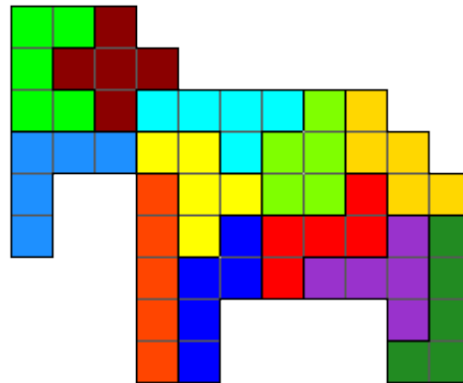


# Enabling In-Turn Processing in Spoken Dialogue Systems

---



Timo Baumann

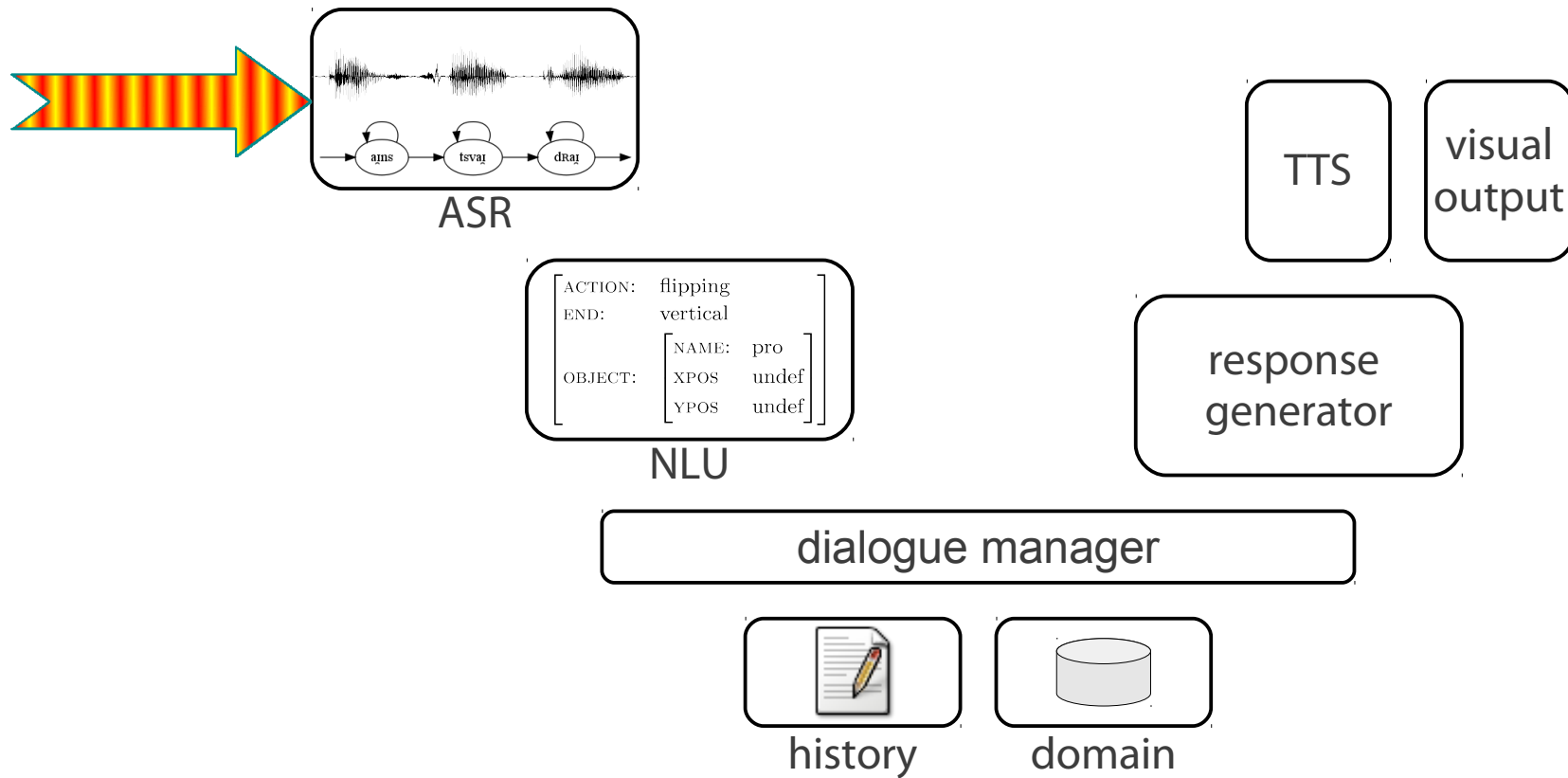
[mail@timobaumann.de](mailto:mail@timobaumann.de)

