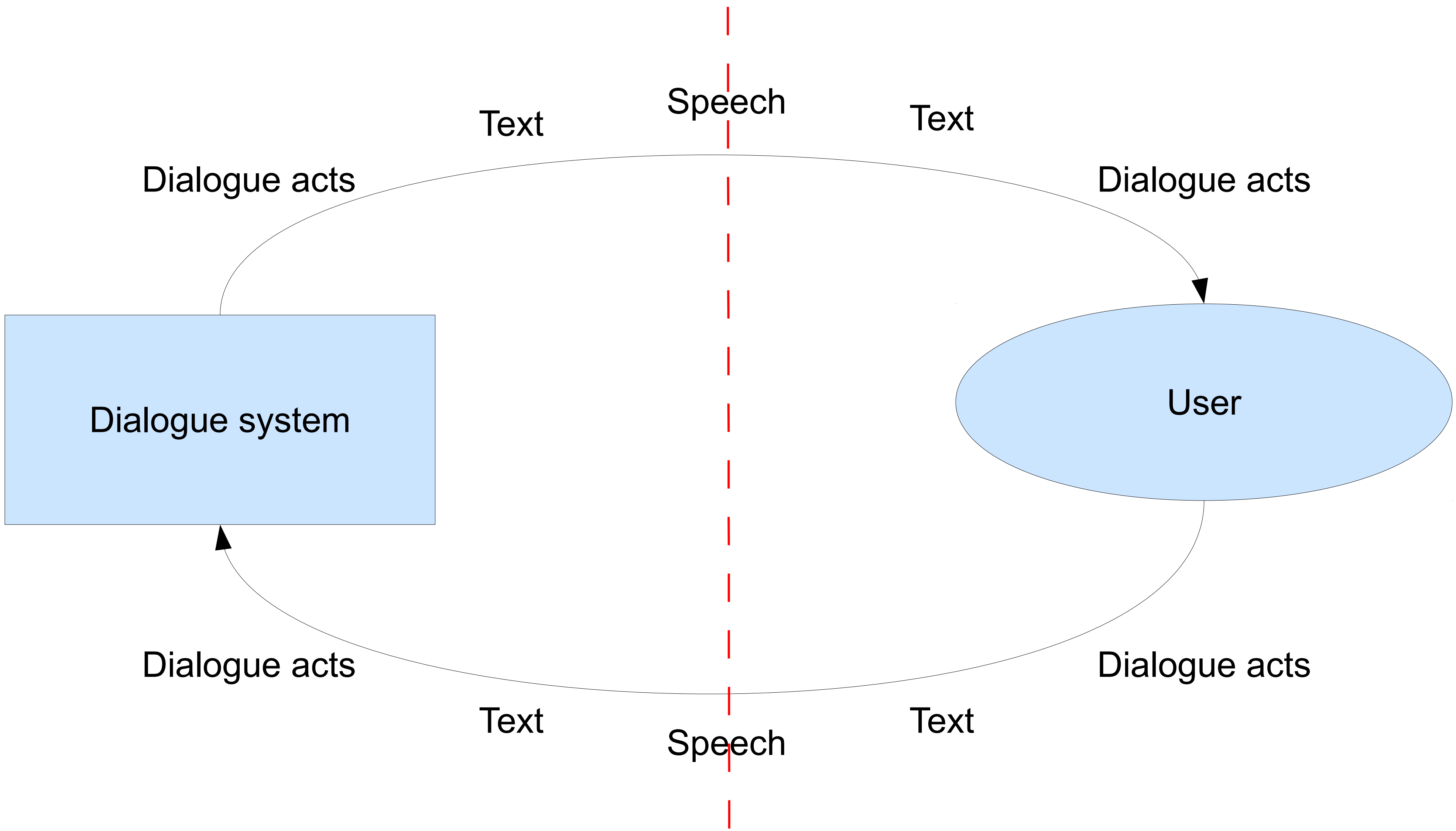
- Ideally the policy is optimized to maximize the reward function

