Toward Interactive Dictation

Belinda Z. Li, Jason Eisner, Adam Pauls, Sam Thomson

Microsoft Semantic Machines Massachusetts Institute of Technology

Abstract

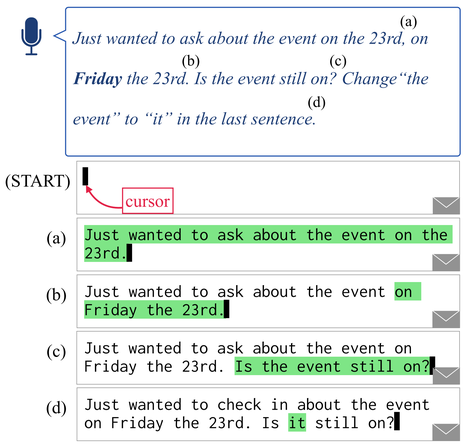
- Current systems for dictation and editing-by-voice only support (1) *limited, inflexible* edit commands, which (2) must be invoked through *trigger words*.
- We introduce a new task, **Interactive Dictation**, that addresses these 2 limitations. We allow users to interrupt their dictation with spoken editing commands in *open-ended* natural language.
- We build a novel data collection interface and collect a dataset for this task, **TERTiUS**.
- We build baseline systems for the task.
- Code and data will be released: `https://aka.ms/tertius`

Introduction

We want to support both *transcription* and *editing* through speech. How do we build an intuitive system that allows users to *flexibly* interleave dictation and editing? How may users invoke *open-ended* edit commands?

⇓

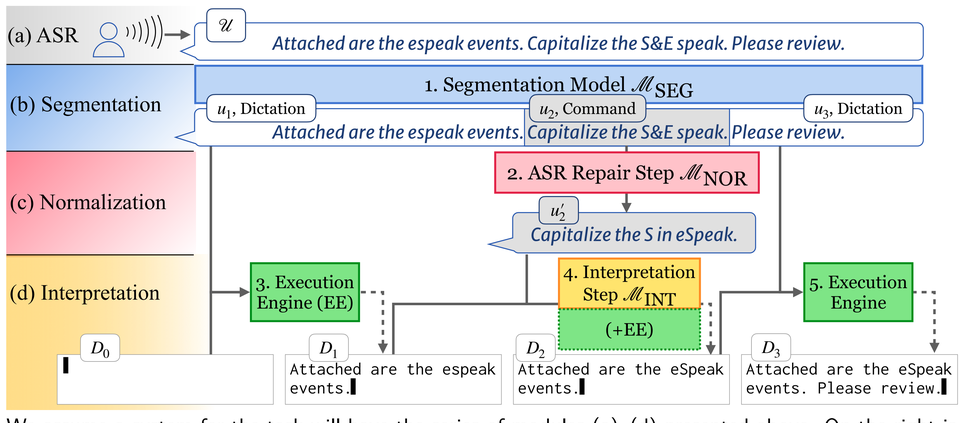
Interactive Dictation



Challenges:

- No reserved trigger words for invoking commands → **segmentation** (are we dictating or commanding? has a new command been invoked?) is nontrivial
- No fixed templates for commands → **interpretation** (which commands to invoke?) is nontrivial.