<http://www.ling.uni-potsdam.de/~timo>

---

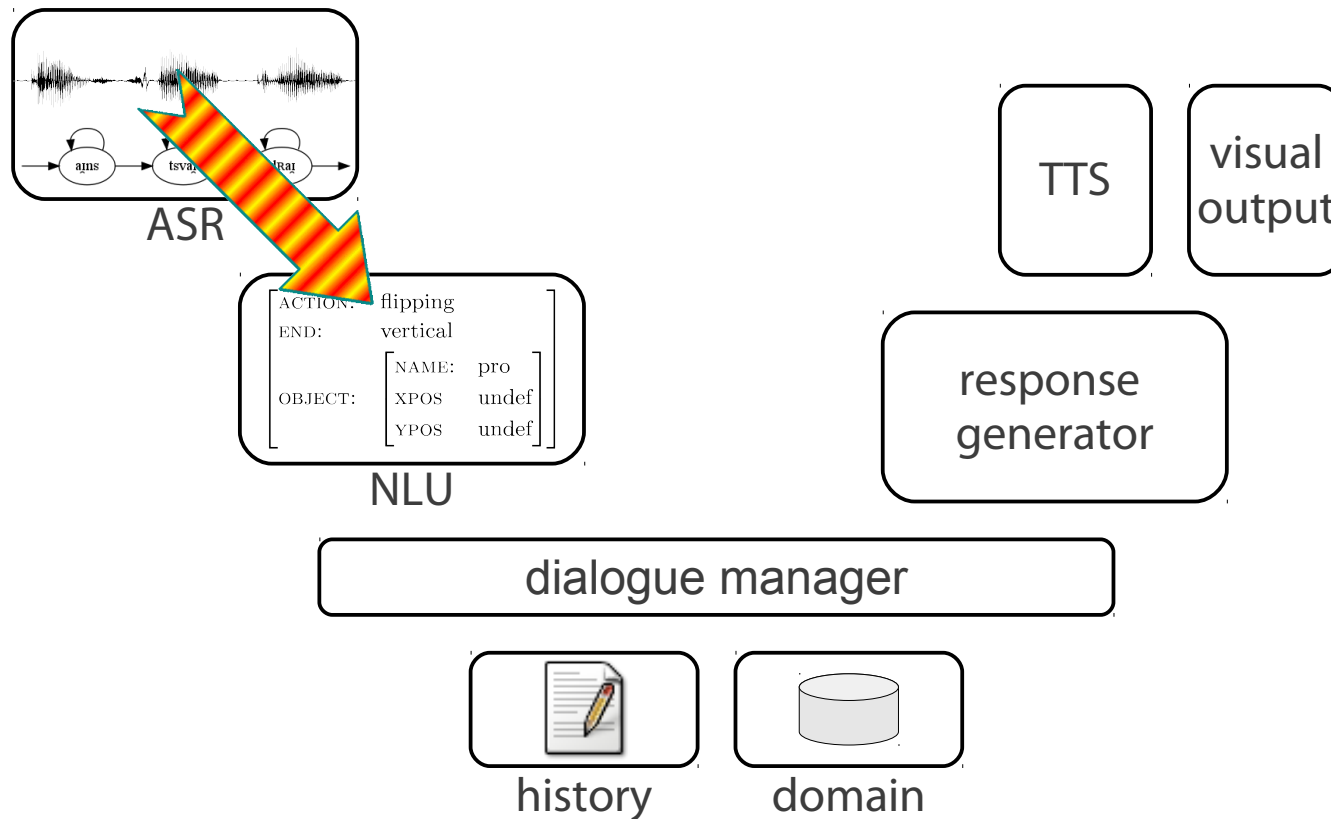
# Spoken Dialogue Systems

---



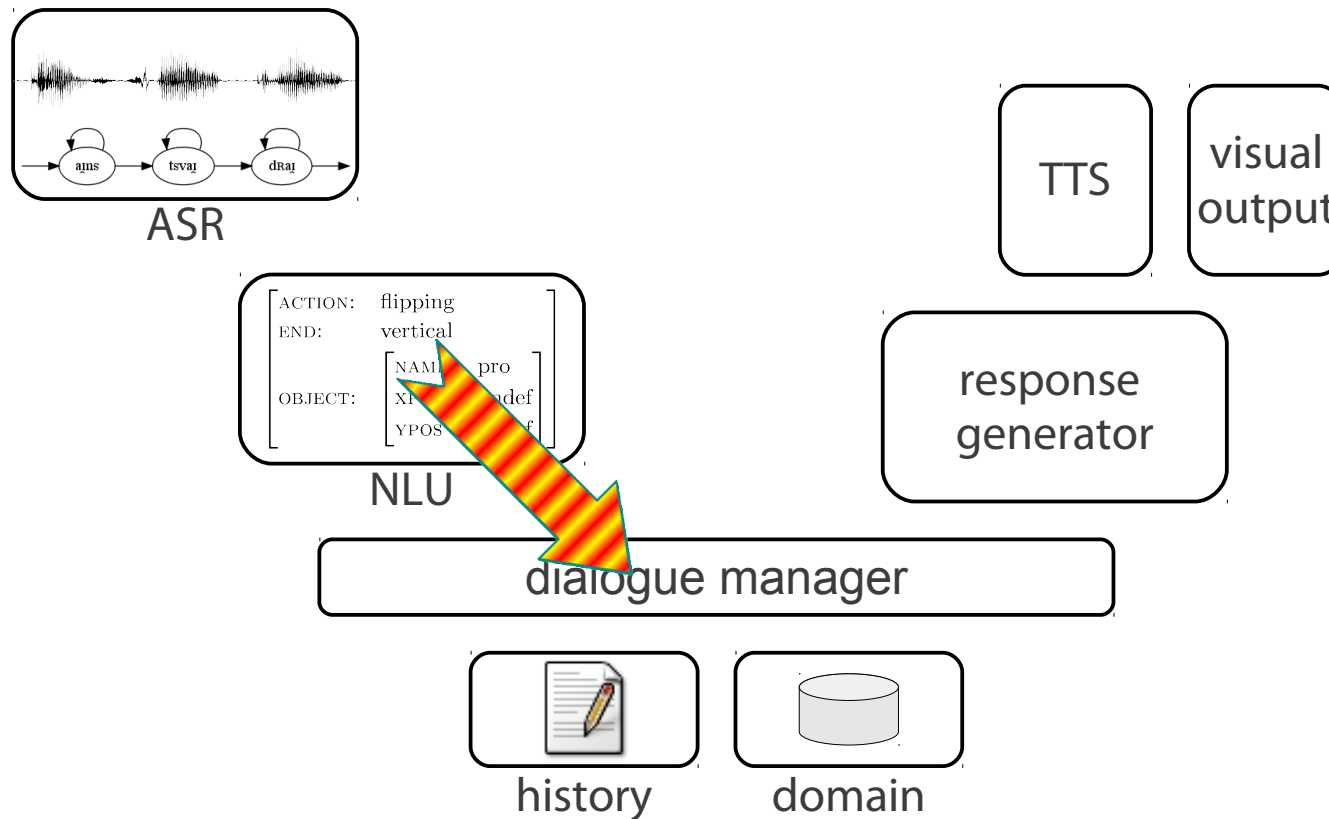
# Spoken Dialogue Systems

---



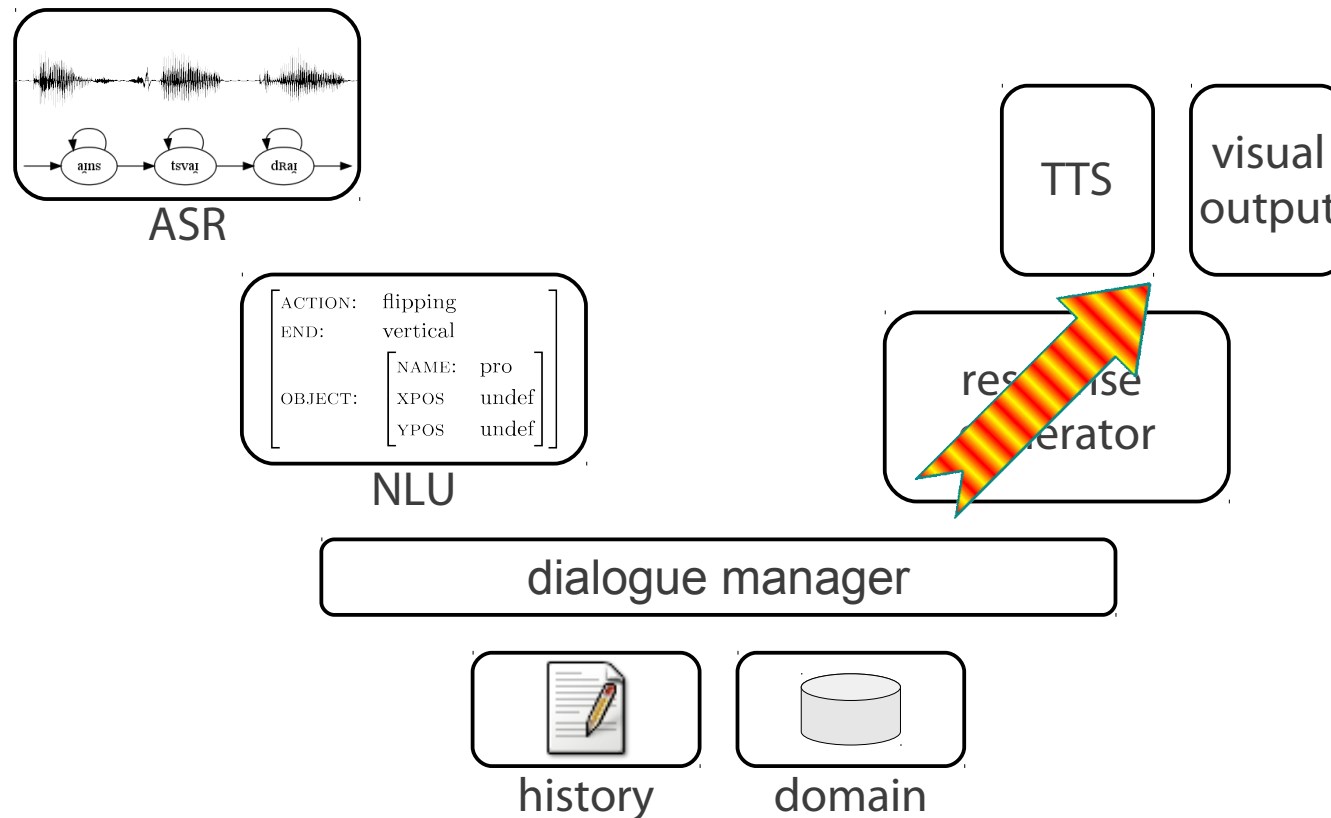
# Spoken Dialogue Systems

---



# Spoken Dialogue Systems

---

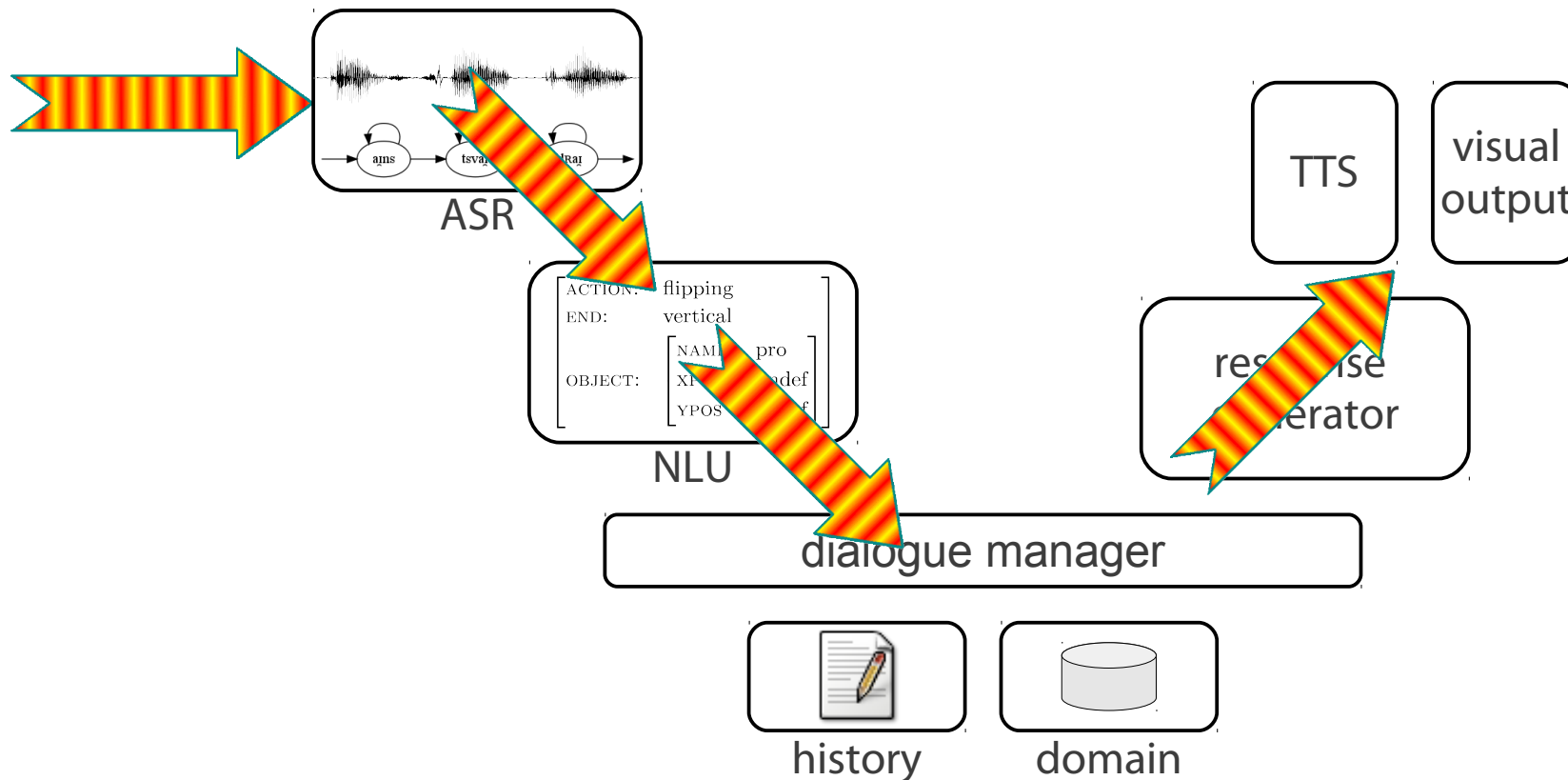


- modules start after their predecessors have finished
-

# Incremental

## Spoken Dialogue Systems

---



- **partial results** are passed on immediately
  - reaction is quicker, interaction ideally more natural
-

# Benefits of Incremental Spoken Dialogue Systems

---

1. react more quickly

as modules process input during a speaker's turn:

U: Ich möchte am Samstag von Berlin  
nach Hamburg fahren.

S: Ok, um wieviel Uhr möchten Sie fahren?

(Crafted examples for an imaginary train timetable information system.)

---

# Benefits of Incremental Spoken Dialogue Systems

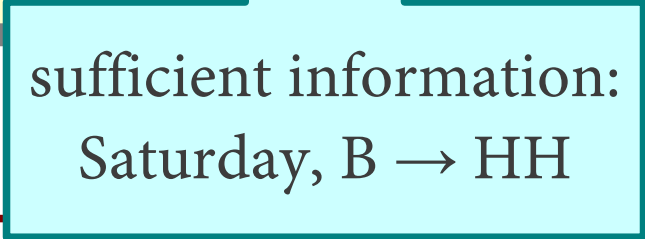
---

1. react more quickly

as modules process input during a speaker's turn:

U: Ich möchte am Samstag von Berlin  
nach Hamburg fahren.

S: Ok, um wievi... möchten Sie fahren?



sufficient information:  
Saturday, B → HH

---

# Benefits of Incremental Spoken Dialogue Systems

---

1. react more quickly  
as modules process input during a speaker's turn
2. give feedback during a speaker's turn:

U: Ich möchte am Samstag mit dem ICE  
Nummer, äh ... warten sie ... 798 ...  
S: ja? ok.

- feedback might be visual in a multi-modal system
-

# Benefits of Incremental Spoken Dialogue Systems

---

1. react more quickly  
as modules process input during a speaker's turn
2. give feedback during a speaker's turn
3. even interrupt a speaker's turn:

U: Ich möchte am Samstag mit dem ICE  
Nummer 798 nach, äh ...

S: Entschuldigung, ICE 798 verkehrt nicht  
samstags, Sie wollen nach Hamburg?