User simulation

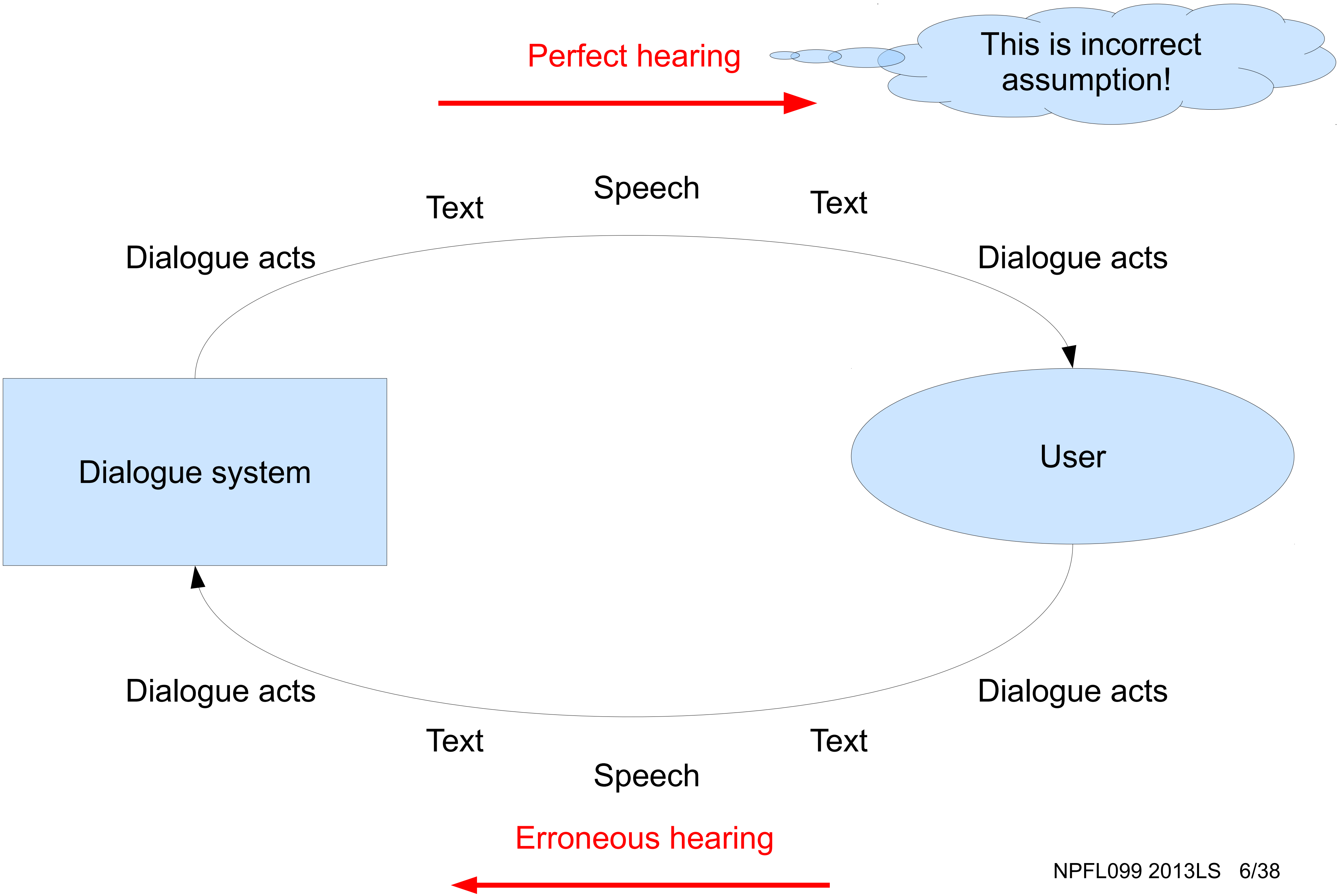
- Ideally, the POMDP dialogue systems would be optimised in interaction with real users
- The problem
  - state-of-the-art techniques still needs to more than 10000 dialogues
- User simulators are used to train and test POMDP techniques
- User simulators are mostly hand-crafted, though parametrised

