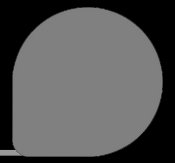
Next Generation Input Methods

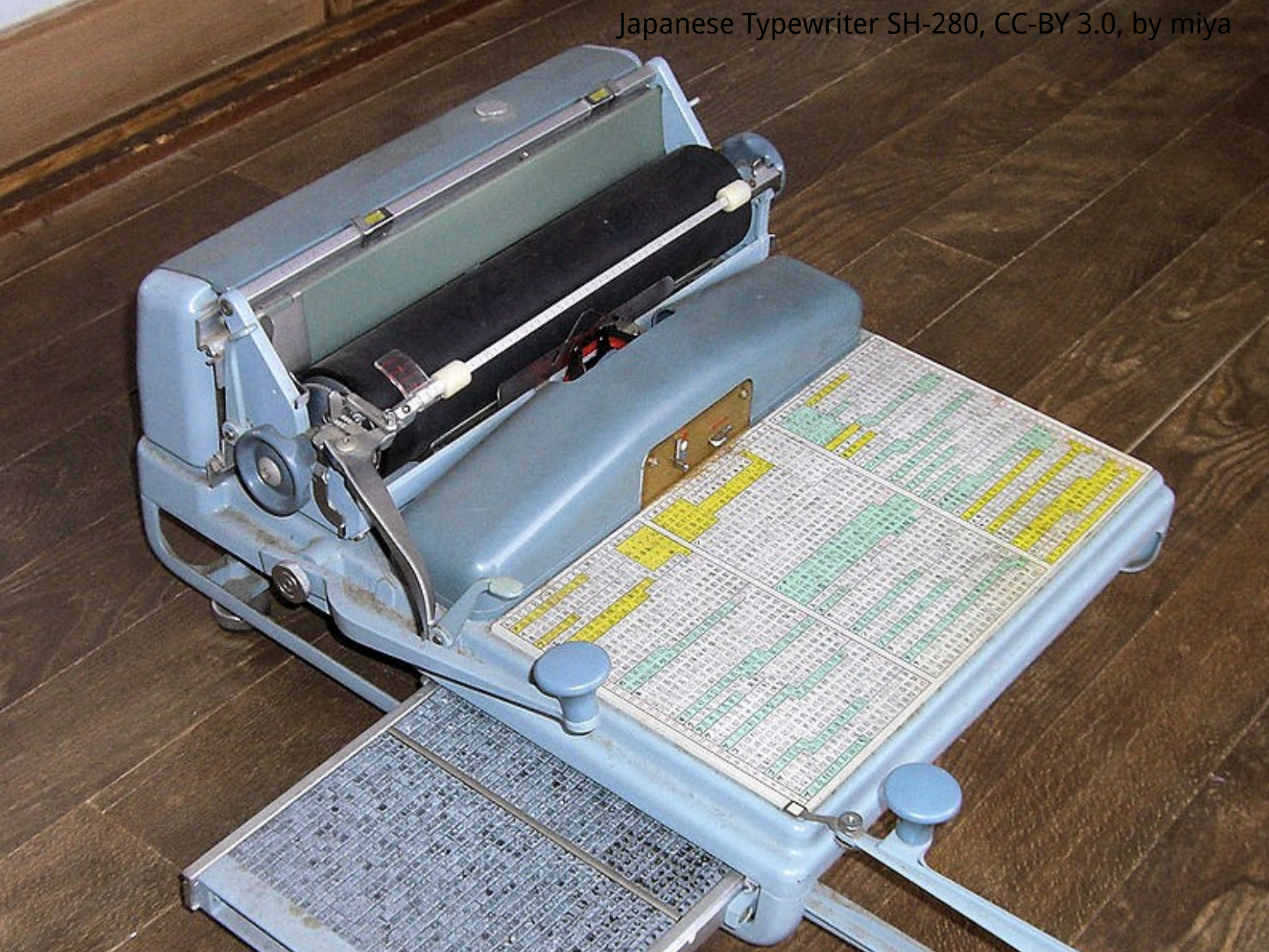
Presented by
Daiki Ueno, Anish Patil

Today's Topics



- Japanese input basics ;-)
- The theory and algorithm behind it
- Next generation features
- Architecture

