Real-Time End-to-End Incrementality in Spoken Dialog Systems

Timo Baumann

baumann@informatik.uni-hamburg.de
http://www.ling.uni-potsdam.de/~timo
Spoken Dialog Systems Architecture

Dialog Manager

NLU

Speech Recognition

Text to Speech

domain

language generation

history
Spoken Dialog Systems Architecture

- **Speech Recognition**
- **NLU**
- **Dialog Manager**
- **Text to Speech**
- **Domain**
- **History**

**Diagram Details**:
- **Speech Recognition** processes audio input.
- **NLU** analyzes the input to understand the user's intent.
- **Dialog Manager** coordinates the dialogue flow.
- **Text to Speech** converts text into speech for output.
- **Domain** and **History** components are also indicated in the diagram.
Spoken Dialog Systems Architecture

- Speech Recognition
- NLU
- Dialog Manager
- Text to Speech
- History
- Domain
- Language generation
Spoken Dialog Systems Architecture

- **Speech Recognition**
- **Domain**
- **Dialog Manager**
- **NLU**
- **Text to Speech**
- **Language Generation**
- **History**

Flow:
- Speech Recognition → Dialog Manager
- Dialog Manager → NLU
- NLU → Dialogue Act (DA)
- DA → Domain
- Domain → SQL
- SQL → Dialog Manager
- Dialog Manager → Language Generation
- Language Generation → Text to Speech
Spoken Dialog Systems Architecture

- Speech Recognition
  - NLU
  - Dialog Manager
  - Domain
  - History
  - Language Generation
  - Text to Speech
Spoken Dialog Systems Architecture

- **Speech Recognition**
- **NLU**
- **Dialog Manager**
- **Domain**
- **Language Generation**
- **Text to Speech**

- **modules start after their predecessors have finished**
Incremental Spoken Dialog Systems

- **Partial results** are passed on and used immediately.
Benefits of Incremental Spoken Dialogue Systems

1. react more quickly as modules process input during a speaker’s turn:

U: I want to go on Saturday from Chicago to El Paso to visit my son.
S: Ok, at what time do you want to go?

(Crafted examples for an imaginary air travel information system.)
Benefits of Incremental Spoken Dialogue Systems

1. react more quickly as modules process input during a speaker’s turn:

U: I want to go on Saturday from Chicago to El Paso to visit my son.
S: Ok, at what time do you want to go?

sufficient information: Saturday, CHI → ELP
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: I want to go on Saturday with flight number, uhm ... hold on ... CO798 ...
S: yea? 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: I want to go on Saturday with flight CO798 to, uh ...

S: Sorry, there's no flight with that # on Saturdays. Do you want to go to El Paso?
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
Advantages of incremental SDSs

Architecture for incremental SDSs

- Predicting the Micro-Timing of User Input
- A demonstration of End-to-End Incrementality: Co-Completing a User’s Ongoing Turn
Incrementality in SDS

- incremental hypotheses are only preliminary
  - hypotheses change with time (some changes are errors)
  - (show video)

- the architecture must support changes to hypotheses

(Baumann et al., 2009)
An Architecture for Incremental SDS

- Modules in the system are connected via buffers

(Schlangen and Skantze, 2009)
An Architecture for Incremental SDS

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

(Schlangen and Skantze, 2009)
An Architecture for Incremental SDS

- 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

(Schlangen and Skantze, 2009)
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
IU Network

- different IU types on different levels to denote different kinds of information
  - DAs
  - words
  - phonemes
IU Network

• belief changes lead to changes in the network
  ▪ a new frame arrives
  ▪ the word hypothesis is revoked …
IU Network

- belief changes lead to changes in the network
  - a new frame arrives
  - the word hypothesis is revoked and replaced by a different one
IU Network

- belief changes lead to changes in the network
  - changes trickle up in the system
  - higher-level reasoning might lead to changes trickling down
Content:

✔ Advantages of incremental SDSs
✔ Architecture for incremental SDSs

● Predicting the Micro-Timing of User Input

● A demonstration of End-to-End Incrementality: Co-Completing a User’s Ongoing Turn
Timeliness, Incrementality and Prediction

- basically, we just want our dialog systems to be on time
- incrementality is just a way to achieve this goal

_can we design systems that are fast enough to achieve good, timely behaviour?_
Real-time End-to-End Incrementality

- I'll present work that shows that
  - we can predict the *micro-timing* of words
    - incrementality makes statements about the recent past
    - predicting the near future is actually very similar
  - use this information to build a system that co-completes the user (i.e. says what the user is saying at the same time that the user is saying it)
    - leaving out the „high-level“ dialog management, i.e., just assuming that we know the completion of the utterance (see DeVault et al. (2009) for how to do that)
Why do we need micro-timing? (and what do I mean by that?)