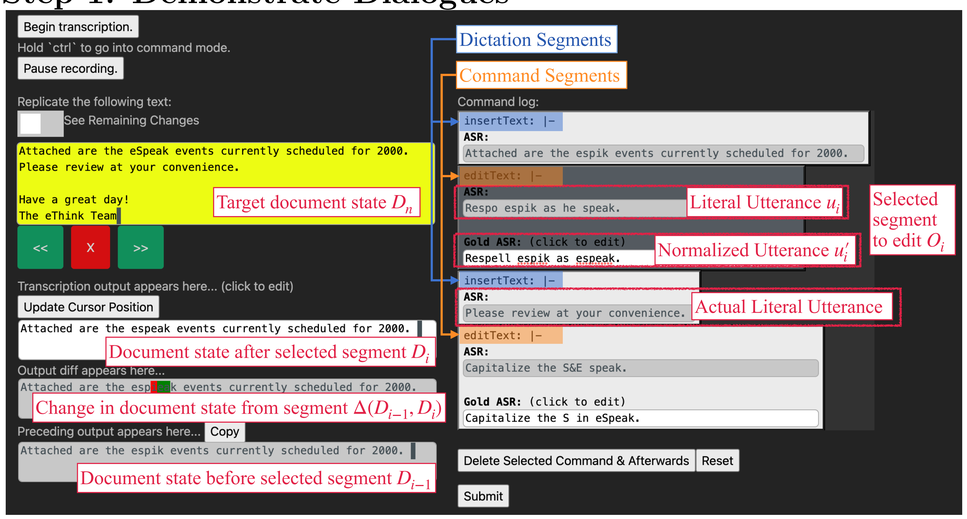
Task Overview



We assume a system for the task will have the series of modules (a)–(d) presented above. On the right is the concrete instantiation of *our* system.

Data Collection

Step 1: Demonstrate Dialogues



Step 2: Annotate Programs for Command Segments

Normalized Command Segment u_i' — *Capitalize the S in eSpeak*

⇓ ⇓

Program p_i

```
(capitalize
 (theText
   (and
     (like "S")
     (in (theText (like "eSpeak"))))))
```

Dataset: TERTiUS

Task	Description	Dialogues	Segments Dict.	Cmd.	Total
Replicate doc	Exactly recreate an email	372	473	1453	1926
Elaborate doc	Expand a terse description of an email to a full email	343	347	473	820
Replicate segment	Exactly recreate the effect of a single command segment sampled from demonstrations of the previous two objectives	605	139	1299	1438
Total		1320	959	3225	4184

Table: Dataset size statistics.

- How diverse/flexible is the dataset?

Command action	Distinct 1^{st} tokens (TERTiUS)	Distinct 1^{st} tokens (DNS)
replace	83	-
delete	22	5
insert	51	1
correction	22	1
lowercase	12	1
...		

Table: Number of ways to invoke various commands, in terms of number of distinct first tokens used to invoke that command. Comparing TERTiUS to prior systems (Dragon NaturallySpeaking)

Models

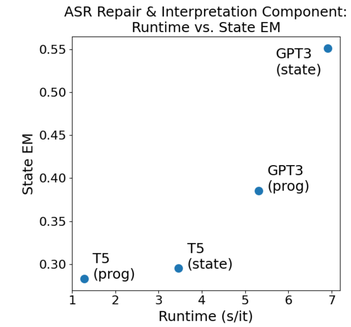
- $\mathcal{M}_{\texttt{SEG}}$: T5 encoder trained to perform BIOES tagging to identify command boundaries.
- $\mathcal{M}_{\texttt{NOR}}$: T5 encoder-decoder model trained to map noisy ASR segments into normalized ASR segments (repairing ASR/speech errors).
- $\mathcal{M}_{\texttt{INT}}$: Maps normalized command ASR segments into either: (1) *prog*: programs which get executed by an execution engine into the end-state, or (2) *state*: the end-state directly.

Experiments

	Metric	T5	GPT3
Segmentation	F1	90.9%	-
	Seg EM	85.3%	-
	Runtime (s/it)	0.097	-

		prog	state	prog	state
ASR Repair + Interpretation	State EM	28.3%	29.5%	38.6%	55.1%
	Program EM	28.3%	-	41.9%	-
	Runtime (s/it)	1.28	3.46	5.32	6.92

We evaluate segmentation (top) and the ASR repair and interpretation components jointly (bottom), reporting accuracy metrics (F1, EM) as well as runtime (in seconds per example). For ASR repair and interpretation, we experiment with a fine-tuned T5 vs. a prompted GPT3 model, each outputting either the end state (state) or a program to carry out the command (prog).



Runtime vs. State EM of various repair & interpretation models. Comparing GPT3 vs. T5 and prog vs. state models. Both increasing model size and predicting state is more accurate, at the expense of runtime.