User simulation

- It is like building another SDS

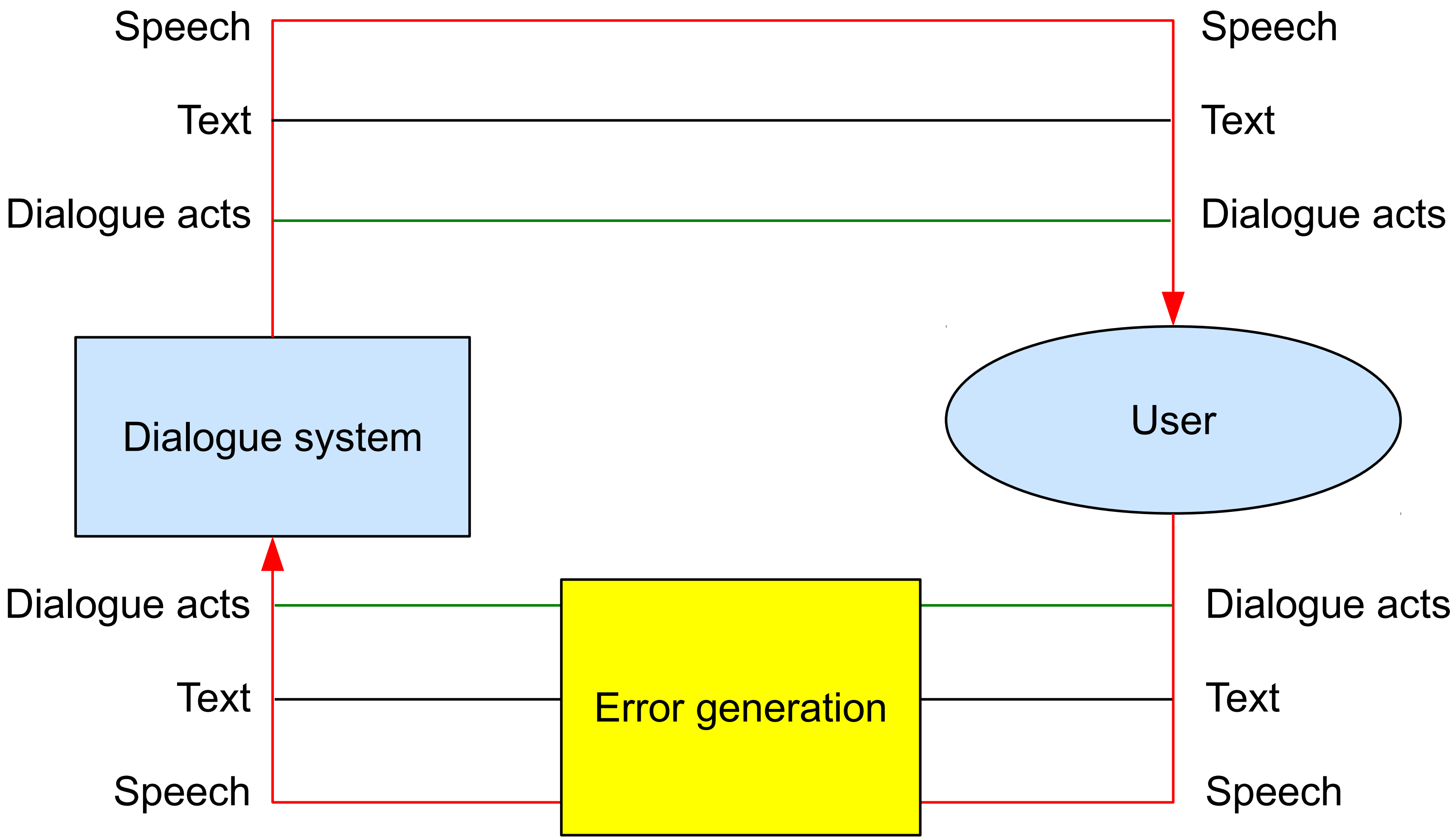


Error simulation



Types of user simulation

- Dialogue act level, Word level, Speech level

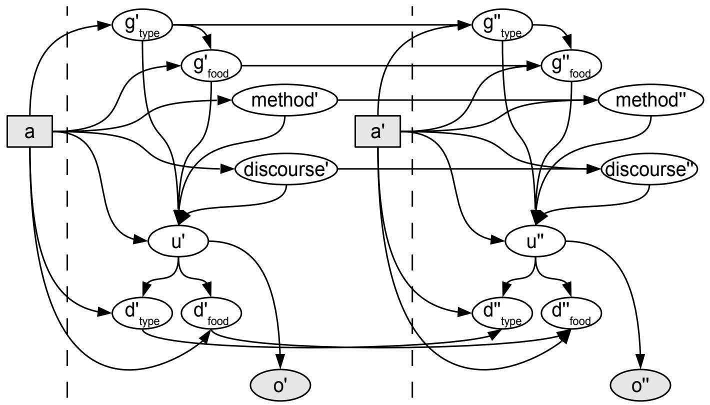


Dialogue act level simulation

- Dialogue act level
  - S: request(food_type)
  - U: inform(food_type=Chinese)

- Many different implementations
  - bigram model for dialogue act types and random sampling slots
  - agenda based simulator
  - using user model to sample user dialogue acts

- Output dialogue act confusion model



N-gram level user simulation

- Dialogue act level simulation
- Predict the next dialogue act given some context

$$P\left(d \middle| \ldots\right)$$

- Sample from a distribution
  - sampling insures variability in the output
- Typical context:
  - None – unigram models
  - Previous dialogue act – bigram models

N-gram dialogue act user simulation

- In general, $P(d|...)$ is too complex to model explicitly

- Model dialogue act type independently of the slots

$$P(dat|...) \approx P(dat_{t+1}|dat_t) \approx P(dat_{t+1})$$

request → inform
inform → inform
inform → deny
inform → confirm
….

- Slots names depends on DAT

$$P(sn_{t+1}|dat_{t+1})$$

  - or are drawn from a uniform distribution

N-gram dialogue act user simulation

- Can be easily learned from the collected data
- These users are not rational due to small context considered
- Values are very often randomly selected
  - there is not enough data to estimate probabilities for this
- Example:
  - S: request(pricerange)
  - U: inform(pricerange=cheap)
  - S: request(pricerange)
  - U: inform(pricerange=expensive)

The use did not changed its mind. **He did not had a clue about what he said before.**

Agenda based simulations

- Inspired by agenda based approach for SDS:
  - X Wei and AI Rudnicky. 1999. An agenda-based dialog man- agement architecture for spoken language systems. In Proc. of IEEE ASRU. Seattle,WA.
- In this case, the dialogue state is factored into
  - the goal – G
  - the agenda – A
- The goal ensures that the user behaves in a consistent, goal-directed manner.
- For example in tourist information domain, the goal can be further factored into
  - requirements - R
  - constraints - C

Agenda based simulations

- The user agenda is a stack-like structure containing the pending user dialogue acts
- Note: the agenda is a special way of representing a dialogue policy and the context history at the same time
- At the start of the dialogue the agenda contains
  - Inform acts about all constraints
  - Request act for all requirements
  - Bye dialogue act

Agenda update

- Based on the systems responses, update the agenda of the simulator, e.g. what to say next

$$P\left(a_{t+1} \middle| a_t, g_t, u_{a:t}\right)$$

- Although it can have probabilistic representation, it is mostly done deterministically, e.g.:
  - If system misunderstands then the user simulator puts on the agenda a correction

- Since agenda operates with a stack-like structure, the allowed operation are:
  - push and pop acts to and from agenda
  - the dialogue end when *bye()* act is popped of the stack

Example

$$C_0 = \begin{bmatrix} type = bar \\ drinks = beer \\ area = central \end{bmatrix}$$

$$R_0 = \begin{bmatrix} name = \\ addr = \\ phone = \end{bmatrix}$$

Sys 0 Hello, how may I help you?

$$A_1 = \begin{bmatrix} inform(type = bar) \\ inform(drinks = beer) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 1 I'm looking for a nice bar serving beer.

Sys 1 Ok, a wine bar. What pricerange?

$$A_2 = \begin{bmatrix} \underline{negate(drinks = beer)} \\ inform(pricerange = cheap) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 2 No, beer please!

Sys 2 A bar serving beer, correct?

$$A_3 = \begin{bmatrix} affirm() \\ inform(pricerange = cheap) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 3 Yeah something cheap in the town centre.