---

# Benefits of Incremental Spoken Dialogue Systems

---

1. react more quickly  
as modules process input during a speaker's turn
2. give feedback during a speaker's turn
3. even interrupt a speaker's turn

→ all these capabilities make the SDS **more similar** to a human interlocutor

---

# Content:

---

- ✓ Advantages of incremental SDSs
  - Architecture for incremental SDSs
  - Evaluating and Optimizing incremental ASR
  - Predicting the Micro-Timing of a User's Words
  - Wrap-up
-

# Requirements: Dealing with Uncertainty

---

- intermediate hypotheses **change with time**
    - we may get things wrong intermittently:  
    „Hamburg“ → /hamburg/
  - *Incrementally* this will look to  
speech recognition as follows ...
-

# Requirements: Dealing with Uncertainty

---

- intermediate hypotheses **change with time**
  - we may get things wrong intermittently:

„Hamburg“ → /hamburg/

- ASR:

this sounds like  
„Hamm“

- NLU:

they must be talking about  
[city:Hamm(Westfalen)]

---

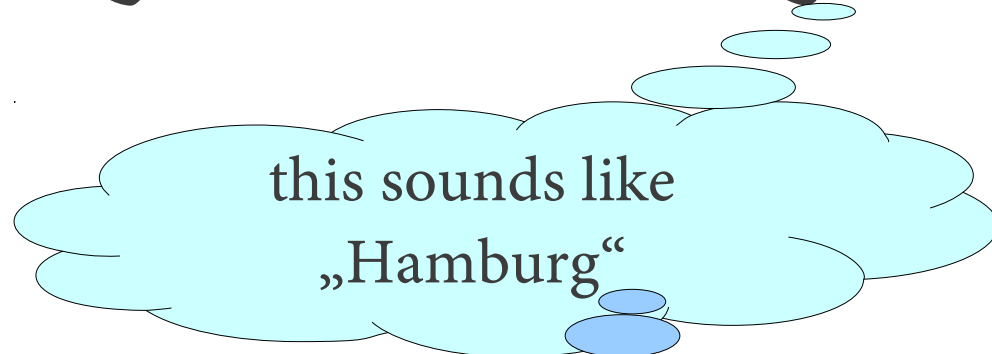
# Requirements: Dealing with Uncertainty

---

- intermediate hypotheses **change with time**
  - we may get things wrong intermittently:

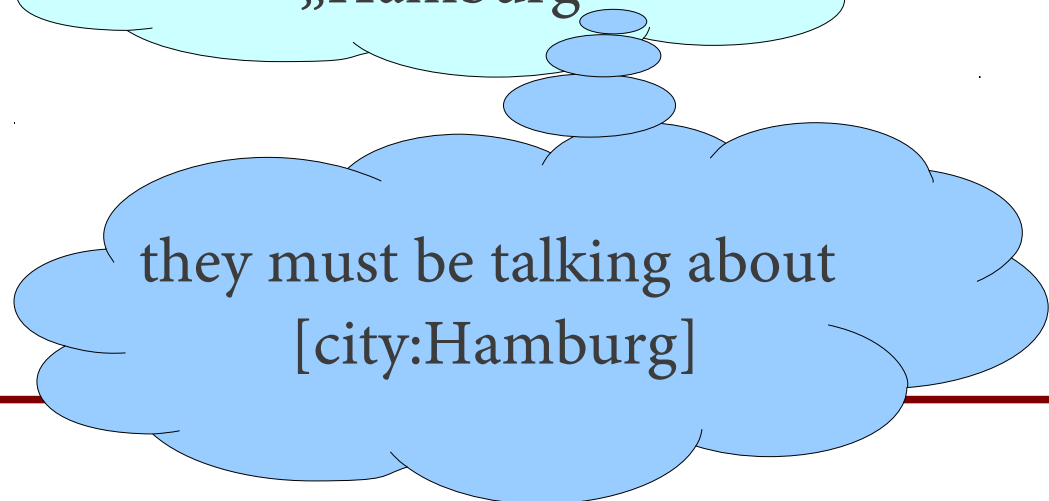
„Hamburg“ → /hamburg/

- ASR:



this sounds like  
„Hamburg“

- NLU:



they must be talking about  
[city:Hamburg]

---

# Requirements: Dealing with Uncertainty

---

- intermediate hypotheses **change with time**
    - we may get things wrong intermittently:  
    „Hamburg“ → /hamburg/
  - Couldn't the ASR just lag behind a little bit?
-

# Requirements:

## Dealing with Uncertainty

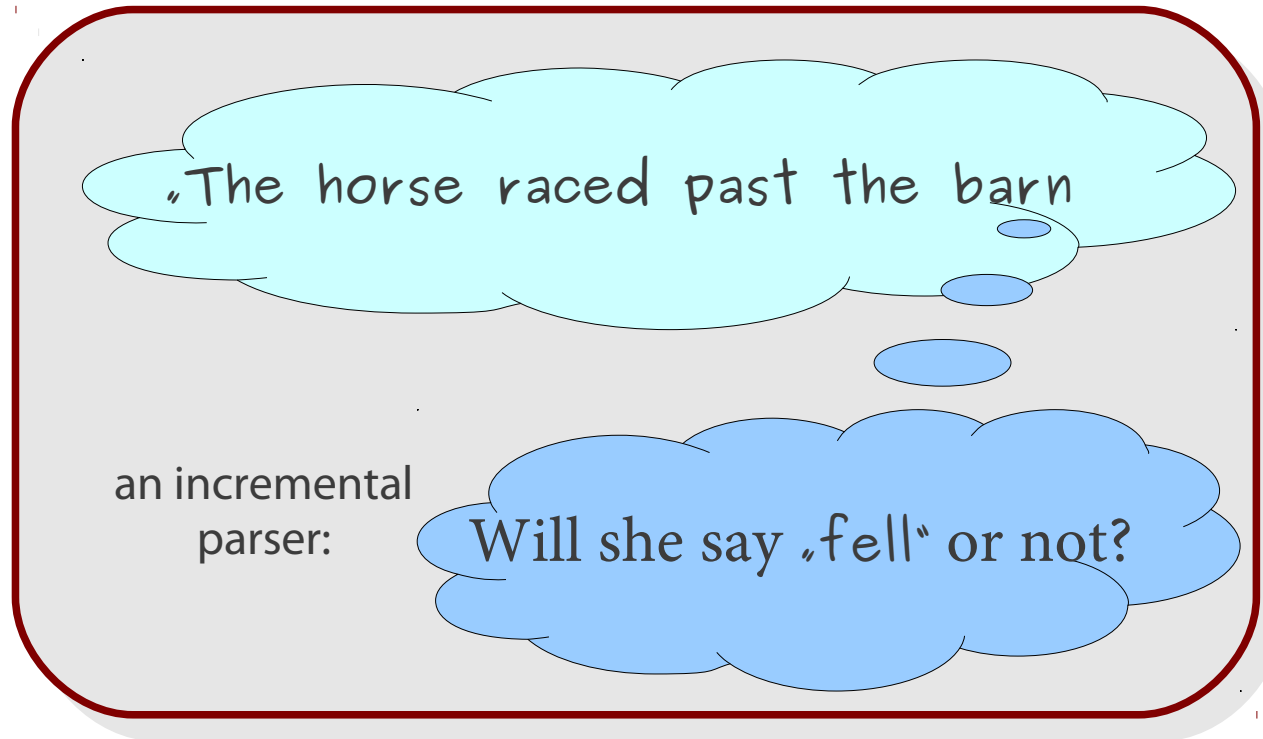
---

- Couldn't the ASR just lag behind a little bit?
  - Yes, but:
    - long-distance dependencies:
      - ♦ disambiguation can be arbitrarily late, no fixed delay will capture that
    - delays in all processors will add up
      - ♦ distance can be great in later processors
- hence, previous hypotheses must be **changeable**
-

# Requirements: Dealing with Uncertainty

---

- Couldn't we just lag behind a little bit?
- Yes, but:
- e.g. garden-path sentences, ...



→ hence, previous hypotheses must be **changeable**

---

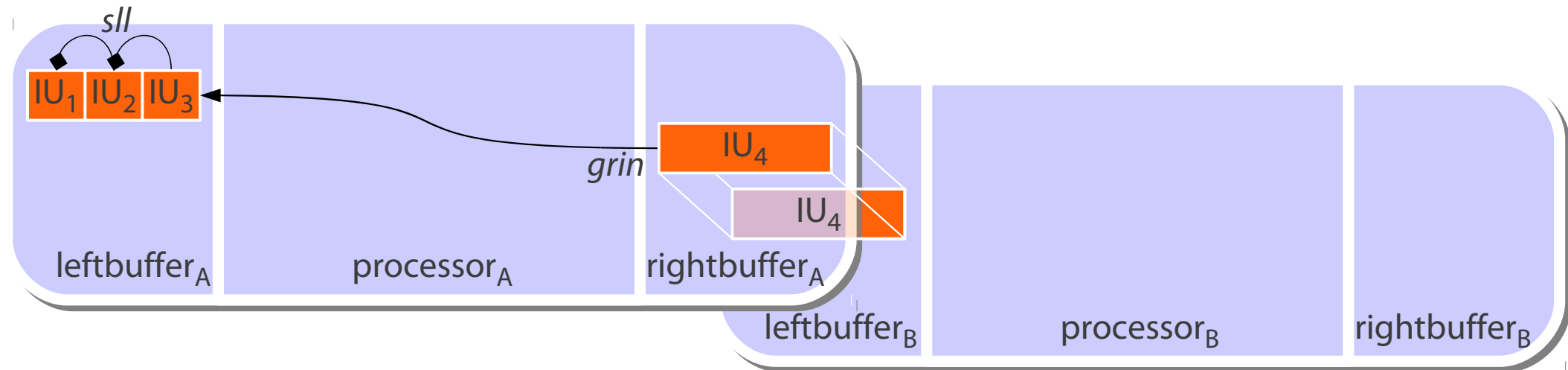
# Requirements: Dealing with Uncertainty

---

- hence, previous hypotheses must be **changeable**
    - this isn't really surprising, I hope
    - however, it challenges the standard SDS architecture (VoiceXML, loosely coupled components, barge-in, recorded prompts, ...)
    - we'll need a refined architecture
-

# Our notion of incrementality: Incremental Units

- Content is shared in the form of **Incremental Units (IUs)**, which are smallest ‘chunks’ of information



- Links between IUs:
  - **grounded-in** links (*grin*) to denote ancestry
  - **same-level** links (*sll*) for information of the same type

# IU Network

---

- all IUs are connected through (*sll* and *grin*) links
    - this network contains all the information believed by the system at a certain point in time
    - the network is highly dynamic, with *changes* to the network reflecting the system's internal state *over time*
  - Modules react to three basic changes:
    - new IUs are **added**
    - erroneously hypothesized IUs are **revoked**
    - IUs are **committed**, i. e. won't be changed anymore
-

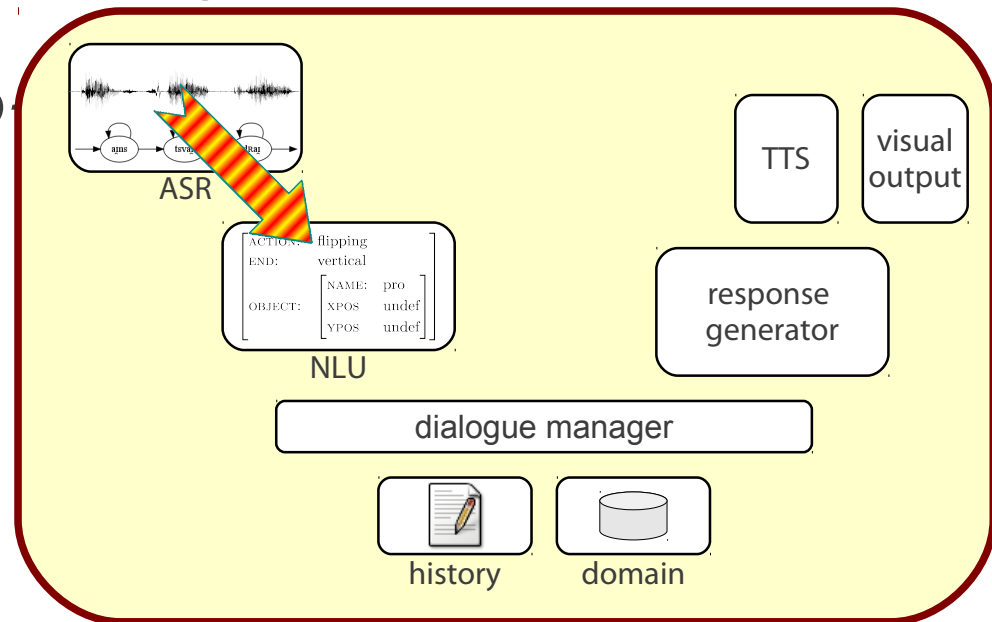
# Content:

---

- ✓ Advantages of incremental SDSs
  - ✓ Architecture for incremental SDSs
  - Evaluating and Optimizing incremental ASR
    - Predicting the Micro-Timing of a User's Words
    - Wrap-up
-

# Content:

- ✓ Advantages of incremental SDSs
- ✓ Architecture for incremental SDSs
- Evaluating and Optimizing incremental ASR
- Predicting the Micro
- Wrap-up



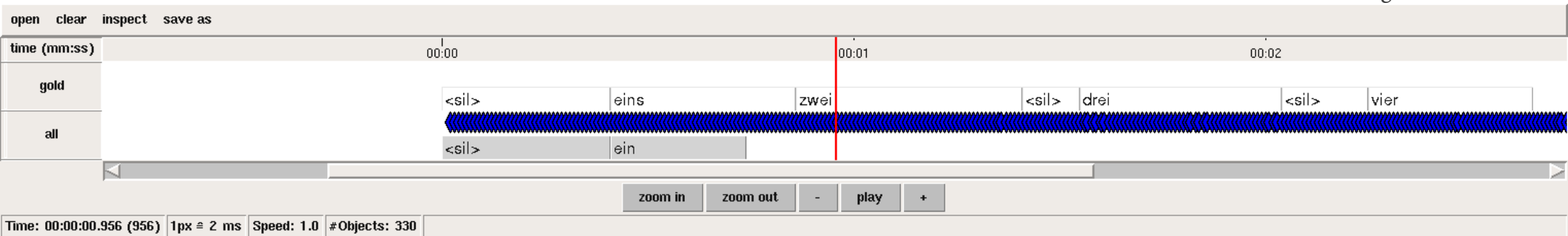
# Incremental Speech Recognition

---

- speech recognition is already incremental
  - walk through all audio frames (Young et al., 1989)
    - keep a list of highest ranking hypotheses up to now
    - update & extend the list with each new frame
    - discard hypotheses that fall out of the beam-size
  - there is a best-ranking hypothesis  $hyp_t$  at all times  $t$
  - let's look at the hypotheses!
-

# A Real-World Example of Incremental ASR Hypotheses

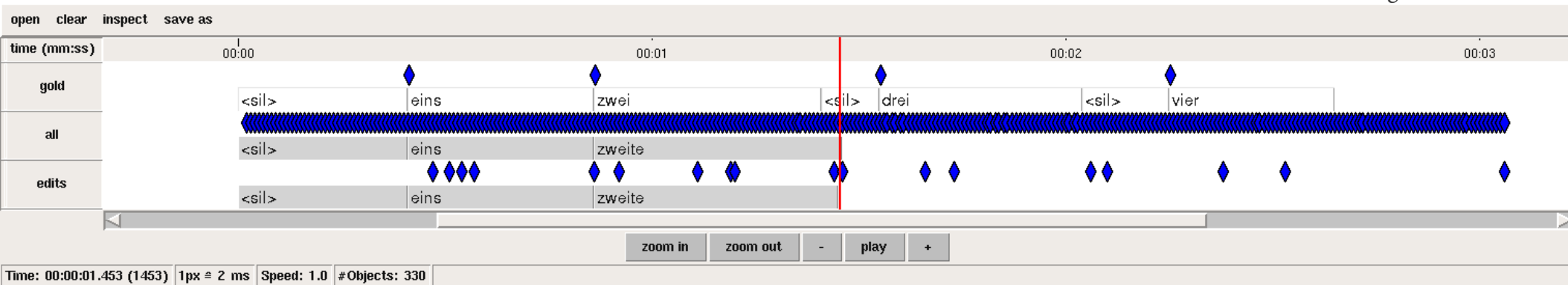
Software from Malsburg et al., 2009



- ASR hypotheses change with time (open video)

# A Real-World Example of Incremental ASR Hypotheses

Software from Malsburg et al., 2009

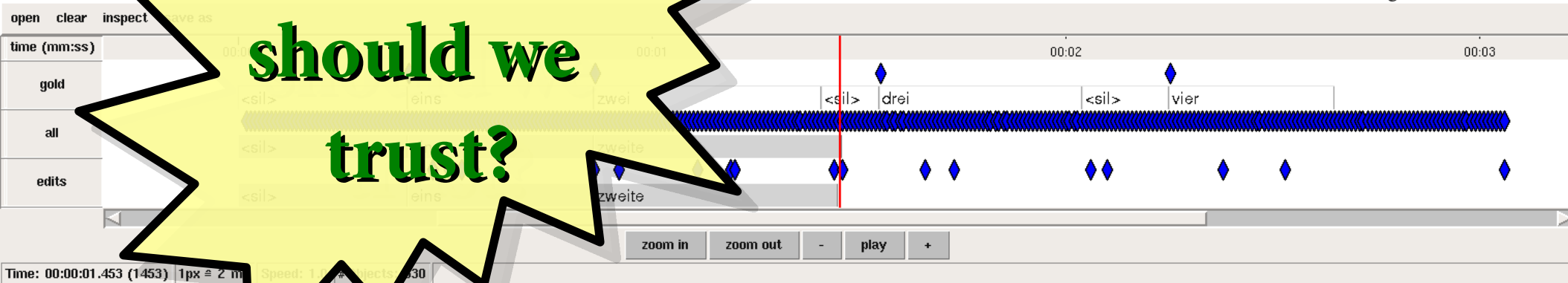


- ASR hypotheses change with time
- more edit than necessary → **overhead ~ 90% !**
  - 90% of a consumer's work will be **useless**

# A Real-World Example of Incremental ASR Hypotheses

**which edits  
should we  
trust?**

Software from Malsburg et al., 2009



- ASR hypotheses change with time
- more edit than necessary → overhead ~ 90 % !

# A Real-World Example of Incremental ASR Hypotheses

Software from Malsburg et al., 2009

**which edits  
should we  
trust?**

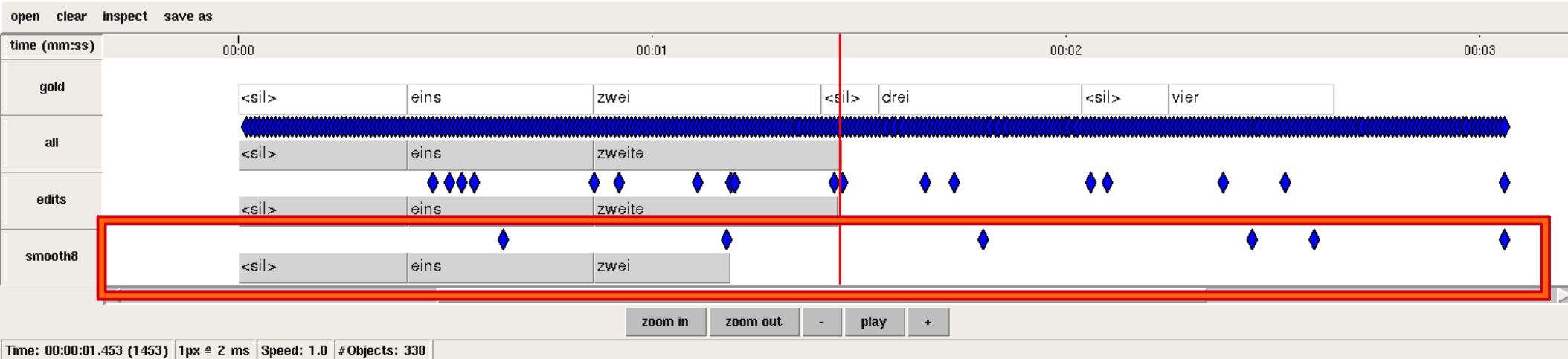
**Patience,  
Young Jedi!  
waiting helps**

- ASR hypotheses change with time
- more edit than necessary → overhead ~ 90%!
- **reduce overhead, sacrifice some timeliness**

# A Real-World Example of Incremental ASR Hypotheses

which edits

Software from Malsburg et al., 2009



waiting helps

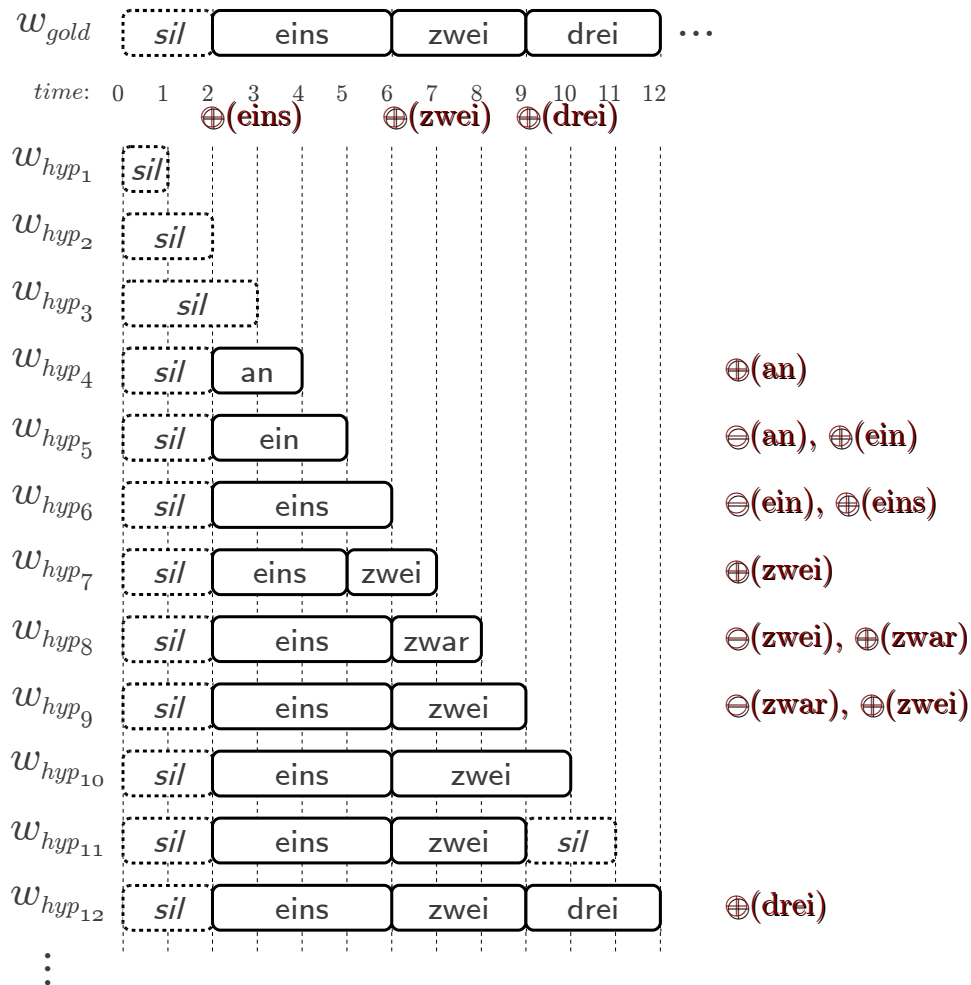
- ASR hypotheses change with time
- more edit than necessary → overhead ~ 90% !
- reduce overhead, sacrifice some timeliness

