Input Method

UI, Architecture, Algorithm, and Future

Presented by
Daiki Ueno

Who am I?

- Free software contributor
  - Committer of Emacs, GnuPG, GNUTLS, and GNOME
  - @ueno at github, ohloh
- One of the core developers of IBus
  - Intelligent Input Bus, an input method framework
    - http://code.google.com/p/ibus/
  - Wrote surrounding-text, Dconf support, etc.
  - Maintainer of ibus-m17n

Today's Topics

1. Intro
   - What's input method and how people use it
2. UI
   - Common UI elements of IM
3. Architecture
   - What components are inside IM
4. Algorithm
   - Algorithms behind complex IM
5. Future

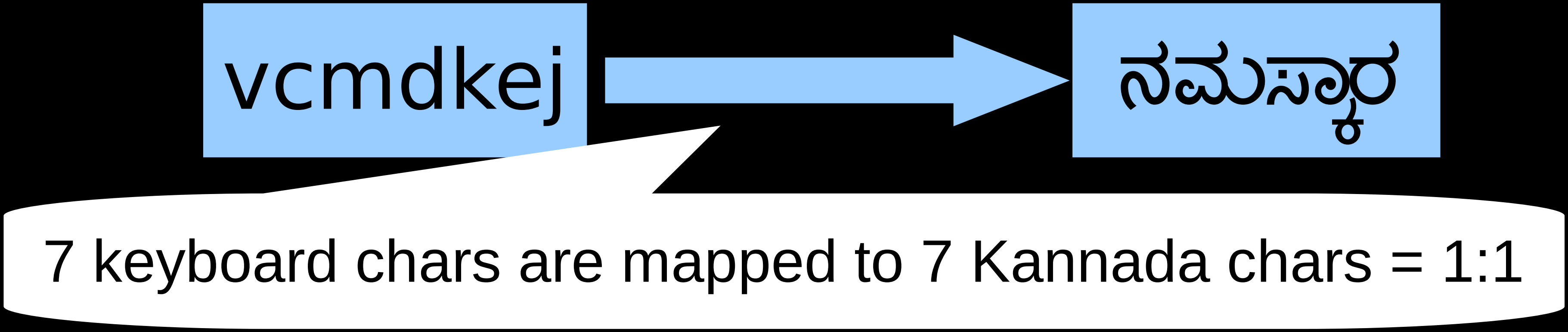
Intro

What is input method?

- Mechanism to input native text to application
  - Through UI toolkits or XIM protocol
- Two IM types
  - Character based IM
    - Cangjie, Hangeul, Indic, Thai, Vietnamese
  - Sentence based IM
    - PinYin, Japanese

Character based IM (CBIM)

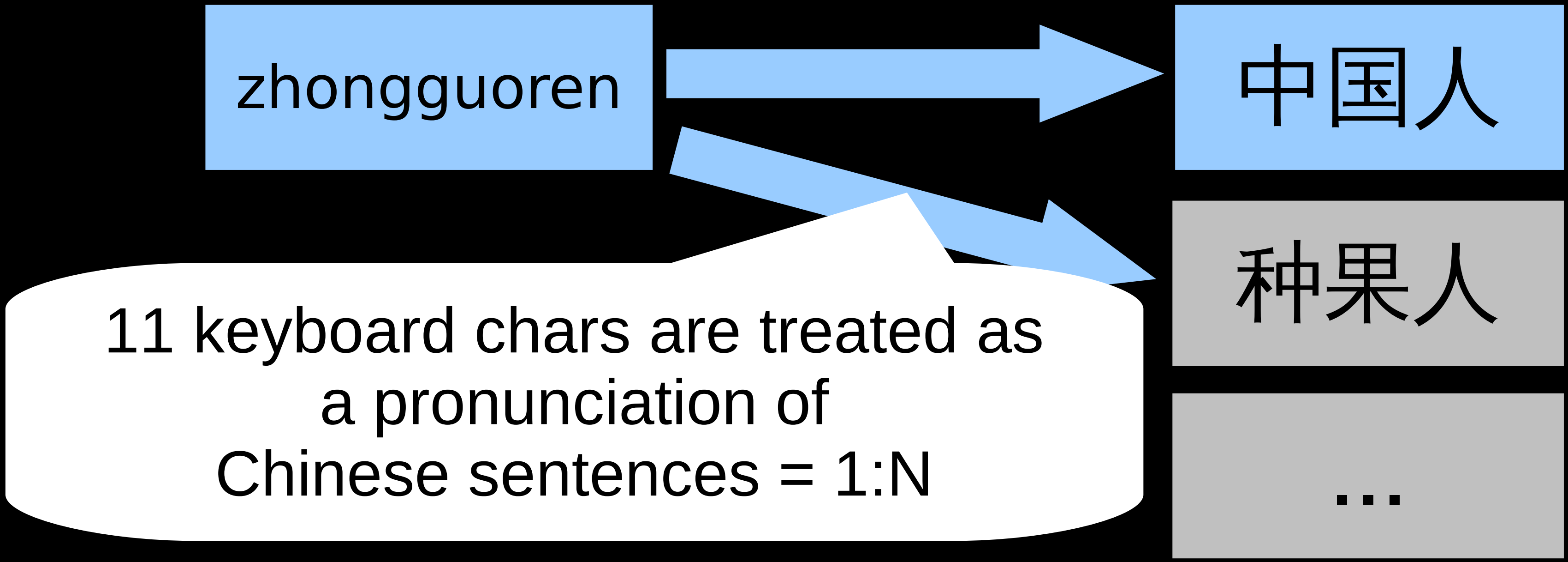
- **Input sequence** is mapped to **characters**
  - Cangjie, Hangeul, Indic, Thai, Vietnamese