Sys 3 Murphy's on Main Square serves cheap beer.

$$A_4 = \begin{bmatrix} request(phone) \\ bye() \end{bmatrix}$$

Usr 4 Ok, and what's the phone number?

Sys 4 The number is 796 69 94.

$$A_5 = \begin{bmatrix} bye() \end{bmatrix}$$

Usr 5 Thanks, goodbye!

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., & Young, S. (2007). Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. Proceedings of HLT/NAACL. Rochester, NY.

Agenda update

- Agenda update tends to be complex
  - due to limited number of ways of updating the agenda
  - !!! the update is limited to make its use easier !!!
- Agenda update is unnecessary factored into to too small and specific operations
- Some trivial things are just too complex to be achieved with push and pop operations

ISU based approach

- ISU based approach would simply build a dialogue system behaving like a user:

- Such system:
  - would track what the dialogue systems said
  - compared that information with the goals of the user
  - then made it decisions using either:
    - a handcrafted policy
    - a stochastic policy

- To limit complexity of the dialogue policy
  - summary action can be used
  - they can be tuned to needs of a user simulator

ISU based approach

- It is like building a dialogue system without reinforcement learning
  - we can learn stochastic policy using maximum likelihood

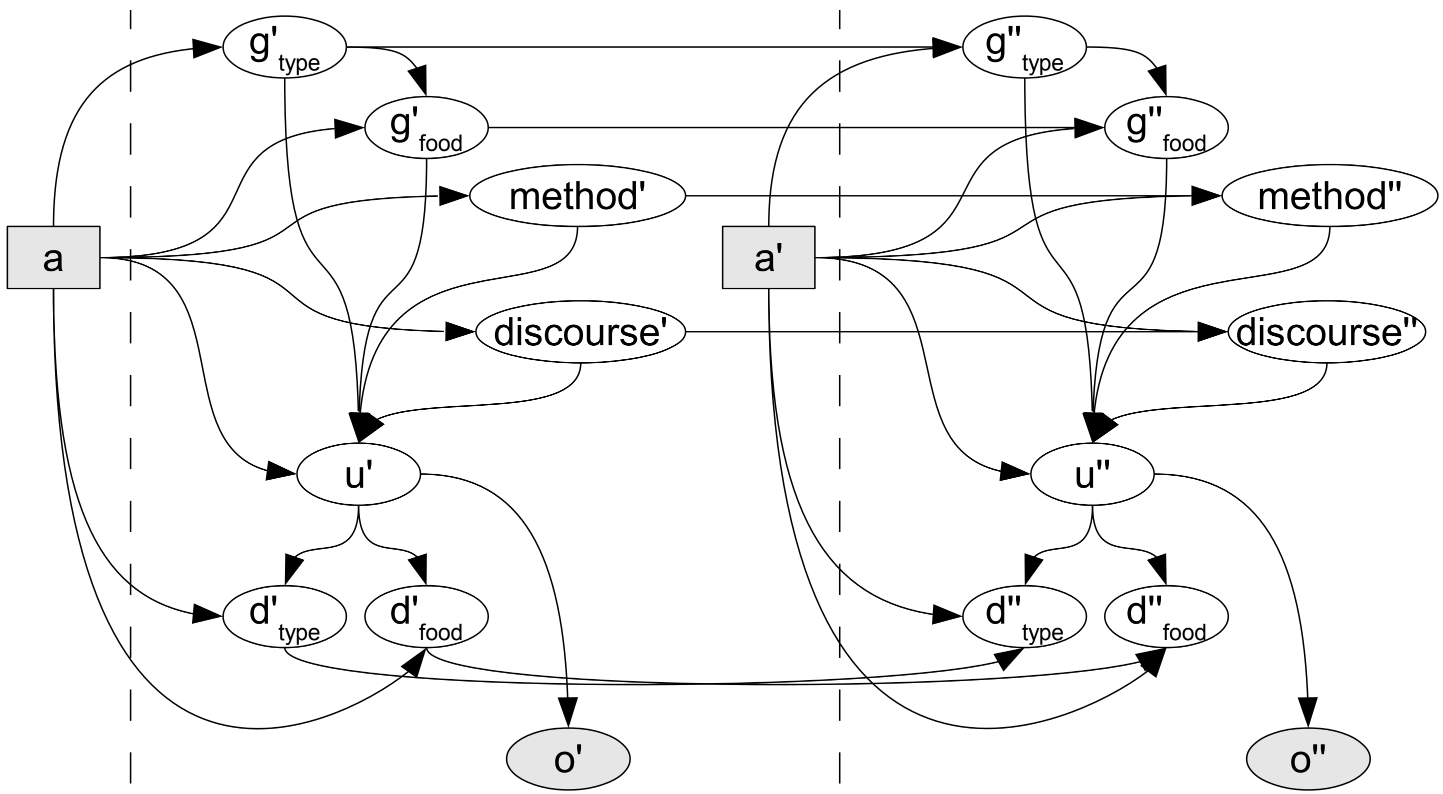
- We do not want to be better than a user

- We want to do exactly what a user is doing

- Though, we could use reward function to constrain our model to, for example, cooperative policies
  - e.g.: a user wants to succeed in a dialogue