Japanese input basics


Japanese Typewriter SH-280, CC-BY 3.0, by miya

Japanese input in one slide

- ASCII sequence
  - kyouhaiitenkidesune
- Japanese alphabets (Kana)
  - きょうはいいてんきですね
- Japanese sentences (Kana + Kanji)
  - 今日はいい天気ですね
  - きょうは良い天気ですね
  - …

Character conversion 1:1

Sentence conversion 1:N

Note: There's no single solution!

How does it work?

1. Split input sequence into segments
2. Assign Chinese characters to segments
3. Find the most likely output

1. Split into segments

- き | ょうはいいてんきですね
- きょ | うはいいてんきですね
- …
- きょう | は | いいてんきですね
- きょう | はい | いてんきですね
- …
- きょう | は | いい | てんきですね

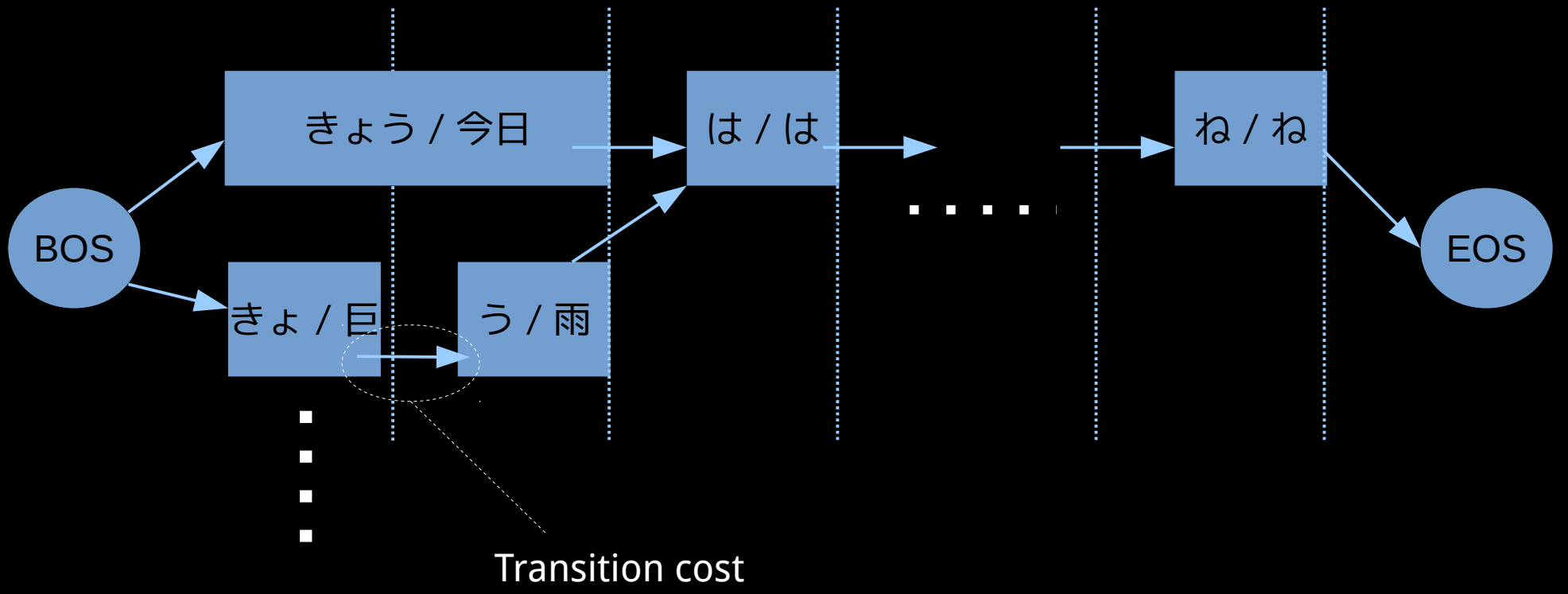
$$N=\frac{n(n-1)}{2}$$

2. Assign Chinese characters

- 木 | ょうはいいてんきですね
- 巨 | うはいいてんきですね
- …
- 今日 | は | いいてんきですね
- 今日 | 杯 | いてんきですね
- …
- 今日 | は | 良い | てんきですね

$$N' = \sum_{k=1}^{N} C_k$$

3. Find the most likely output



Now it turned into the shortest path problem,
though we need a **language model** to compute costs

