# Bringing Neural Spelling Correction to Mobile Keyboards Using a Client-Server Architecture

Hagen Tilmann Mogalle

University of Freiburg
Chair of Algorithms and Data Structures

27.08.2025

# Problem – Background

**Why spelling correction matters on mobile devices?**

- Germany: 2.8 billion chat messages per day [1]
- Error sources: Small keys, fingers cover letters
- Distractions: typing while walking, single-hand typing

**Users expect high correction quality**

[1]: Statista, 2025

# Problem – Spelling Correction

Example: Word-level vs. Sentence-level Correction

| Input | Correction Type | Output |
|---|---|---|
| Whats your naem? | Word-level | [*What's, name*] |
| | Sentence-level | *What's your name?* |

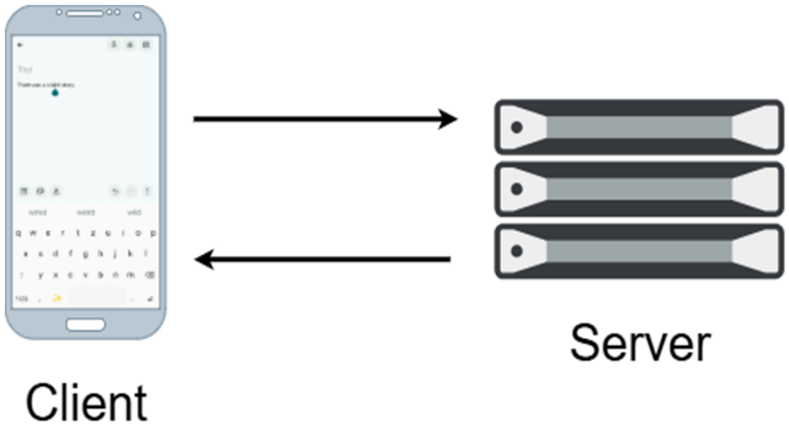
# Problem – Status Quo

**Neural models are accurate, but heavy…**

- Too large for smartphones (memory, compute, battery)
- Commercial keyboards use smaller local models → limited quality

**Solution: Client-Server Architecture?**

- Minimal computation on-device
- **But: Additional network latency**



# Problem

**Research question:**

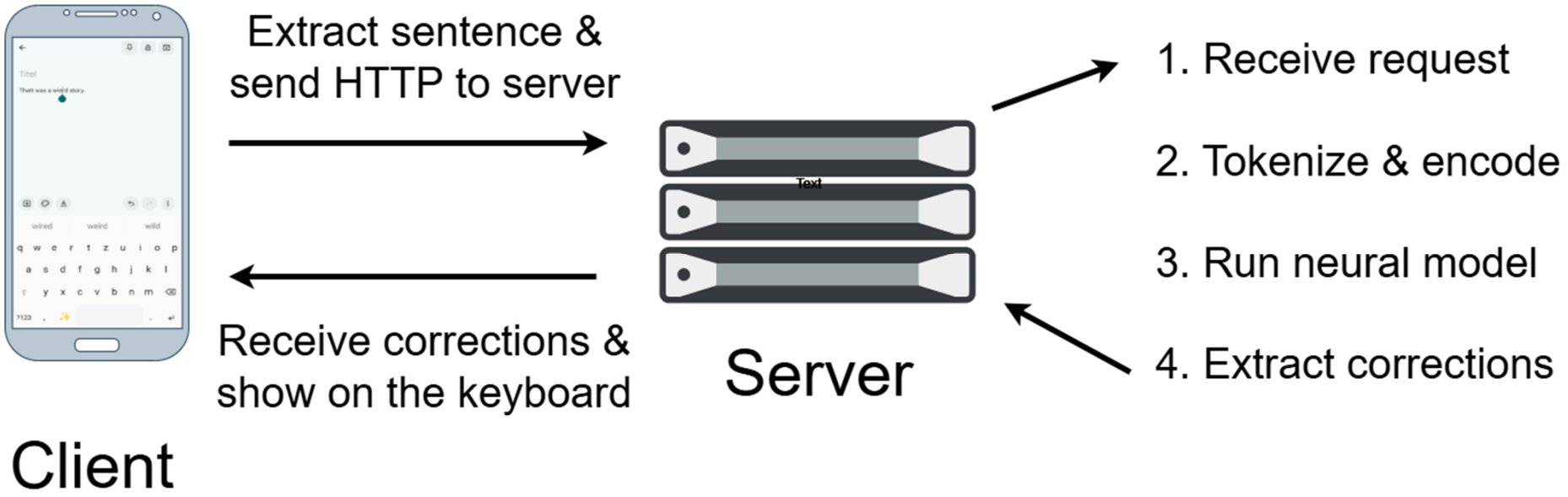
- Can correction be outsourced to a server?
- Is latency acceptable to users?
  - Word-level Correction
  - Sentence-level Correction

# Problem

Questions?