# Evaluating and Optimizing Incremental Speech Recognition

---

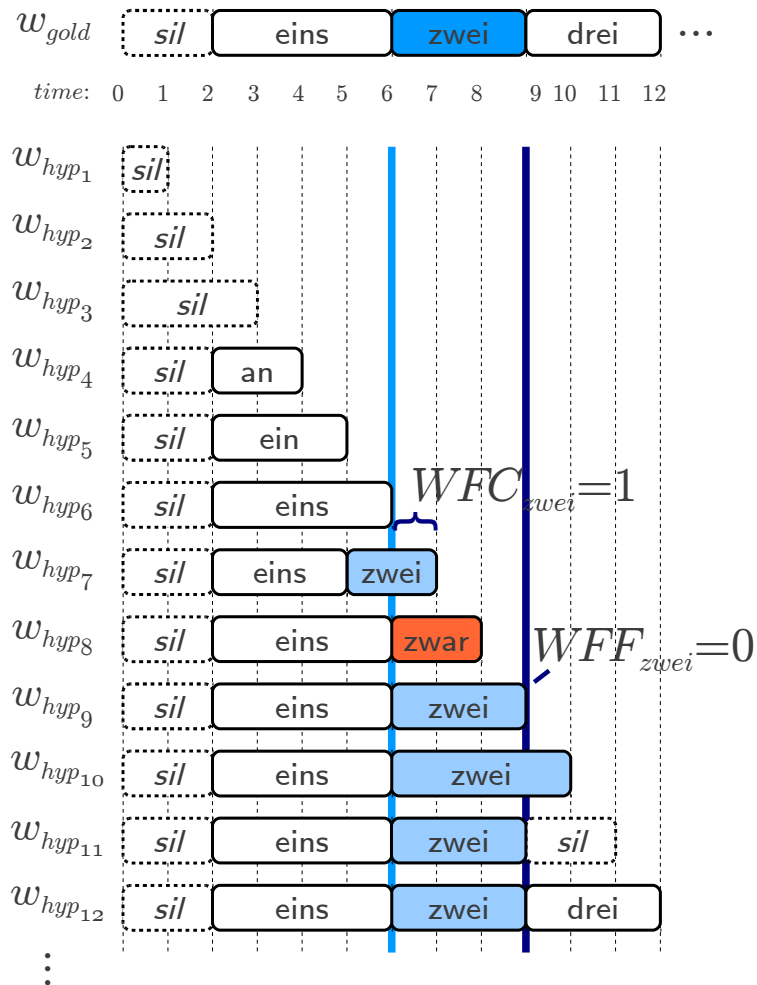
- Speech recognition only optimizes for *final* results
  - intermittent results are thus flaky
- I'll outline ...
  - metrics relevant to incremental performance
  - trade-offs when optimizing
  - two generic optimization techniques
  - incremental performance statistics for our ASR

# Measuring Change



- usually, there are more edits than necessary
- ideally: one *add* per word
- in fact: **edit overhead**
- $EO = \frac{|unnecessary\ edits|}{|edits|}$

# Timing Measures



- when do we find out about a word?
  - word first correct: **WFC**
- when do we become certain about a word?
  - word first final: **WFF**
- this is per word
  - averages are important

# Timing Measures

---

- depending on the use-case we may care for ...
    - if we want to **assume** as soon as possible → low **WFC**
    - if we want to **know** as soon as possible → low **WFF**
  - we can similarly measure the timing of concepts (relative to concerned words), or decisions
-

# Base Measurements

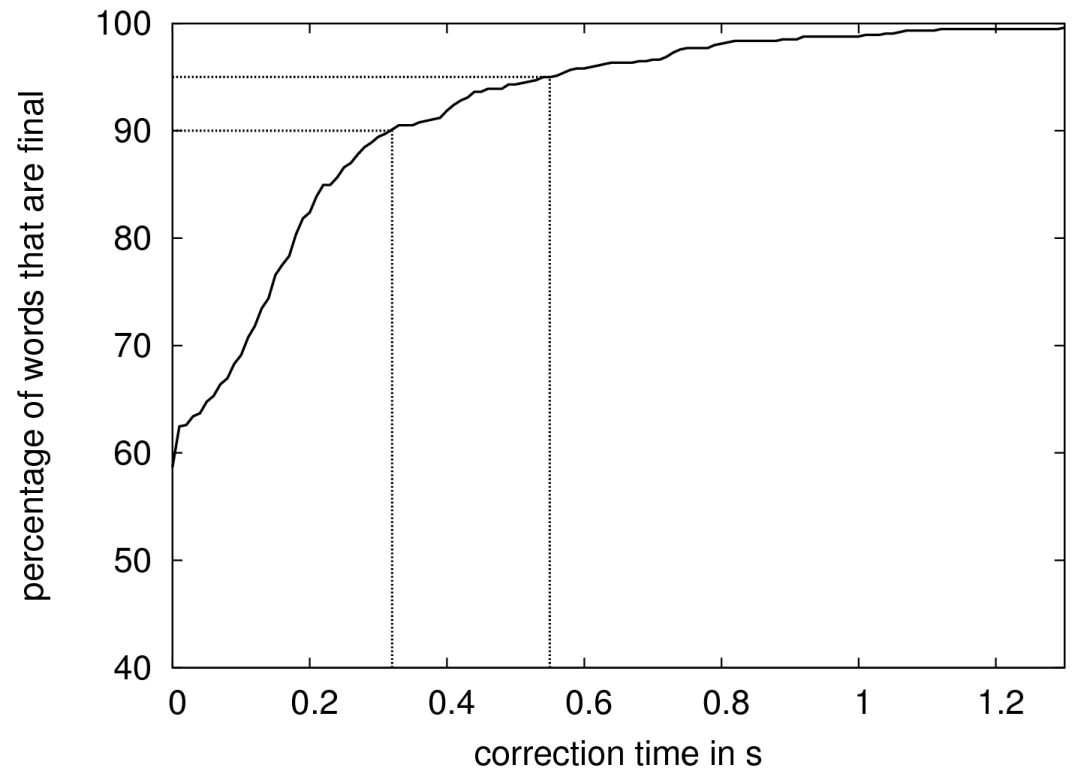
---

- **edit overhead:** 90.5 %
  - most (9 of 10) edits are unnecessary!
- **WFC:** mean=0.276 s, stddev=0.186 s, median=0.230 s
  - average at  $\frac{3}{4}$  of the average word length
- **WFF:** mean=0.004 s, stddev=0.286 s, median=-0.06 s
  - final around word end (on average)

# Certainty Considerations

---

- the **correction time** for a word is **WFF–WFC**
- 58.6 % of all words are immediately correct
- we can calculate the degree of **certainty** for given hypothesis ages
- e.g. if a correct hyp. lasts for 0.55 s, we can be certain (95 %) that it will not change anymore

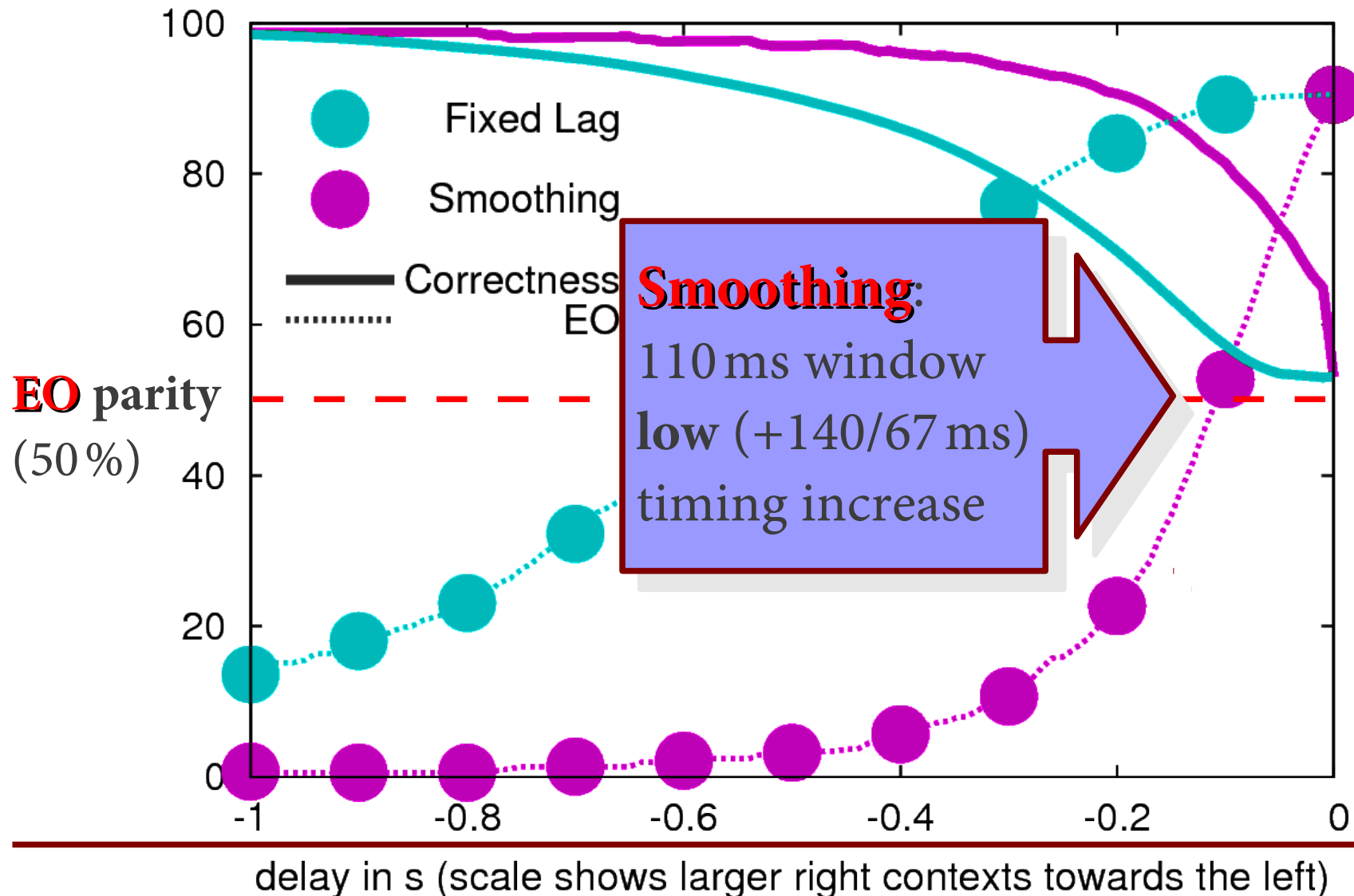


# Improving Incremental ASR

---

- our primary goal is to reduce **edit overhead**
- ... by deferring or suppressing edits
  - results come later, but bad results are suppressed
  - the final (non-incremental) result does not change
- only **trust older parts** of hyps. (Right Context)
  - most of the ASR's *jitter* is at the hypotheses' right end
- only **trust mature edits** (Message Smoothing)
  - most errors only last for a short while

# Right Context vs. Smoothing



# Great, our incremental ASR is *fast*!

---

- it often recognizes a word before it's even over
    - first intuition around  $\frac{3}{4}$  of the word
    - final recognition around end of the word
  - we've got methods to reduce *jitter* in hypotheses
  - now, could there be problems, if recognition is actually too fast?
-

# Great, our incremental ASR is *fast*!

---

- it often recognizes a word before it's even over
  - first intuition around  $\frac{3}{4}$  of the word
  - final recognition around end of the word
- we've got methods to reduce *jitter* in hypotheses
- now, could there be problems, if recognition is actually too fast?

yes ...