Language models

- Assign probability of sentence or words
  - 1-gram: only one word
  - 2-gram: 2 consecutive words
  - 3-gram: 3 consecutive words
- Generated from corpus
  - Considering **features** of each word
    - Notation, part of speech, length, ...

Implementation: libkkc

- Language model
  - 3-gram language model generated from:
    - Wikipedia (Japanese): 100,000 sentences
    - Yahoo! Chiebukuro (Q&A site): 20,000 sentences
  - Only using notation of each word
- > 90% accuracy
  - To recover sentences from newspaper articles

Next generation features

Problems

- Our language is changing
- Human beings are lazy

Our language is changing

- Languages reflect the current events
  - あべ (pronunciation: əbe) is a popular Japanese surname, written as:
    - 阿部 , 安倍 , 安部 , or 阿倍
  - When Mr. 安倍 was appointed as the Japanese prime minister
    - あべしゅしょう should be 安部首相 , not 阿部首相
    - あべせいけん should be 安部政権 , not 阿部政権

Our language is changing

- Misuse sometimes becomes formal
  - × 怒り心頭に達する = たっする
  - ○ 怒り心頭に発する = はっする

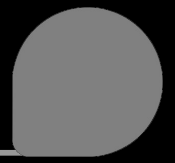
Solutions

- Use on-line language model
  - Privacy issues
- Release language model data frequently
  - Requires bandwidth
- Interpolate language model with updates
  - May sacrifice accuracy

Human beings are lazy

- Cumbersome to type the whole sentence
- Can't remember the meaning of a word
- Can't remember the pronunciation of a character
  - We have thousands of characters!

Solutions



- Predictive input
- Handwriting input

Predictive input

A system that suggest the next possible word from the previously input words

- Pros
  - Users don't need to type the whole sentence
  - More information could be presented to user
    - Thesaurus, derivation of word, etc.
- Cons
  - User distractions
  - Privacy issues