Bayesian approach

- Use model similar to the dialogue model in a dialogue manager



Sample from the Bayesian model

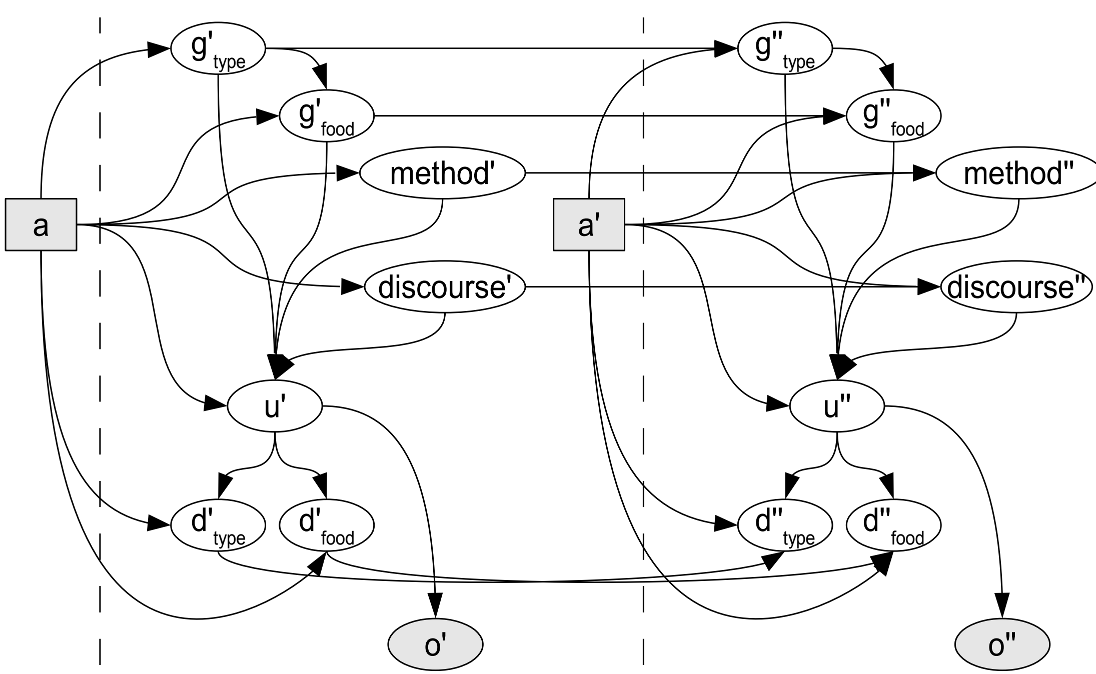
- Sample next dialogue state

$$s \sim P(s_{t+1} | s_t, a_t)$$

- Sample next user action

$$o_{t+1} \sim P(s_{t+1})$$

- Problem
  - it is not always rational
  - this is similar to the N-gram dialogue act level simulations



Bayesian approach

- Ideally the model for simulations and the dialogue system would be the same
- However, as typical for generative models:
  - generative models can be useful for classification
  - without being good for generation!
- The model for user simulations must be:
  - more constrained → more (probabilistic) handcrafting
  - it needs longer context
  - it turns out to be complex and hard to maintain
- However, it appears to be the principled way to do

Error generation

- So far, we wanted from the user simulator reasonable behaviour

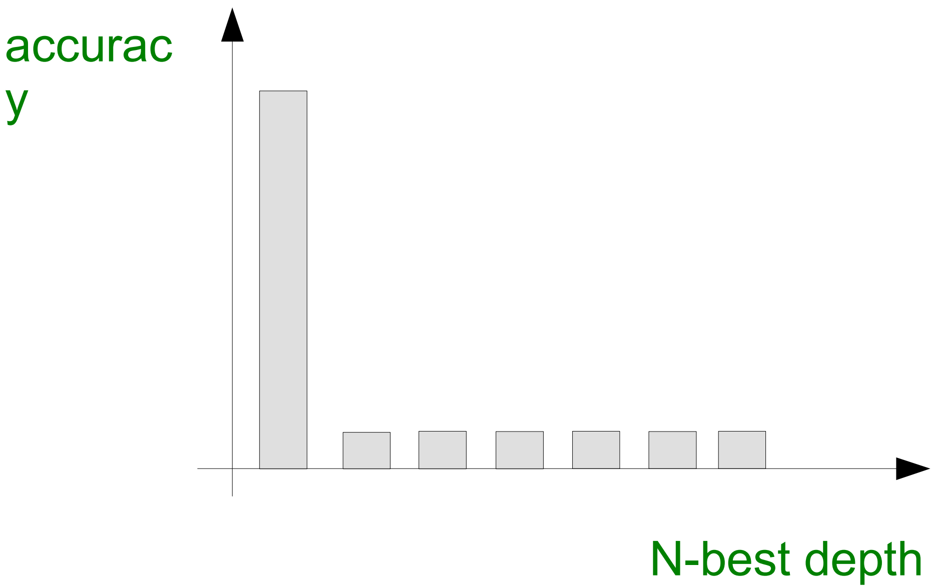
- However, for training a robust dialogue system we need a model of errors: ASR + SLU

inform(food=Italian)&request(price): 1.0	→	inform(food=Italian)	: 0.50
	→	inform(food=Italian)&request(price)	: 0.20
	→	request(price)	: 0.20
	→	confirm(food=Italian)	: 0.05
	→	null()	: 0.05

