Contact: matthias.sperber@kit.edu

Optimizing Computer-Assisted Transcription Quality with Iterative User Interfaces

Matthias Sperber[1], Graham Neubig[2], Satoshi Nakamura[2], Alex Waibel[1]

[1] Karlsruhe Institute of Technology, Germany; [2] Nara Institute of Science and Technology, Japan

Motivation

Transcription of speech is an important step in making language resources.

Want to improve your transcription resource quality, or create it more cheaply?

Want to make your less skilled workers produce high-quality transcriptions?

This research: design transcription UI to achieve this.

Experiments

Set Up

- 9 participants, different skills:
  - 5 crowd, 2 linguists, 2 computer scientist
- 4 TED talks
  - segmented into short utterances
- Randomly alternate between the 4 UIs
- Statistical analysis: mixed effects model

Mixed Effects Models

- Problem: random factors
  e.g. particular transcriber, test data
- Traditional solution: average over lots of them
- Better: mixed effects model, explicitly account for rand. factors:
  - Response variable *e.g. WER-reduction*
  - Fixed effects *e.g. UI, segment length…*
  - **Random effects** *transcriber, talk, segment ID*
  - Noise term

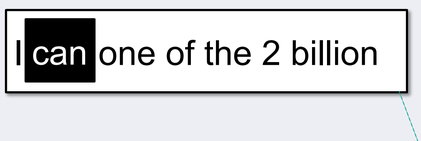
User Interface Designs

FS-

Traditional from-scratch
- High effort
- No guidance from ASR

PE-

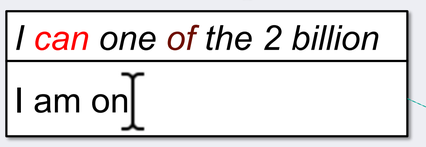
Traditional post-editing
- Less effort
- ASR aids recognition and short-term memory
- Errors sometimes overlooked



FS+

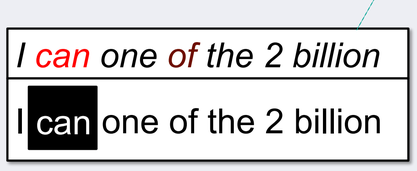
retyping (*type from-scratch while reading initial suggestion)*
- ASR guidance
- Enforces concentrated work, less errors overlooked
- Word confidences highlighted
- High effort, like FS-

Novel interface



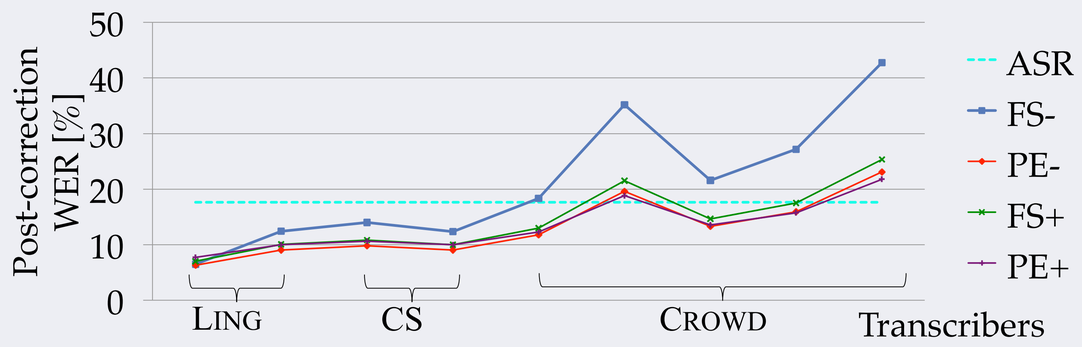
PE+

Extended post-editing
- Similar to PE-
- Plus confidence highlights

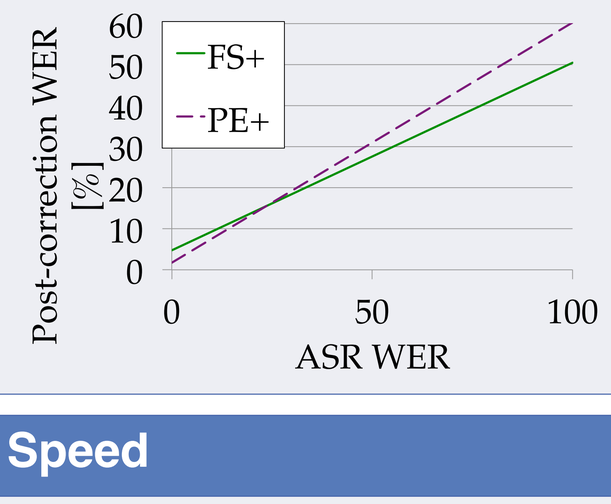


Findings: Quality

Method	Initial ASR	FS-	PE-	FS+	PE+
Resulting word error rate	17.6	21.2	13.1	14.4	13.4
Resulting phoneme error rate	15.4	13.3	8.2	9.2	8.3

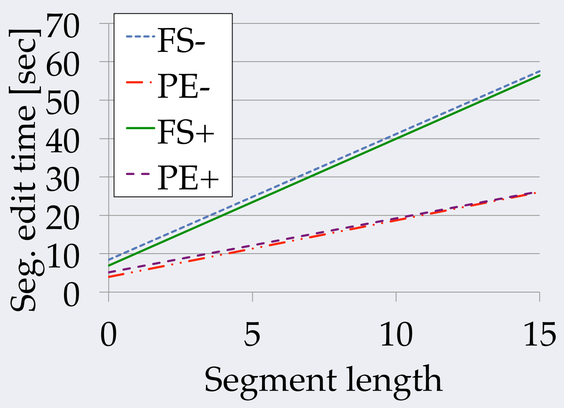


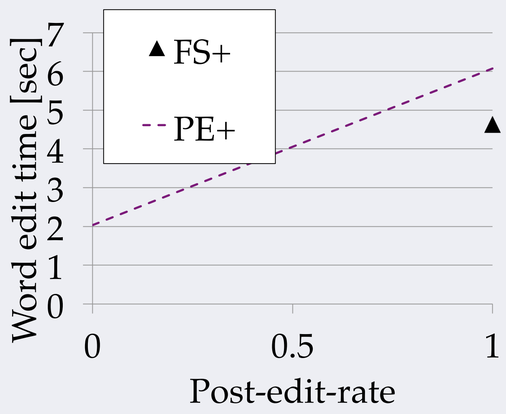
"Iterative" interfaces → crucial for crowd, and still helpful for skilled workers



→ *Retyping*: better quality especially for the more critical segments

Findings: Speed





→ *Post-editing* usually faster than *retyping*

(except when almost every word needs to be changed)

Human Correction Errors

FS+ time per typed word	**PE+** time per *unedited* word	**PE+** time per *edited* word
4.46 s	1.91 s	5.81 s

In addition: Errors *after* human correction made correction process **.98 seconds/error** slower
→ useful to detect human correction errors?