Handwriting input

Find a character by handwriting shape, drawn using a pointing device

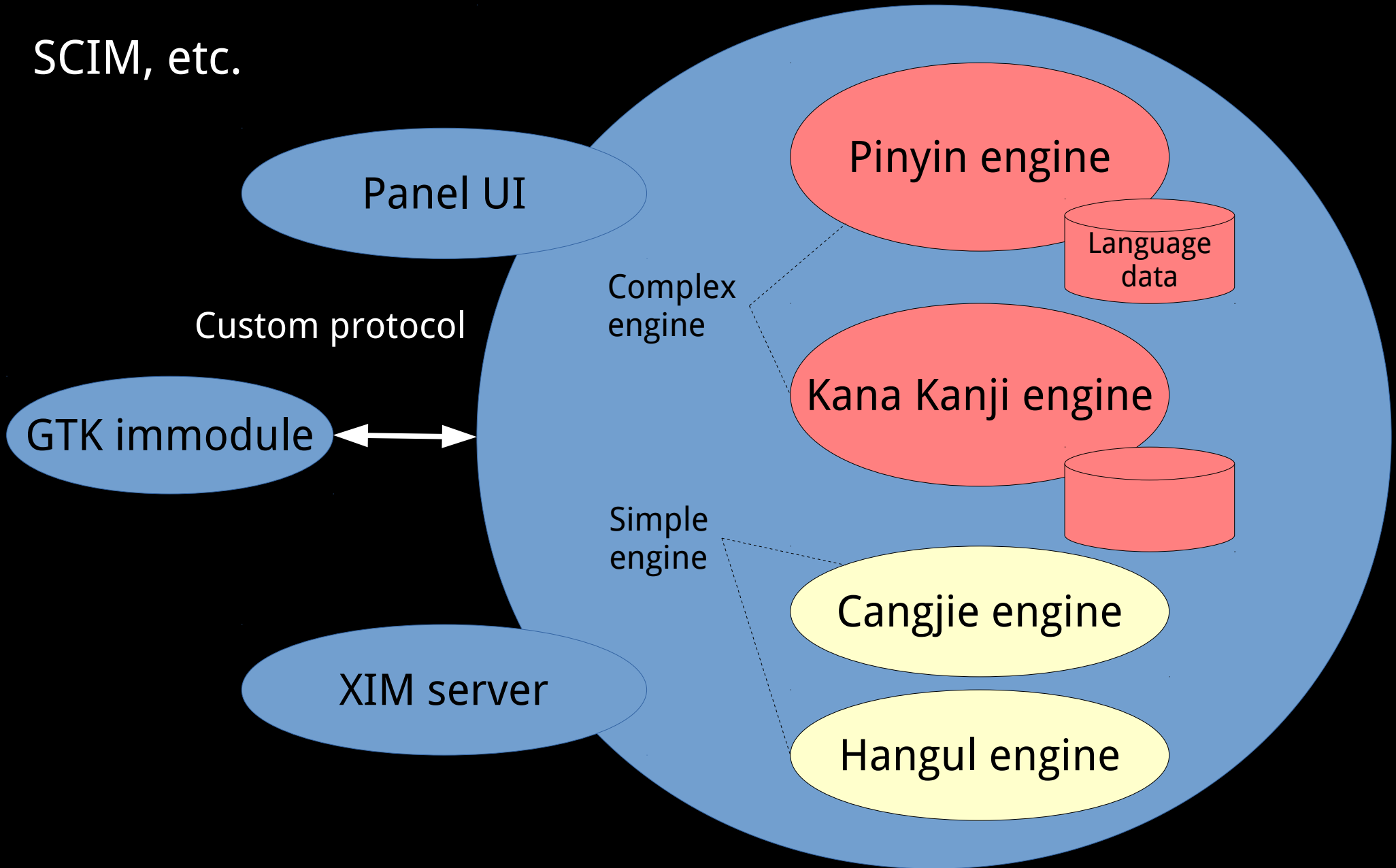
- Pros
  - No need to bring a dictionary
- Cons
  - Accuracy
  - Typing speed

Architecture

Architecture

- No, I'm not proposing an IBus competitor?
- Is the current IBus architecture ideal?