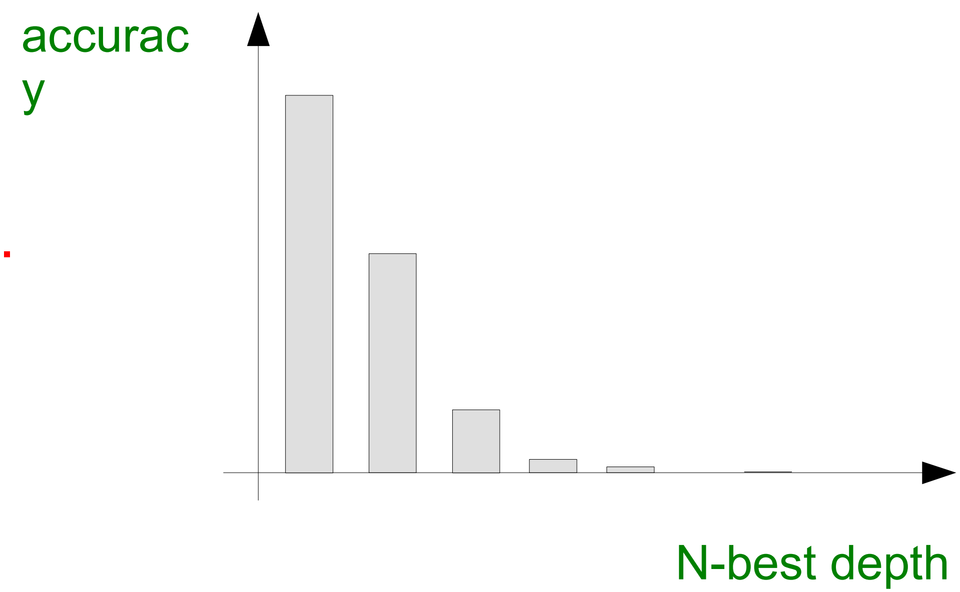
- Ideally, we can control this error model using a simple variable: error rate
  - measure of the number of errors in the N-best lists / top hypothesis

Error generation

- We want to control both the accuracy of the top hypothesis and the distribution of the accuracy at different positions in the N-best list

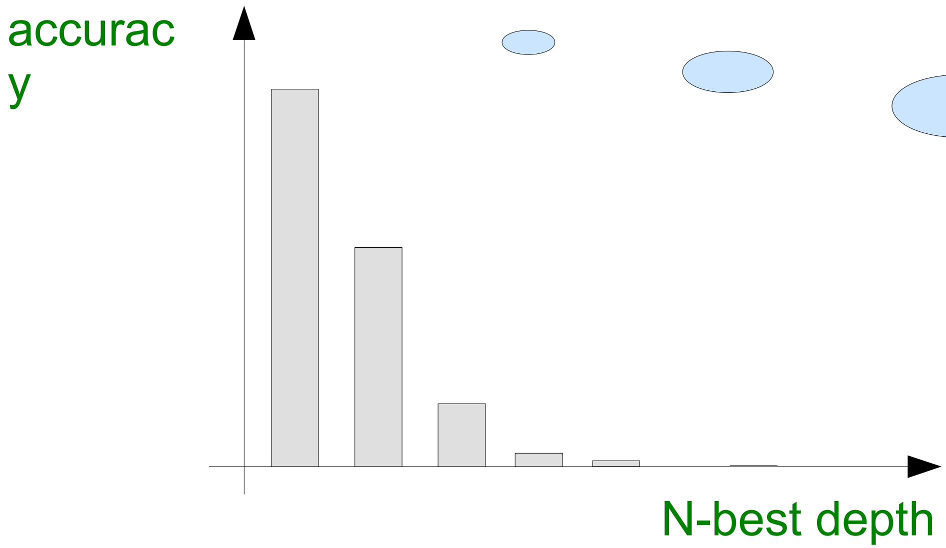


VS.



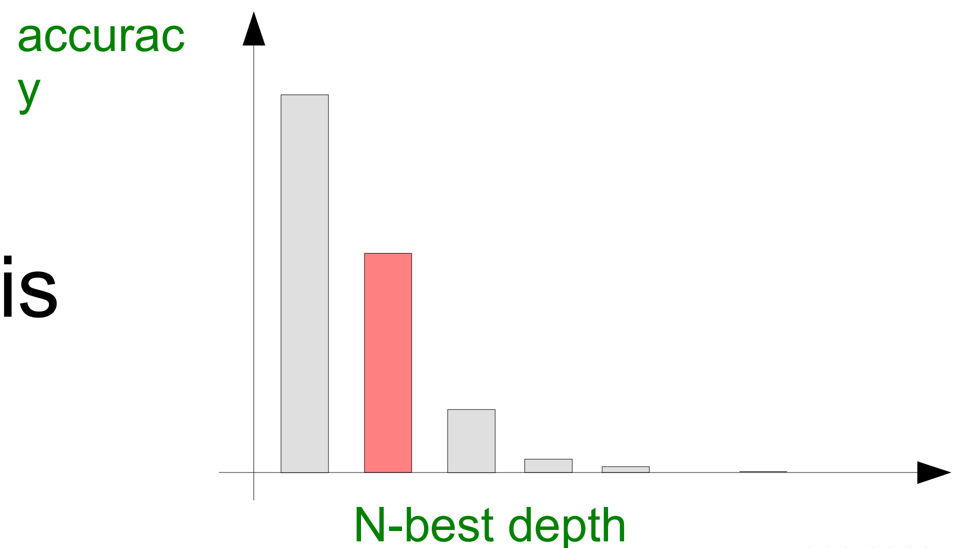
Model for error generation

- Error model : accuracy = P(position in the N-best list)



This is multinomial, possibly infinite, distribution.