# Solution – Correction Pipeline

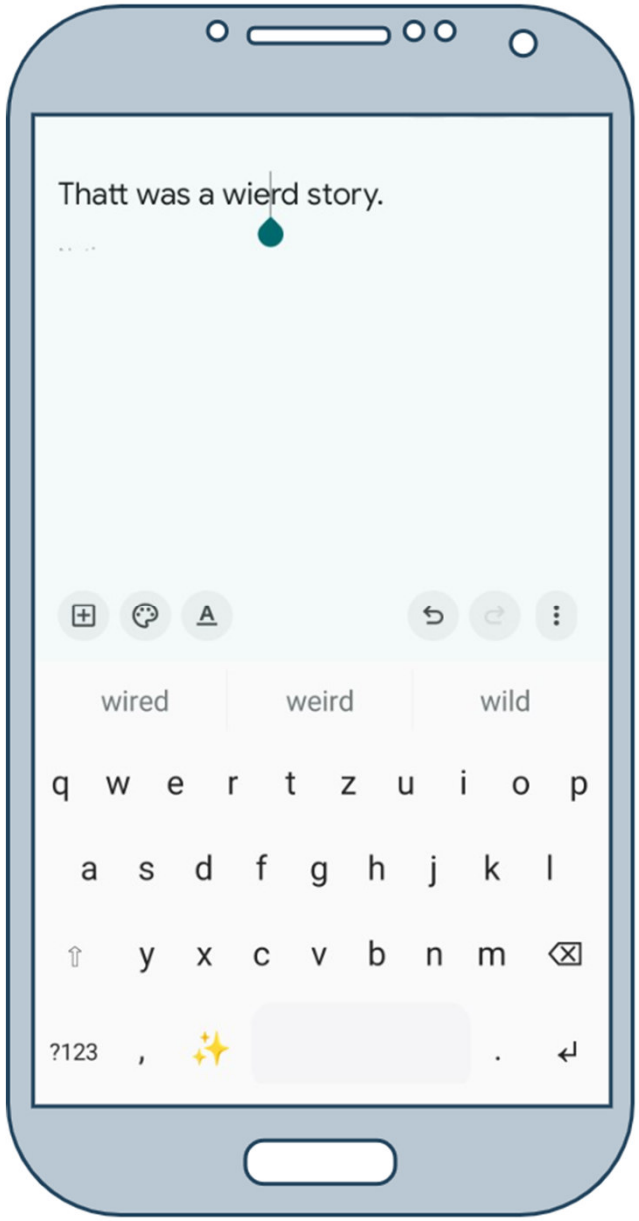
## Correction Pipeline



# Solution – Keyboard Implementation

## Keyboard Implementation

- Suggestion bar above keys
- **Word-level correction**:
  - trigged on each keystroke
  - up to 3 corrections
- **Sentence-level correction**:
  - triggered on sentence ending character & Magic Key ✨
  - Toggle between 3 alternatives




Example of word correction

# Solution – Server

**Neural model**

- Encoder-Decoder Transformer model (interchangeable)
- Beam size k=3 → outputs three sequences

**Word Extraction**

Difflib.SequenceMatcher: detects insertions, deletions, and replacements

Model input:

1. I can't find teh file.

Model output:

1. I can't find the file.
2. I can't find that file.
3. I can't find these file

→ Position 3: [the, that, these]

# Solution

Question?

# Evaluation

## Evaluation

**Goal**: Assess user acceptance of the server-client architecture

1. Technical Evaluation
   - Latency
   - Correction accuracy vs. Gboard

2. User Study:
   - Acceptance of Word-level & Sentence-level Correction

# Evaluation – Technical Evaluation

## Setup

- Server: Uni-Host, Nvidia RTX 4090, Remote (VPN), Wi-Fi
- Client: Google Pixel 9 Pro XL, Android 16
- Data: 500 Sentences from Reddit Dataset, artificial errors
  - End-to-End-Latency: Subset of 50 sentences

# Evaluation – Technical Evaluation

**Model inference time (isolated)**

| Beam Size | Mean (ms) | Std (ms) |
|---|---|---|
| 1 | 13.92 | 3.40 |
| 3 | 41.75 | 10.73 |

→ Beam size 3 tripples the inference time

# Evaluation – Technical Evaluation

**End-to-End latency (Beam size k = 3)**

| Component | Mean (ms) |
|---|---|
| RTT (network latency) | 83.28 |
| Server processing (SPT) | 44.97 |
| - Model inference | 44.70 |
| - Post-processing | 0.26 |
| End-to-end total | 128.25 |