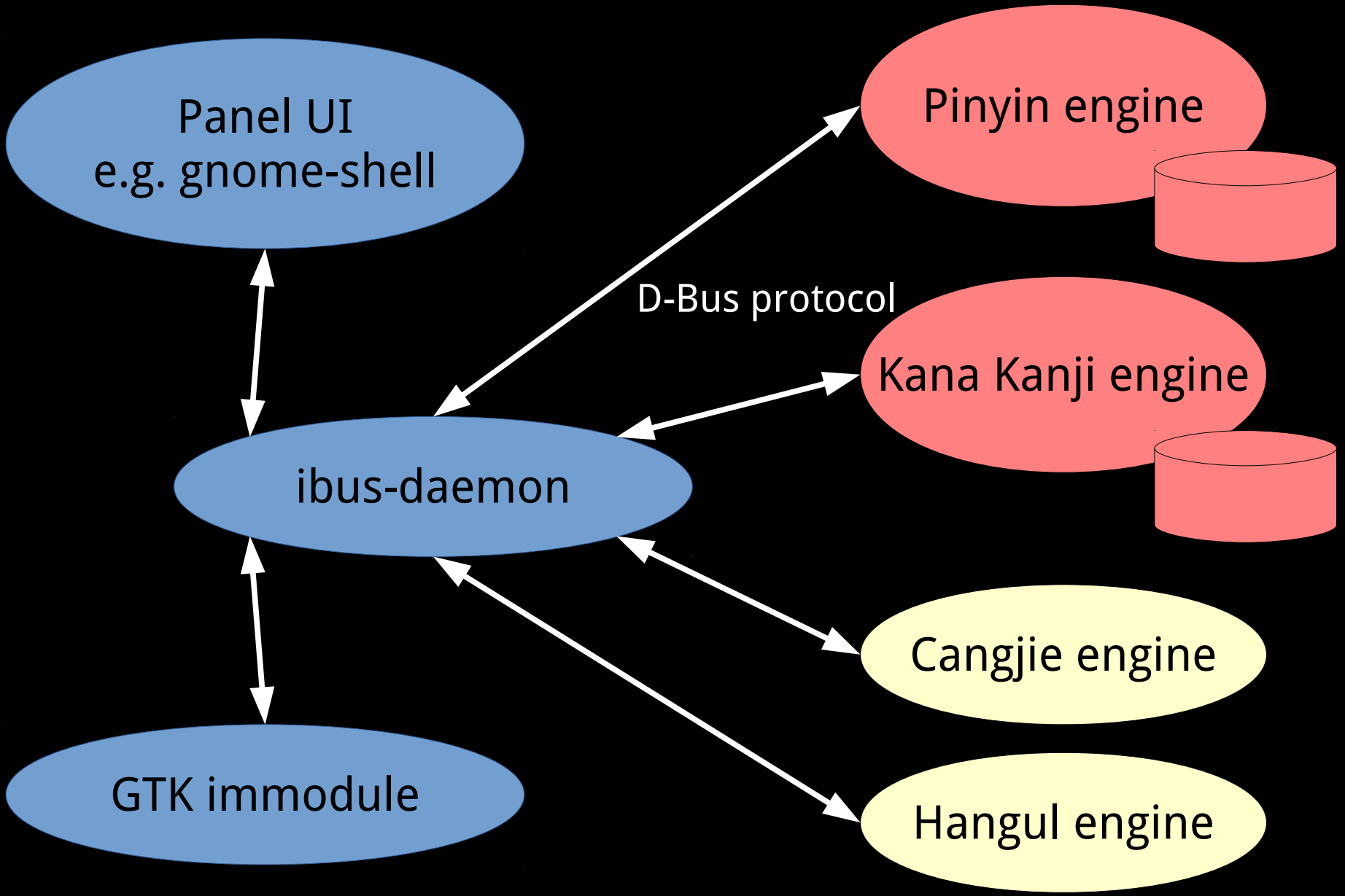
Traditional IM architecture



Traditional IM architecture

- Pros
  - Fast response
- Cons
  - Engines can make the whole system unusable
    - Some engines are very complex and sometimes become irresponsive on high resource usage

IBus architecture



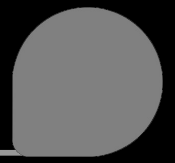
IBus architecture

- Pros
  - Crash resistant
  - Stable frontend (panel) API, based on D-Bus
- Cons
  - Slow response
  - Complicated implementation