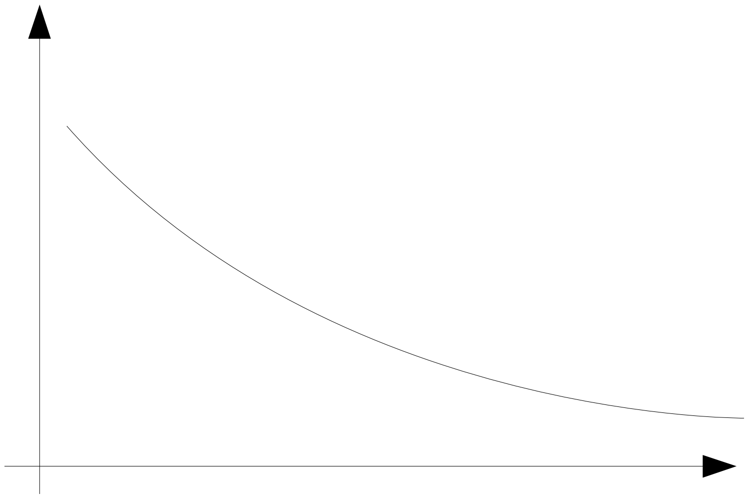
- Sample position of the correct hypothesis



Model for error generation

- Example implementation

$$P(n|err;\alpha) \approx e^{-\alpha \frac{n}{err}}$$



- Having a position of the correct hypothesis

inform(food=Italian)&request(price): 1.0	→	…	: 0.50
	→	inform(food=Italian)&request(price)	: 0.20
	→	…	: 0.20
	→	…	: 0.05
	→	…	: 0.05

- We have to fill in the missing hypotheses

Model for error generation

- Sample from a dialogue act confusion model

$$d \sim P\left(d_{err} \middle| d, err, n\right)$$

- This is usually factored into

$$d \sim \prod_i P\left(dact^i_{err} \middle| dact^i, err, n\right) P\left(sn^i_{err} \middle| dact^i_{err}, sn^i, err, n\right)$$
$$P\left(sv^i_{err} \middle| sn^i_{err}, err, n\right)$$

- And then the individual probabilities are usually handcrafted:
  - typically ignores **err** and **n**

Trained model for error generation

- Use logistic (multiclass) regression to train the model from errors generated by ASR, SLU

$$P\left(dact_{err} \middle| dact, err, n\right) \approx e^{\theta^T \Phi(dact_{err}, dact, err, n)}$$

$$P\left(sn^i_{err} \middle| act^i_{err}, sn^i, err, n\right) \approx e^{\theta^T \Phi(sn_{err}, act_{err}, sn, err, n)}$$

$$P\left(sv^i_{err} \middle| sn^i_{err}, err, n\right) \approx e^{\theta^T \Phi(sv^i_{err}, sn^i_{err}, err, n)}$$

- Every time you change ASR (AM, LM) or SLU, you have to re-train these models

Word level user simulation

- In addition to dialogue act level simulation
  - implements reasoning and goal oriented behaviour
- It include the generation of sentences
  - e.g. using a corpus of dialogue act annotated sentences
- The confusion model then operate on words
  - it learns mapping between correct word strings and confused words
  - it can use again pre-recorded and consequently recognised corpus
  - however, some generalisation is needed to obtain larger variability in the output

Word level user simulation

- Tries to solve the problem of similarity of
  - inform(type=bar) - “A bar please!”
  - inform(drinks=beer) - “Uh beer please!”
- and dissimilarity of
  - inform(type=restaurant) - “A restaurant please!”
- In general, semantically similar items does not have to be similar on the word/phone level

Word level simulation

- Phone level simulation would be ideal
- However, this is far too complex

- It is more efficient to learn how to confuse individual words or word sequences

- Note: this is still too computationally expensive anyway

Word level confusion model

$$P(\tilde{a_u}|a_u) = \sum_{\tilde{w_u}} \sum_{w_u} P(\tilde{a_u}, \tilde{w_u}, w_u|a_u)$$

$$= \sum_{\tilde{w_u}} \underbrace{P(\tilde{a_u}|\tilde{w_u})}_{\substack{\text{semantic}\\\text{decoder}}} \sum_{w_u} \underbrace{P(\tilde{w_u}|w_u)}_{\substack{\text{confusion}\\\text{model}}} \underbrace{P(w_u|a_u)}_{\substack{\text{utterance}\\\text{generation}}}$$

$$P(\tilde{w}_u|w_u) = \sum_{\lambda} P(\tilde{w}_u, \lambda | w_u)$$

Schatzmann, J., Thomson, B., & Young, S. (2007). Error Simulation for Training Statistical Dialogue Systems. ASRU (pp. 2-7). Kyoto, Japan.

Utterance generation model

- For error generation, a simple template generation model can be considered

Source:	inform(food=Chinese, pricerange=cheap) CHINESE FOOD IN THE CHEAP PRICERANGE
Template:	inform(food=$X, pricerange=$Y) $X FOOD IN THE $Y PRICERANGE
Unseen:	inform(food=French, pricerange=expensive) FRENCH FOOD IN THE EXPENSIVE PRICERANGE

- Templates can be easily derived from the corpus using a database of slot values
- P(template|dialogue act) can be also collected

Word confusion model

- At the word-level, ASR confusions can be viewed as translations of a source utterance w_s to a confused target utterance w_t

$$P(\tilde{w}_u | w_u) = \sum_{\lambda} P(\tilde{w}_u, \lambda | w_u)$$

```
I WANT AN EXPENSIVE HOTEL PLEASE
1 2    3  3         4     5
| |    | /          |     |
1 2    3            4     5
  ONE  INEXPENSIVE  HOTEL PLEASE
```