---

# Content:

---

- ✓ Advantages of incremental SDSs
  - ✓ Architecture for incremental SDSs
  - ✓ Evaluating and Optimizing incremental ASR
    - Predicting the Micro-Timing of a User's Words
  - Wrap-up
-

# Why do we need micro-timing? (and what do I mean by that?)

---

1. react more quickly,  
**but not too quick:**

sufficient information:  
Saturday, B → HH

U: Ich möchte Samstag von Berlin  
nach Hamburg fahren.

S: ~~Ok.~~ Ok.

---

# Why do we need micro-timing? (and what do I mean by that?)

---

1. react more quickly  
but not too quick
2. when giving back-channel feedback:

U: Ich möchte am Samstag mit dem ICE  
Nummer, äh ... warten sie ... 798 ...  
S: ja? ok.

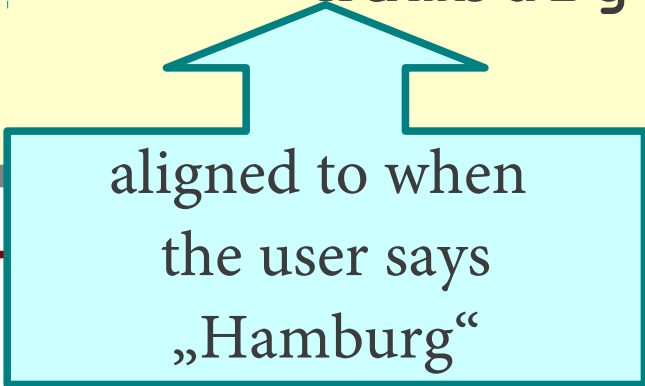
---

# Why do we need micro-timing? (and what do I mean by that?)

---

1. react more quickly  
but not too quick
2. when giving back-channel feedback
3. when interrupting a speaker:

U: Ich möchte am Samstag mit dem ICE  
Nummer 798 nach Hamburg ja klar!  
S: Hamburg, wie immer.



aligned to when  
the user says  
„Hamburg“

---

# Interrupting, Shadowing or Co-Completing the User ...

---

quick questionnaire:  
who would like this feature in a  
train-timetable information system?

please raise your hands.

---

# Interrupting, Shadowing or Co-Completing the User ...

---

My girlfriend doesn't like *any* of this either.

She says it's *no coincidence* that  
I investigate this stuff ...

... and she thinks I should be  
*forced* to use such a system.

---

# Why we really want to monitor the user's micro-timing

---

1. in some domains we might actually want to co-complete a user's turn *occasionally* (as above)
    - conversational systems, negotiation training, ...
  2. use timings to place back-channels and next turns
    - most likely in combination with prosodic cues
  3. use it to monitor the user's *fluency*:
    - how much unexpected deviation is there?
    - what does this mean for WER, understanding, ...
-

# Predicting the Micro-Timing of a User's Ongoing Words

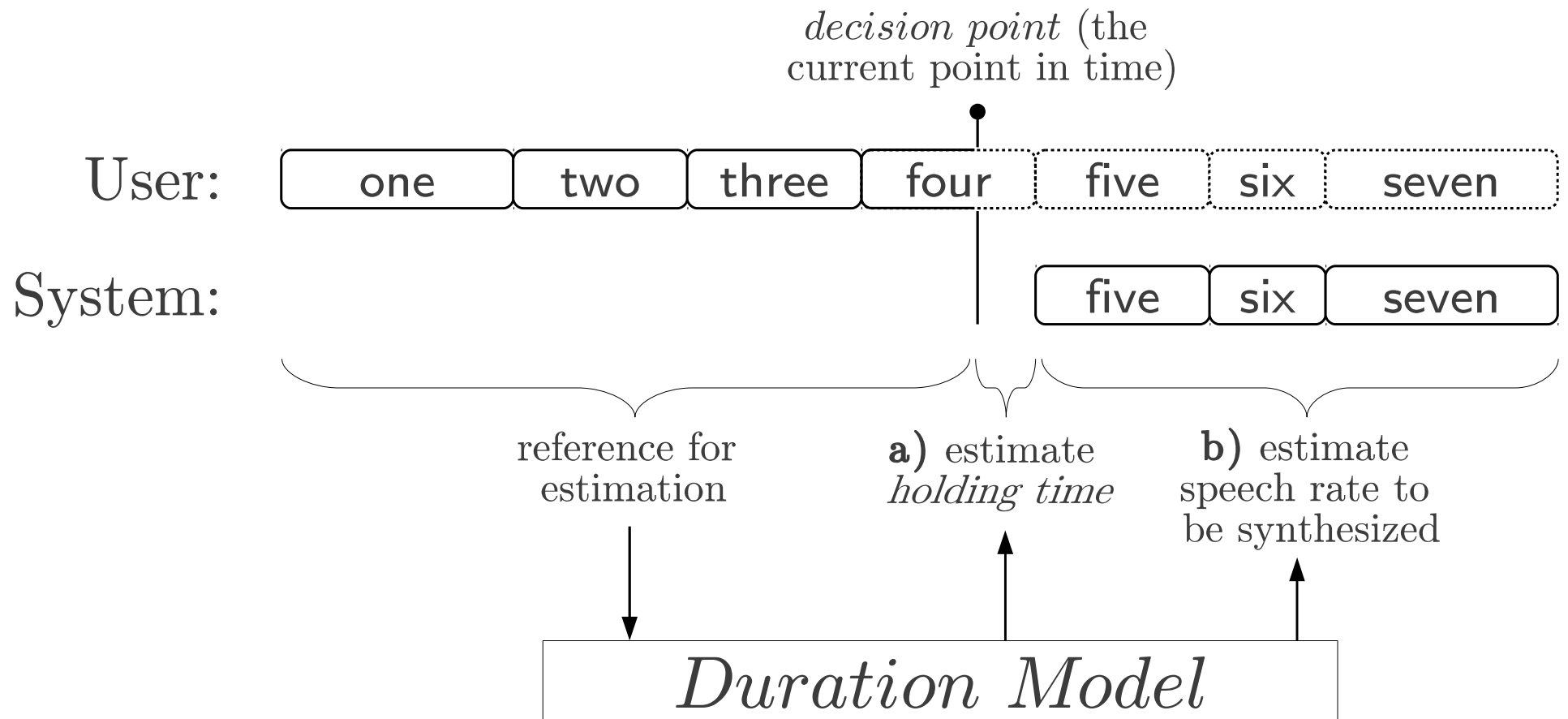
---

Our Example Task:

- let's *shadow* the user while she is speaking, i.e. say the same thing that she says and in the same way
  - we assume that she's reading a text that we know
- for this we have to
  - understand her current word before it's over
  - estimate the time remaining for the current word
  - estimate the speech rate for the next word

# A Duration-Model for Micro-Timing

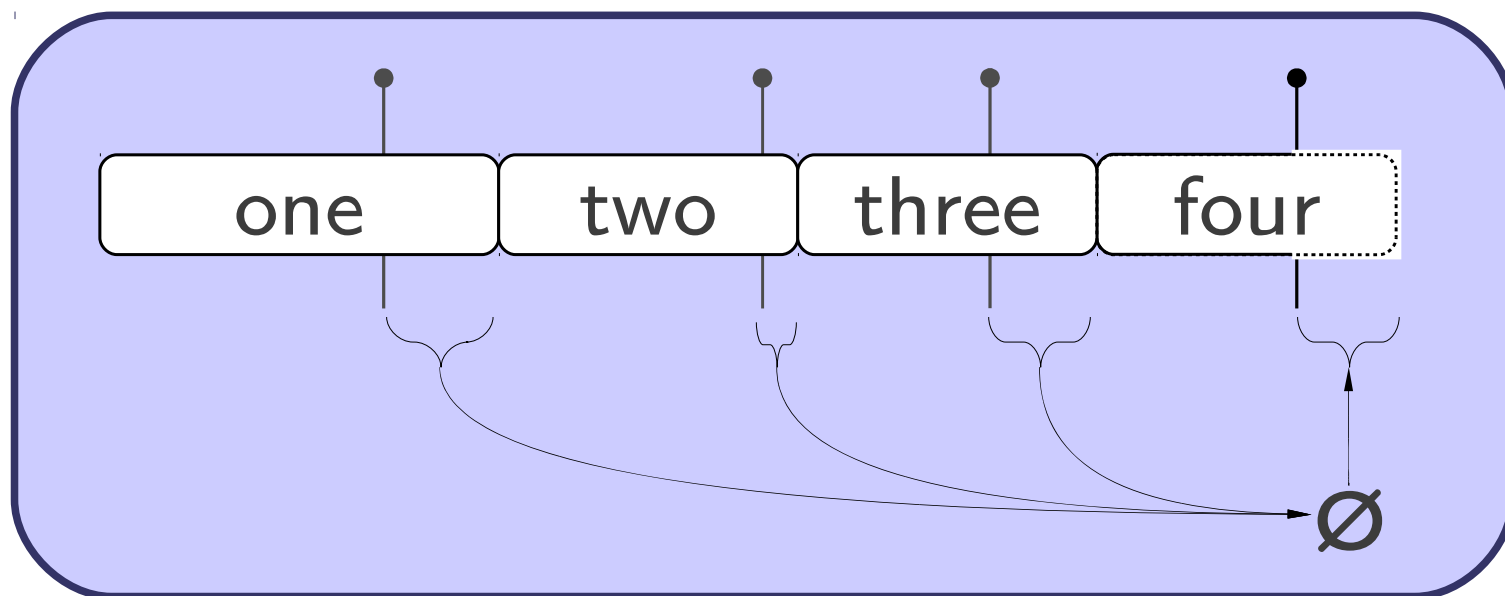
---



# Strategy 1: average ASR lookahead

---

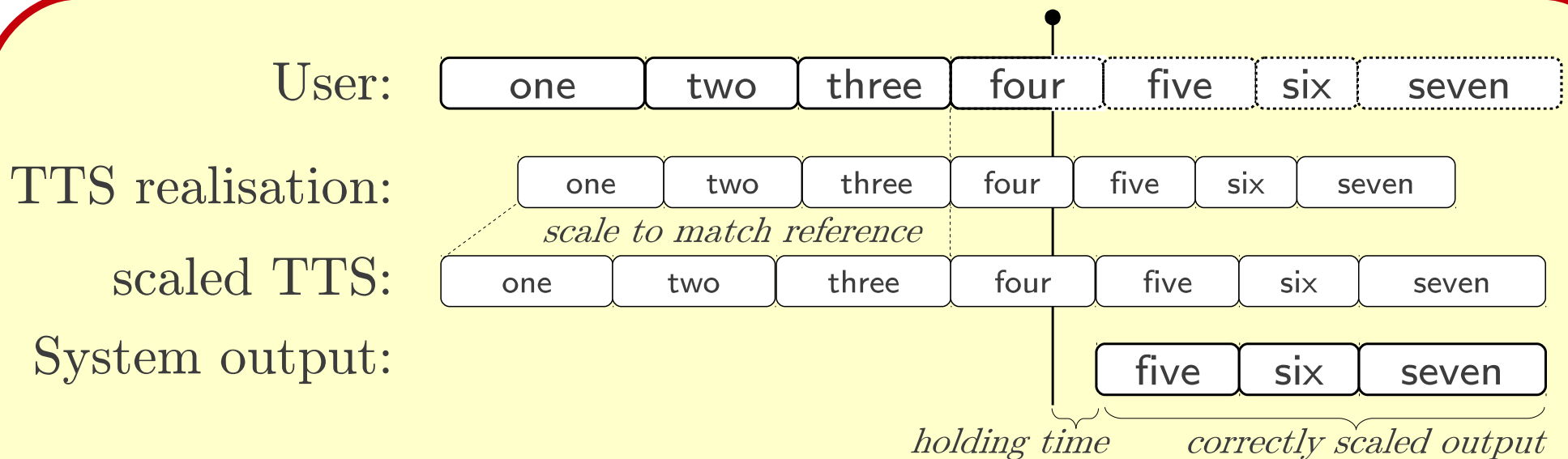
- assume, that ASR results are similarly timed
- use average lookahead as holding time