Implementation issues

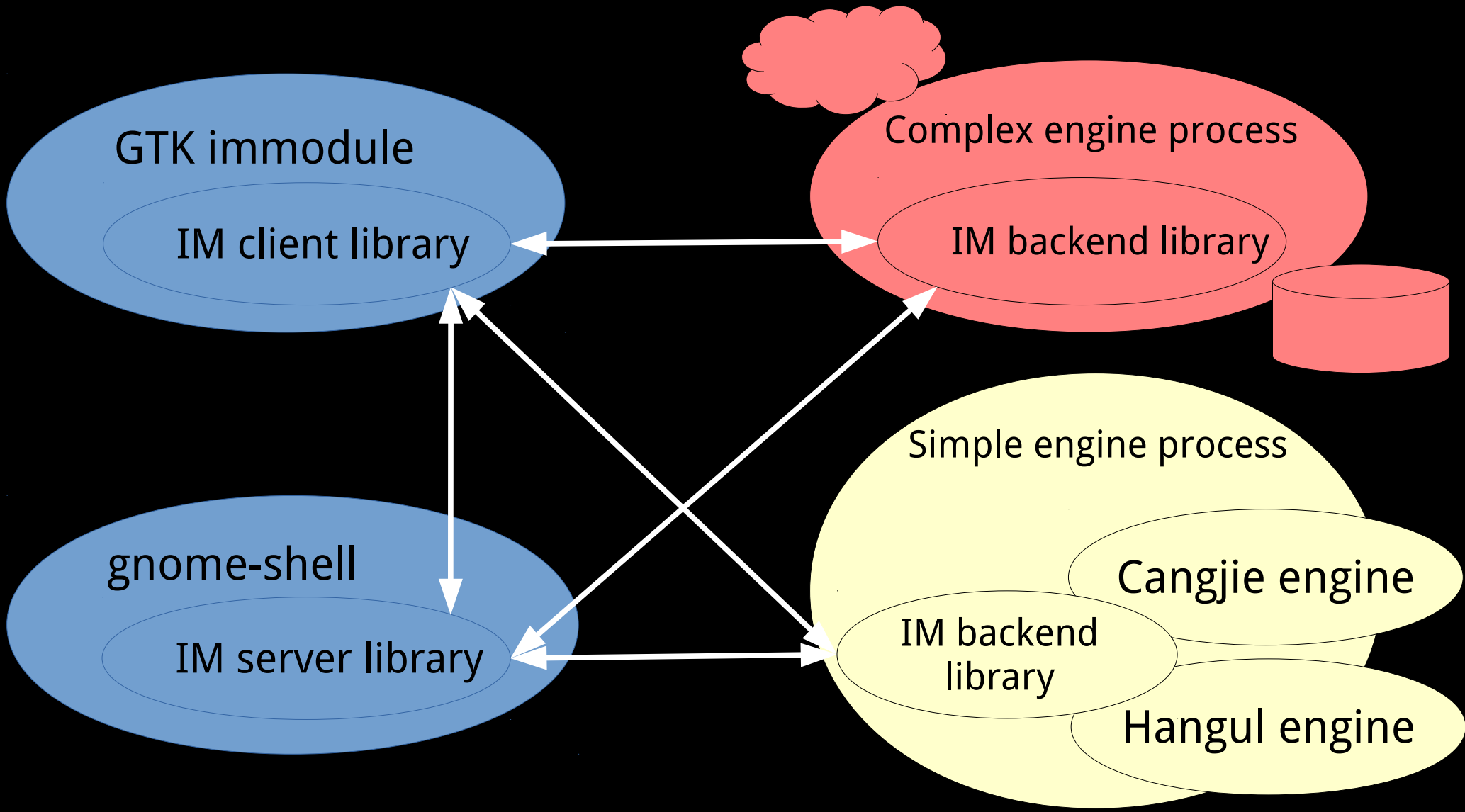
- The backend API is not fully asynchronous nor cancellable
- Process management glitches
  - No mechanism to recover crashed engine
  - Newly installed engines are not recognized until ibus-daemon restarts
- Small number of test cases
  - ~30% code coverage

Objectives



- For the next generation features, the architecture should carefully handle
  - Privacy, on-line updates, UI for predictive input, ...
- Make complex engines more like an ordinary GNOME application
  - Could be registered through .desktop file
  - Take advantage of sandboxing
- While providing lightweight access to simple engines

Proposed architecture



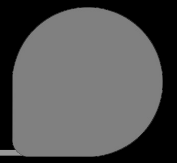
Proposed architecture

- A client and engine communicate through a peer-to-peer connection
- An engine and panel communicate through the session bus
- A single process can accommodate multiple engines (e.g. Cangjie and Hangul)
- Libraries mediate those connections

Libraries

- IM backend library
  - Exports engine service(s)
- IM client library
  - Communicates with the backend library
  - Responsible for input events
    - Key press, focus, ...

Libraries (cont'd)



- IM server library
  - Brokers connection between engine and client
  - Responsible for management events
    - IM menu, candidate list, handwriting, ...
  - Provides IBus compatible front-end API

Questions?