→ RTT dominates, Post Processing negligible

→ For reference: Lu's local n-gram based keyboard: 69 ms

# Evaluation – Accuracy

## Correction Accuracy vs. Gboard

**Dataset:** 50 sentences (Reddit), 111 spelling errors

**Methodology:** Insert whole sentence, move cursor on misspelled words, count correct suggestions

| | Top-1 | Top-3 |
|---|---|---|
| Gboard | 57.7% | 66.6% |
| Our keyboard | 83.8% | 89.2% |

→ Significantly higher error correction accuracy, in both Top-1 & Top-3 corrections

# Evaluation – User Study

**Design**

- Participants: n = 7
- Two Phases:
  - Phase A: Word-level correction
  - Phase B: Sentence-level correction

**Task**

- Copy 10 sentences per phase
- Fix self-made errors using correction mode
- Measures: 5-point Likert-scale questionnaire after each phase

# Evaluation – User Study

## Questionnaire Answers

| Question focus | Word-level correction (⌀) | Sentence-level correction (⌀) |
|---|---|---|
| Comfort | 3.43 | 4.14 |
| Typing speed | 2.86 | 3.71 |
| Correction quality | 3.57 | 4.29 |
| Cognitive load | 2.43 | 3.00 |
| Latency | 3.71 | 4.14 |
| Higher latency for higher accuracy | 2.57 | 3.57 |
| Overall Satisfaction | 3.29 | 4.00 |
| Daily use | 3.43 | 3.71 |

# Evaluation – User Study

**User Study Results**

- Both modes work and are accepted (Overall satisfaction ~3.3/5 & 4.0/5)

**Word-level correction**:

- Feels slower & moderate correction quality

**Sentence-level correction**:

- Feels faster
- Corrections perceived as very helpful
- Latency well accepted; willingness to accept more for higher correction quality

# Evaluation – Conclusion

**Conclusion**

1. End-to-end latency: ~128 ms / sentence

2. Outperform Gboard in correction accuracy: (67% vs. 89%)

3. Sentence-level correction especially accepted
   - Latency not perceived as disruptive

→ **Client-Server Architecture for spelling correction feasible and accepted by users**

# Evaluation – Future Work

**Future Work**

- **Latency thresholds:** Larger study to define limits of acceptable latency
- **Word correction extraction:** Alignment (esp. concatenated words)
- **Better UI design to evaluate correction modes in isolation**

# Evaluation

Question?

# Extras – Model Parameters

| Parameter | Value |
|---|---|
| Architecture | Transformer Encoder-Decoder |
| Encoder Layers | 2 |
| Decoder Layers | 2 |
| Attention Heads | 5 |
| Hidden Dimension | 256 |
| Embedding Dimension | 256 |
| Total Parameters | 12,533,312 |
| Tokenizer | BPE (Byte-Pair Encoding) |
| Vocabulary Size | 8,000 |
| Max Sequence Length | 35 tokens |
| Dataset | English Reddit (subset) |
| Training Sentences | 64 million |
| Batch Size | 256 |
| Learning Rate | 0.001 |
| Optimizer | Adam |
| Epochs | 2 |
| Deployment | ONNX format |
| Beam Search Width | $k = 3$ |

# Extras – User Study Questionnaire

| Mode B | | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
|---|---|---|---|---|---|---|
| 8 | Typing in this mode was comfortable. | 1 | 2 | 3 | 4 | 5 |
| 9 | I was able to type the sentences quickly. | 1 | 2 | 3 | 4 | 5 |
| 10 | The corrections were helpful. | 1 | 2 | 3 | 4 | 5 |
| 11 | I often had to think whether I should accept a correction. | 1 | 2 | 3 | 4 | 5 |
| 12 | The corrections appeared fast enough to be useful. | 1 | 2 | 3 | 4 | 5 |
| 13 | I would prefer more accurate corrections, even if they take slightly longer. | 1 | 2 | 3 | 4 | 5 |
| 14 | Overall, I am satisfied with this mode. | 1 | 2 | 3 | 4 | 5 |

| Final Questions | | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
|---|---|---|---|---|---|---|
| 13 | I would use mode A for daily use. | 1 | 2 | 3 | 4 | 5 |
| 14 | I would use mode B for daily use. | 1 | 2 | 3 | 4 | 5 |
| 15 | I would use this keyboard regularly even if it requires continuous internet connection. | 1 | 2 | 3 | 4 | 5 |
| 16 | I am comfortable with my typed text being processed on a remote server. | 1 | 2 | 3 | 4 | 5 |

17. What would stop you from using this keyboard in everyday life?

# Extras – Latency

Nielssen:

- 100 ms: feeling of instantaneous, direct interaction
- 1 s: uninterrupted flow of thought, a perceived delay

Lu:

- 200 ms perceived as instant

# Extras – Concatenated words

Input: Concatenatedvwords suck!

Model Output: Concatenated words suck!

**Word-Level Correction**

Extraction on word level: 2 Operations: Replace + Insert

1. Replace: ✅

→ Concatenated suck!

2. Insert: ❌

→ Concatenated suck!