- no estimate of the next word's duration ☹️
-

# Strategy 2: Analysis-by-Synthesis

- use a TTS to (independently) estimate durations
- scale to the user's speech rate so far



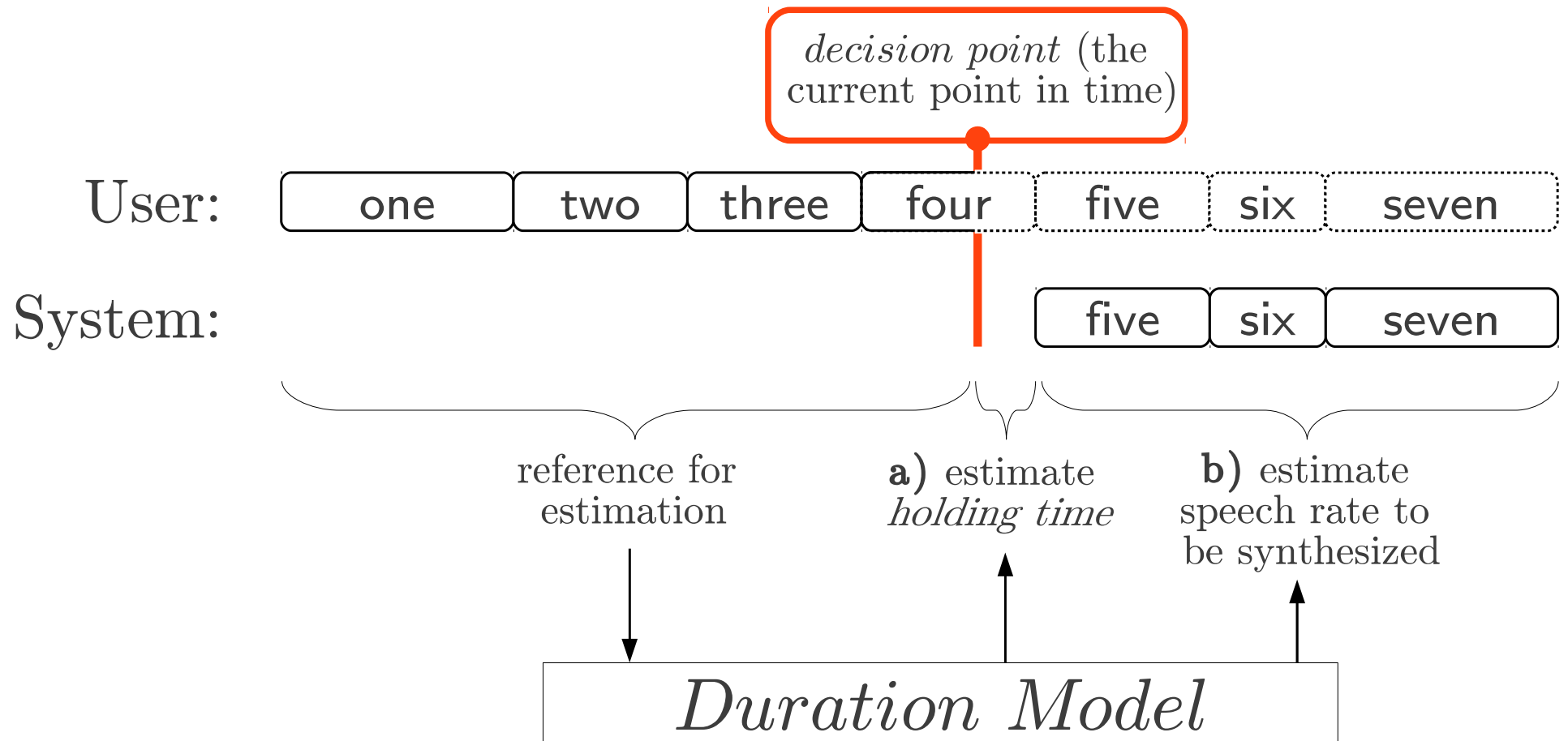
# Experiment Setup

---

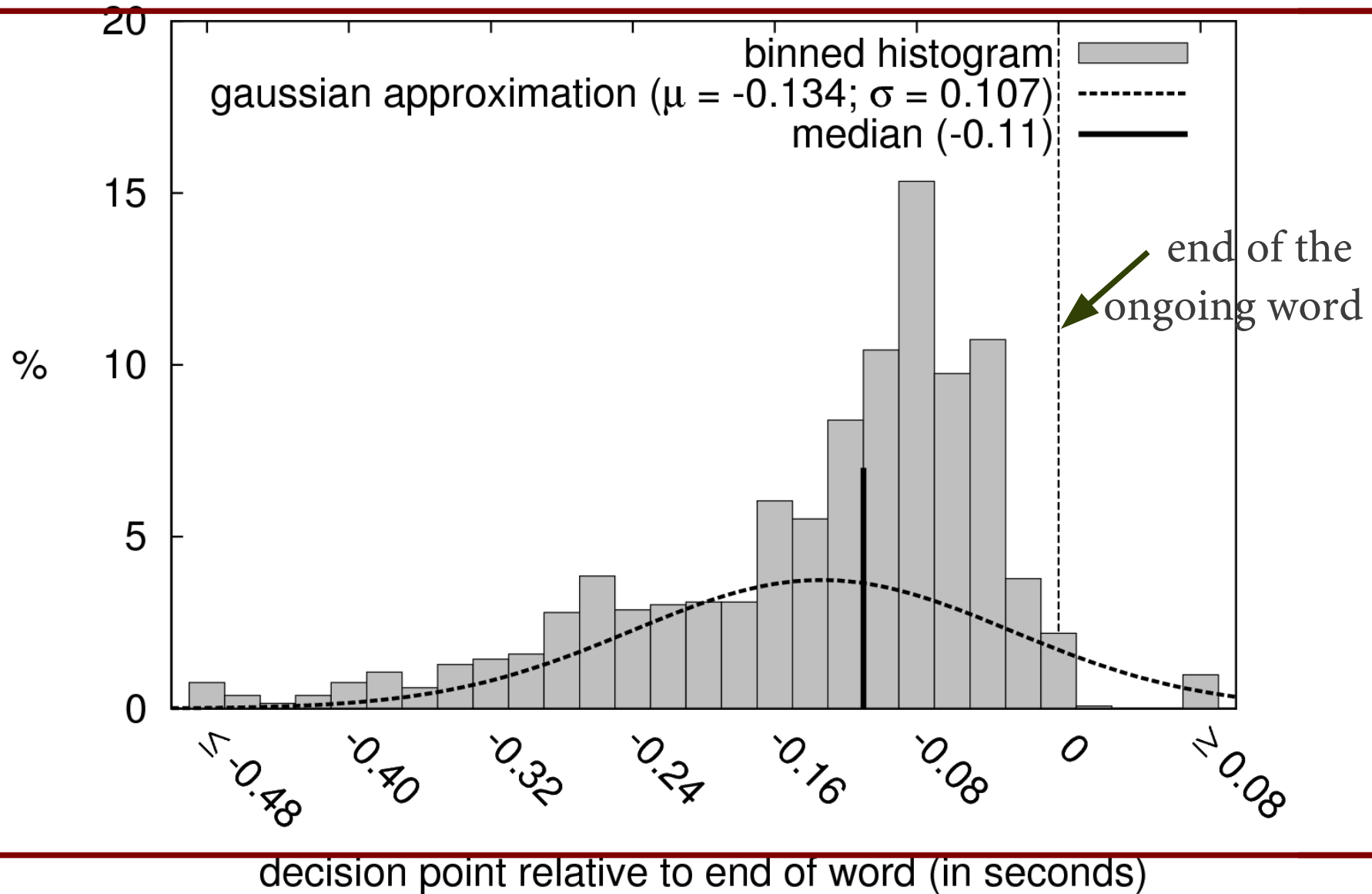
- recognize utterances from a known corpus („Nordwind und Sonne“ – *Kiel Corpus of Read Sp.*)
  - for every word:
    - how much before its end do we recognize it?
      - ♦ only if we're before the end, can we act on time
    - predict how much time is remaining (*holding time*)
    - predict the duration of the next word
    - demo: talk *in sync* with the speaker
-