Word confusion model

- There are many ways how to define this confusion model

- Nevertheless, all depend on some alignment of between w_s and w_t

- The easiest is to use a Levenshtein distance
    - automatically generated alignment
    - word level with costs defined on the letter/phone level

- This is a bit different from machine translation since we work with the same language on both sides

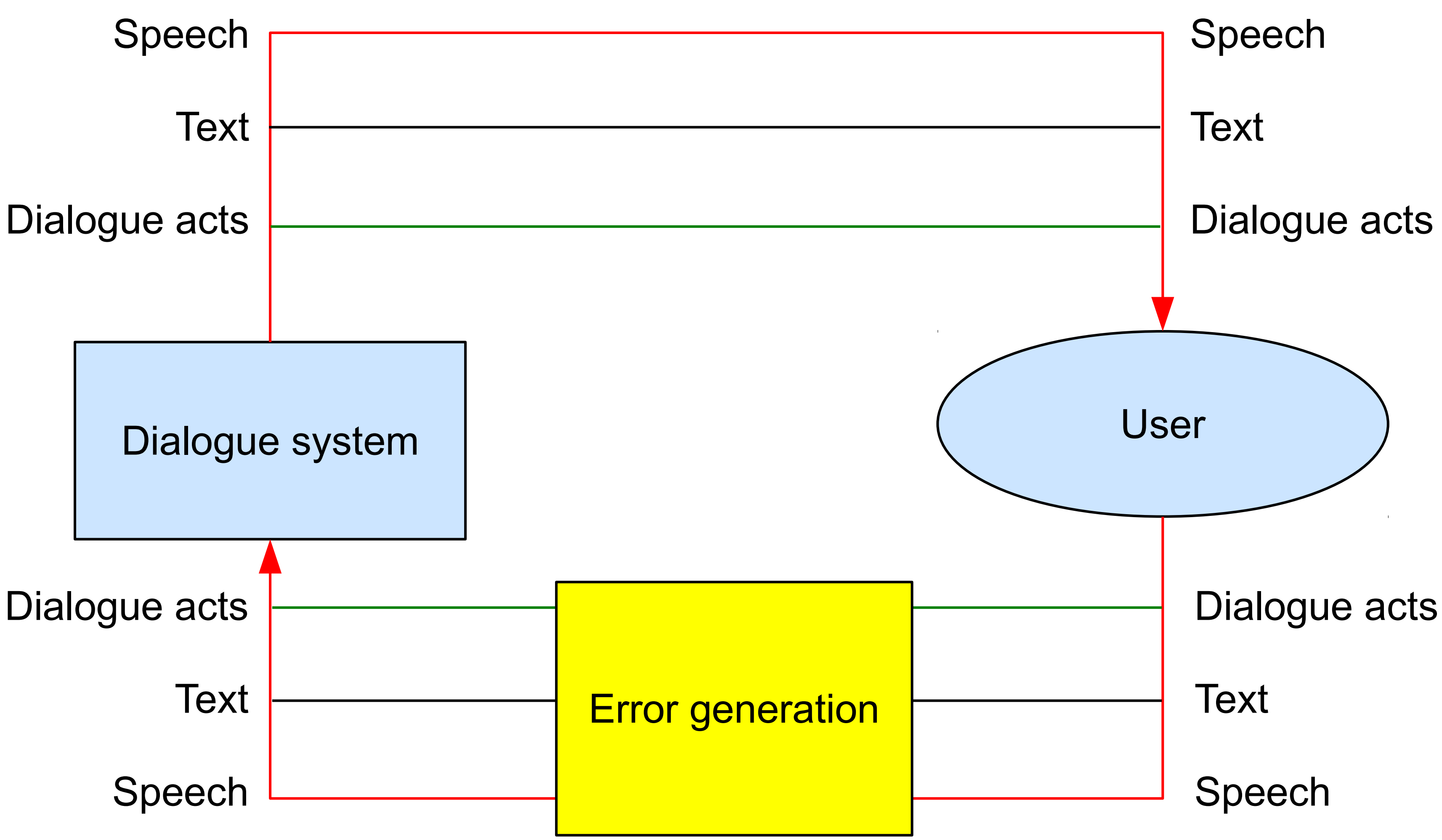
Word confusion model

- Once you get the alignments,
  - you can compute word / word sequence confusion model
  - e.g. "A BAR" maps to
    - "ALL", "ART", "A BAR", "A BAR", "A CAR", "BAR", "BAR", "BAR", "BAR", "CAR"
  - and corresponding probabilities
- Then the following can be used compute the probability of the alignment

$$P\left(\tilde{w}_u, \lambda \middle| w_u\right) \approx \prod_i P\left(\tilde{w}_{\lambda_i} \middle| w_{\lambda_i}, w_{\lambda_{i-1}}, \tilde{w}_{\lambda_{i-1}}\right)$$
$$\approx \prod_i P\left(\tilde{w}_{\lambda_i} \middle| w_{\lambda_i}\right)$$

Types of user simulation

- Dialogue act level, Word level, Speech level



Speech level user simulation

- In addition to word level simulation, it generate speech
- The generation of speech is done using TTS on correctly generated text
- The noise is added by simply mixing generated speech and some artificial noise
- The noise is controlled by its volume and type
- Problems:
  - The generated text has low variability and can be unnatural in some cases
  - The synthesised speech has low variability
  - Obviously, the acoustic noise is not responsible for all uncertainty in the SDS input

Thank you!

Filip Jurčíček

Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic

Home page: http://ufal.mff.cuni.cz/~jurcicek