1. react more quickly, but not too quick:

U: I want to go on Saturday from Chicago to El Paso to visit my son.
S: Ok.

sufficient information: Saturday, CHI→ ELP
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:

- we want the back-channels to be precisely aligned

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: I want to go on Saturday with flight CO798 to El Paso ... right!
S: El Paso, as always.

aligned to the user saying „El Paso“
Co-Completing the User

- computers should certainly not *always complete* a turn that they understand (not even often)
- however, this can be an efficient interactional device if used occasionally in certain situations
  - conversational systems, negotiation training, …
Co-Completing the User

- computers should certainly not always complete a turn that they understand (not even often)
- however, this can be an efficient interactional device if used occasionally in certain situations
  - conversational systems, negotiation training, …
- frequency of occurrence in human dialogue:
  - sentence cooperations in task-oriented German: 3.4 %
  - split utterance boundaries in the BNC: 2.8 %

(Skuplik 1999; Purver et al., 2009)
The 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*
  - identical to *synchronous reading task* (Cummins 2002)
The 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
  - identical to *synchronous reading task* (Cummins 2002)

- to be able to shadow we have to
  1. identify the user’s current word before it’s over
  2. estimate the time remaining for the current word
  3. estimate the speech rate for the next word
Incremental ASR is very fast

- *when* does the ASR notice words?
  - first intuition around \( \frac{3}{4} \) into the word
  - final recognition around end of the word

(Baumann et al., 2009)
Incremental ASR is very fast

- *when* does the ASR notice words?
  - *first intuition* around $\frac{3}{4}$ into the word
  - *final recognition* around end of the word

\begin{tabular}{|c|c|c|}
  \hline
  a & few & words \\
  \hline
\end{tabular}

(Baumann et al., 2009)
Incremental ASR is very fast

- *when* does the ASR notice words?
  - **first intuition** around \( \frac{3}{4} \) into the word
  - **final recognition** around end of the word

![Diagram](Baumann%20et%20al.,%202009)
The Task

User:

System:

\textit{decision point} (the current point in time)

reference for estimation

a) estimate \textit{holding time}

b) estimate speech rate to be synthesized
Shadowing *iteratively* word-by-word

User:

one two three four five six seven

System:

handle next word now ...

decision point

reference for estimation

... remaining words later
We need a duration model

- given some partial input (words and durations)
- and the expected completion (words, no durations)
- assign expected durations for the completion
We need a duration model

- given some partial input (words and durations)
- and the expected completion (words, no durations)
- assign expected durations for the completion

- What model can generate the canonical durations?
  - hey, TTSs have very good duration models!
  - and we need a TTS to synthesize a completion anyway
Strategy: Analysis-by-Synthesis

- listen to what is being said (prev.words, curr.w.), predict what will be said (compl.),
- feed combined full utterance to (symbolic) TTS

\[
\text{scaling factor} := \frac{\text{length}_{User}(\text{prev.words})}{\text{length}_{TTS}(\text{prev.words})}
\]

\[
\text{holding time} := \text{length}_{TTS}(\text{curr.w}) \times \text{scaling factor} - \text{length}_{User}(\text{curr.w})
\]

- scale completion with scaling factor, send to (acoustic) TTS
- play output at predicted time
**Strategy**: Analysis-by-Synthesis

User:

TTS realisation:

scaled TTS:

System output:

one two three four five six seven

scale to match reference

one two three four five six seven

holding time correctly scaled output
Experiment Setup

- recognize utterances from a fixed corpus („Nordwind und Sonne“ – *Kiel Corpus of Read Sp.*)
- for every word:
  - how long before its end do we recognize it?
    - because 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
Results

• words are recognized sufficiently early ($\mu = -134$ ms)
• errors in holding time prediction and next words’ duration are significantly reduced by Analysis-by-Synthesis method (std dev = 85 ms / 77 ms / 94 ms)
• median absolute error (MAE = 74 ms) is close to human performance for synchronous speech (56 ms)

… alright, but how does it sound? (Cummins 2002)
Excerpt from “The Northwind and the Sun”

Endlich gab der Nordwind den Kampf auf. Nun erwärmte die Sonne die Luft mit ihren freundlichen Strahlen und schon nach wenigen Augenblicken zog der Wanderer seinen Mantel aus.

At last the North Wind gave up the attempt. Then the Sun shined out warmly, and immediately the traveler took off his cloak.

(IPA 1999; drawing by Milo Winter, 1919, public domain)
How does it sound?
Conclusions

- real-time end-to-end incrementality is feasible
- we can predict the user’s current and next words’ durations
- we can synchronize to the user’s speech with close-to-human performance
Ongoing and Future Work

- the predictions should include an error estimate
  - only co-complete if you're sufficiently sure to do it right
- the demo shows that the current synthesis method is far from satisfactory
  - I'm currently focussing my work on incrementalizing speech synthesis
- currently, TTS and ASR are separate components; I think they should be merged for future SDS
Thank you!

Questions and Comments?

Thanks also to David Schlangen, Okko Buss, Petra Wagner, and Benjamin Weiss