# Step 1: raw ASR timing

---

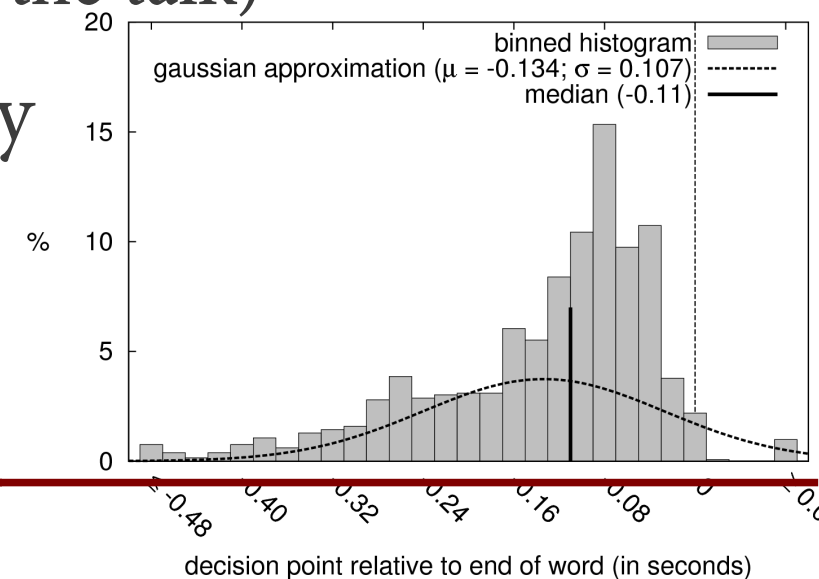


# Results: raw ASR timing

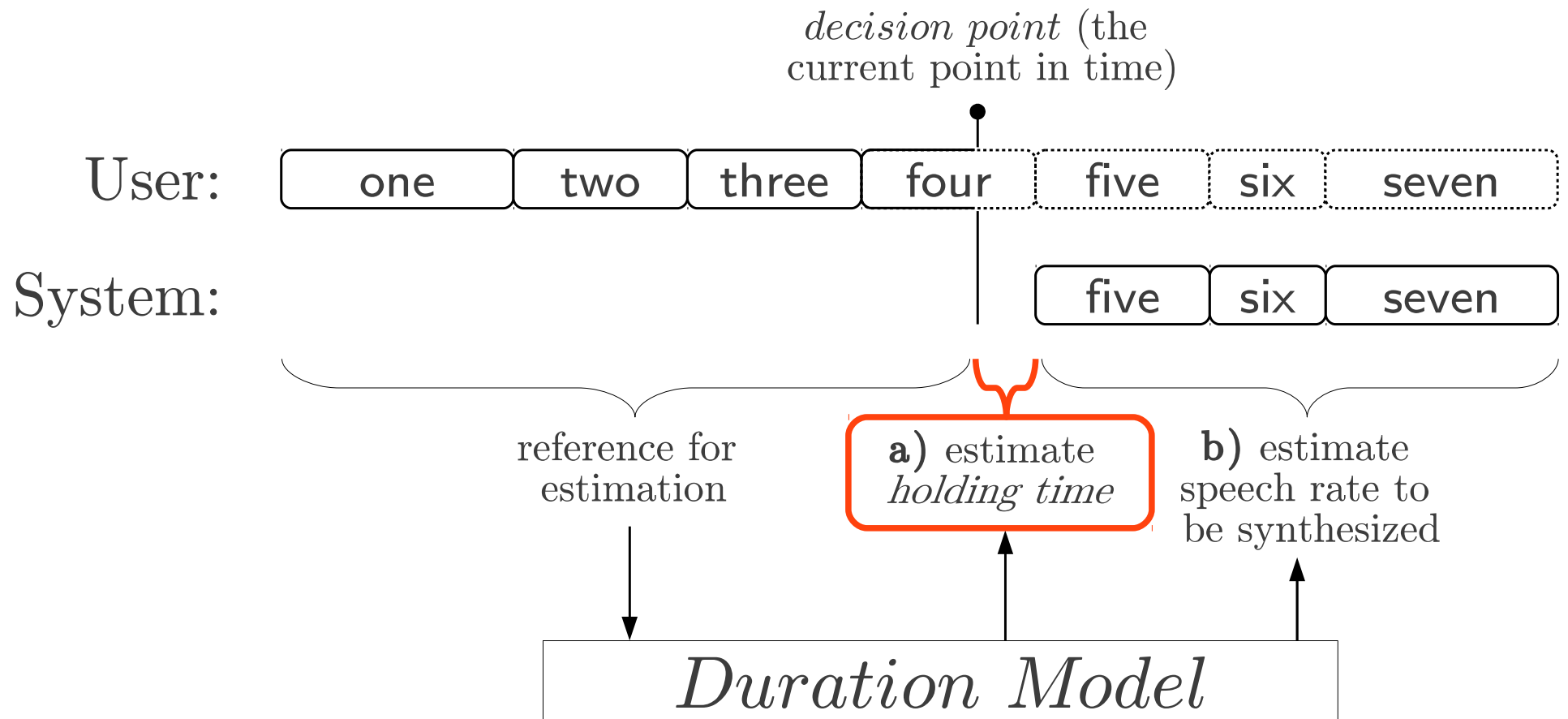


# Results: raw ASR timing

- recognition happens before the word's end (good!)
- often there is still plenty of time ( $\mu=134\text{ms}$ )
  - processing, TTS synthesis, soundcard delays, ...
  - we could use this time for ASR smoothing (as described in the first part of the talk)
- high between-speaker variability ( $\mu$  between 97 and 237 ms)



# Step 2: Holding-Time Estimation



# Results: Holding-Time Estimation

model	error distribution metrics (in ms)			
	mean	median	std dev	MAE
baseline: all	-134	-110	107	110
baseline $-\mu$	0	23	107	63
<b>ASR-based : all</b>	-2	19	105	60
IPU-internal	26	33	82	51
IPU-final	-148	-143	87	142
<b>TTS-based : all</b>	-3	4	85	45
IPU-internal	12	11	77	41
IPU-final	-78	-76	83	79

# Results: Holding-Time Estimation

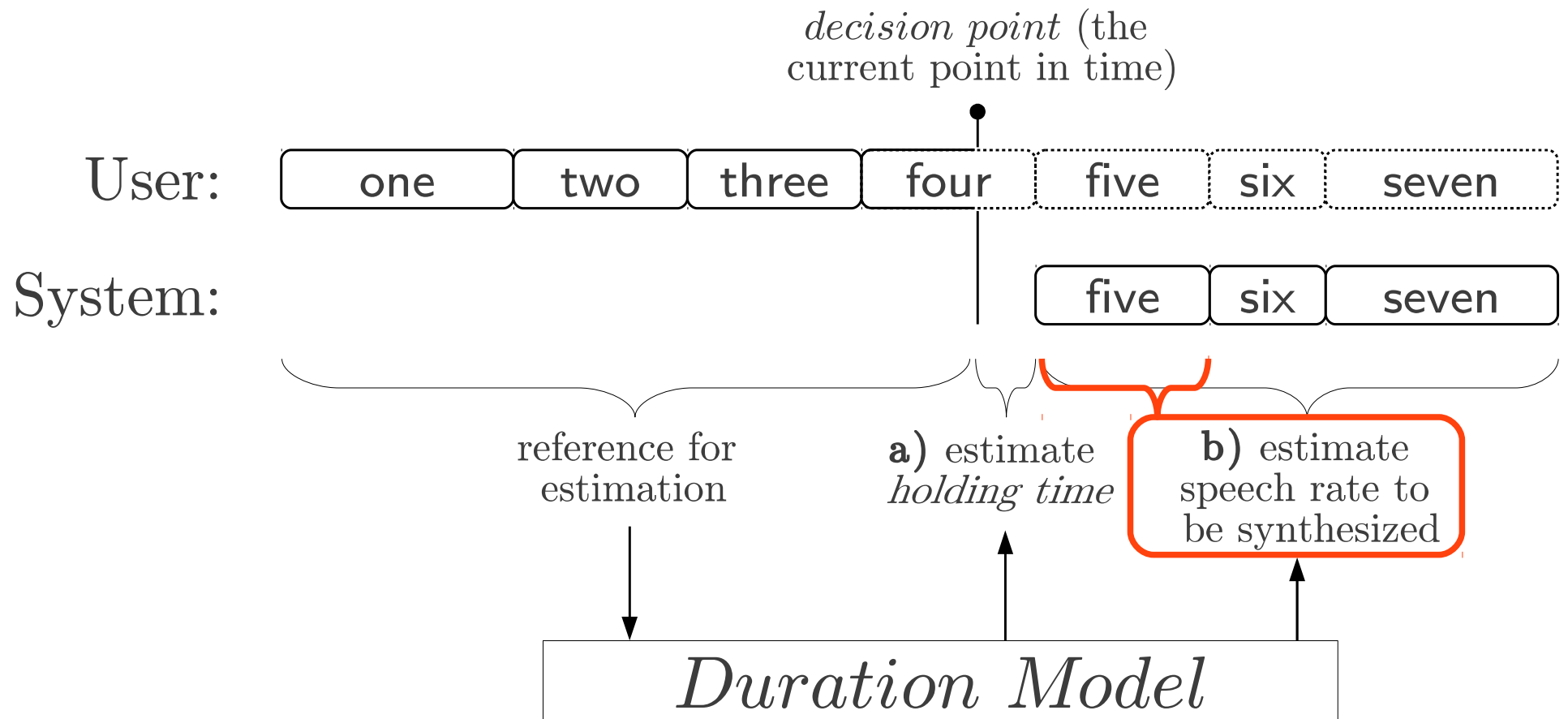
model	bias		jitter	
	mean	median	std dev	MAE
baseline: all	-134	-110	107	110
baseline $-\mu$	0	23	107	63
<b>ASR-based : all</b>	-2	19	105	60
IPU-internal	26	33	82	51
IPU-final	-148	-143	87	142
<b>TTS-based : all</b>	-3	4	85	45
IPU-internal	12	11	77	41
IPU-final	-78	-76	83	79

# Results: Holding-Time Estimation

---

- both strategies reduce bias close to zero
    - when distinguishing between IPU-internal and IPU-final words, TTS-based strategy is significantly better
  - TTS-based strategy significantly reduces jitter
    - median absolute error (MAE) similar to human performance for *synchronous speech* (Cummins 2002)
  - IPU-internal and IPU-final predictions differ
    - likely due to final lengthening
-

# Step 3: The Next Word's Duration



# Results: The Next Word's Duration

task	error distribution metric (in ms)			
	mean	median	std dev	MAE
<b>TTS-based : duration</b>	-5	4	75	45
+ <b>ASR-based : onset</b>	26	33	82	51
= end of word	25	30	100	81
+ <b>TTS-based : onset</b>	12	11	77	41
= end of word	7	10	94	74

# Results: The Next Word's Duration

---

- duration prediction (with TTS-strategy) performs almost as good as in step 2
  - however, the errors of step 2 and step 3 add up:
    - $\sigma_{\text{step 2} = 77 \text{ ms}} + \sigma_{\text{step 3} = 75 \text{ ms}} = 94 \text{ ms}$
  - deviation will likely increase for longer completions
    - underlines the need for an incremental TTS to allow instant adaptation of output as it occurs
-



# Content:

---

- ✓ Advantages of incremental SDSs
  - ✓ Architecture for incremental SDSs
  - ✓ Evaluating and Optimizing incremental ASR
  - ✓ Predicting the Micro-Timing of a User's Words
  - Wrap-up
-

# done / to do

---

1.incr. SDS architecture

2.incremental ASR

- evaluation framework
- optimizations

3.micro-timing

- of ongoing words
- of following words

• incremental TTS

- we need to be able to control synthesis on a fine-grained level

• timing error estimation

• use timing to estimate user state (through deviations in timing)

---

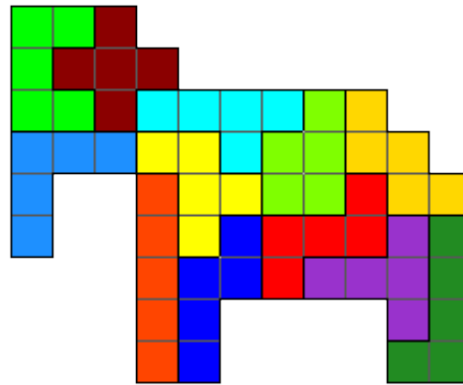
# Conclusion

---

- I hope to have convinced you that ...
    - incremental processing is vital for more natural dialogue systems
    - predictive processing is important once incremental systems improve sufficiently
    - spoken dialogue processing is fun and I'd like to work with you on these interesting phenomena!
-

# Thank you!

---



Acknowledgements:

Okko Buß and David Schlangen, my collaborators.  
DFG for funding (Emmy Noether programme)

---

# Bibliography

---

- Schlangen and Skantze: „A General, Abstract Model of Incremental Dialogue Processing“, EACL 2009.
  - Baumann, Atterer and Schlangen: „Assessing and Improving the Performance of Speech Recognition for Incremental Systems“, NAACL 2009.
  - von der Malsburg, Baumann and Schlangen: „TELIDA: A Package for Manipulation and Visualisation of Timed Linguistic Data“, SigDial 2009.
  - Baumann, Buß and Schlangen: „Evaluation and Optimisation of Incremental Processors“, Dialogue & Discourse 2(1), 2011.
  - Baumann and Schlangen: „Predicting the Micro-Timing of User Input for an Incremental Spoken Dialogue System that Completes a User's Ongoing Turn“, SigDial 2011.
-