7 keyboard chars are mapped to 7 Kannada chars = 1:1

Sentence based IM (SBIM)

- **Input sequence** is mapped to **sentences**
  - PinYin, Japanese
  - Candidate sentences are provided as a list
    - The most likely sentence will be shown in front of the list
    - User chooses a desired sentence through some UI



11 keyboard chars are treated as
a pronunciation of
Chinese sentences = 1:N

Summary of CBIM and SBIM

- CBIM - Easy
  - Can be implemented with simple dictionary lookup
- SBIM - Not so easy
  - More tasks
    - How to split input sequence into words
    - How to find most likely output sequence
    - How to recover from failed conversion
- Some of IM are hybrid of CBIM and SBIM

Hybrid example: Japanese

- Input 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

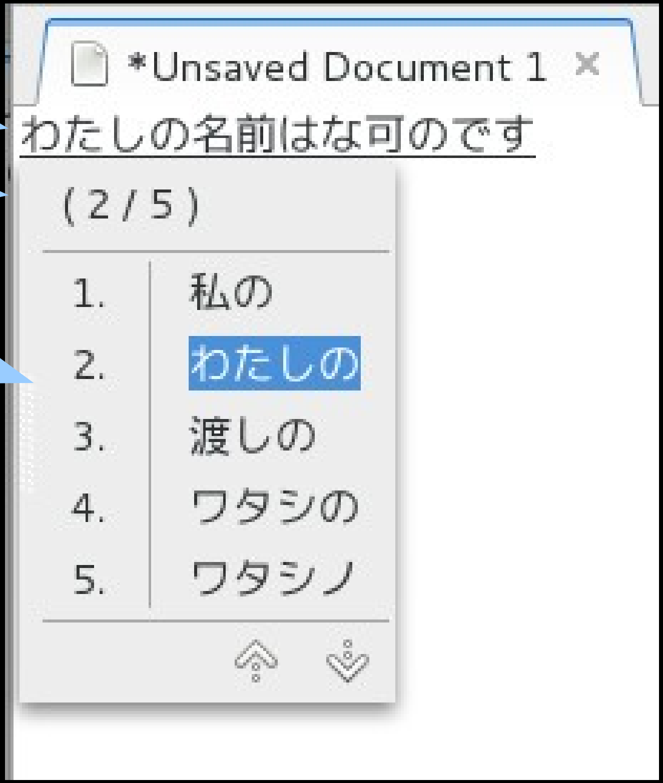
Sentence conversion 1:N

Some Kana chars are translated to Kanji (Chinese) chars

UI

Basic UI elements

- Present intermediate user input, hints, and candidates
  - Pre-edit text
  - Auxiliary text
  - Candidate list



*Unsaved Document 1 ×

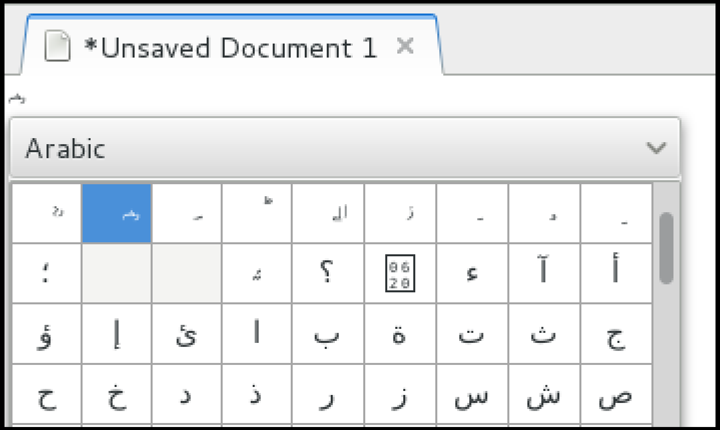
わたしの名前はな可のです

(2/5)

1. 私の
2. わたしの
3. 渡しの
4. ワタシの
5. ワタシノ

Additional UI elements

- Some IM implement more UI elements
  - Drawing pad for handwriting recognition
  - Character map



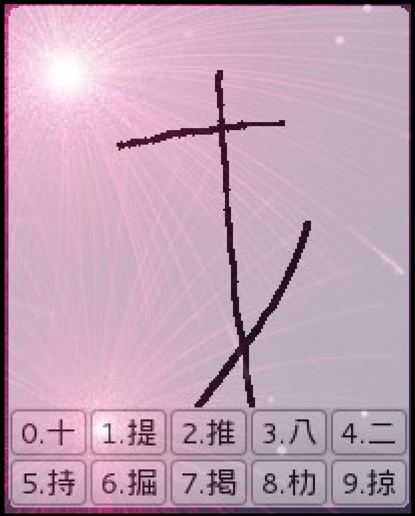
*Unsaved Document 1

Arabic

؛ ، ؟ 0620 ء آ أ

ؤ إ ئ ا ب ة ت ث ج

ح خ د ذ ر ز س ش ص



0.十 1.提 2.推 3.八 4.二

5.持 6.掘 7.揭 8.朸 9.掠

Architecture

IM components

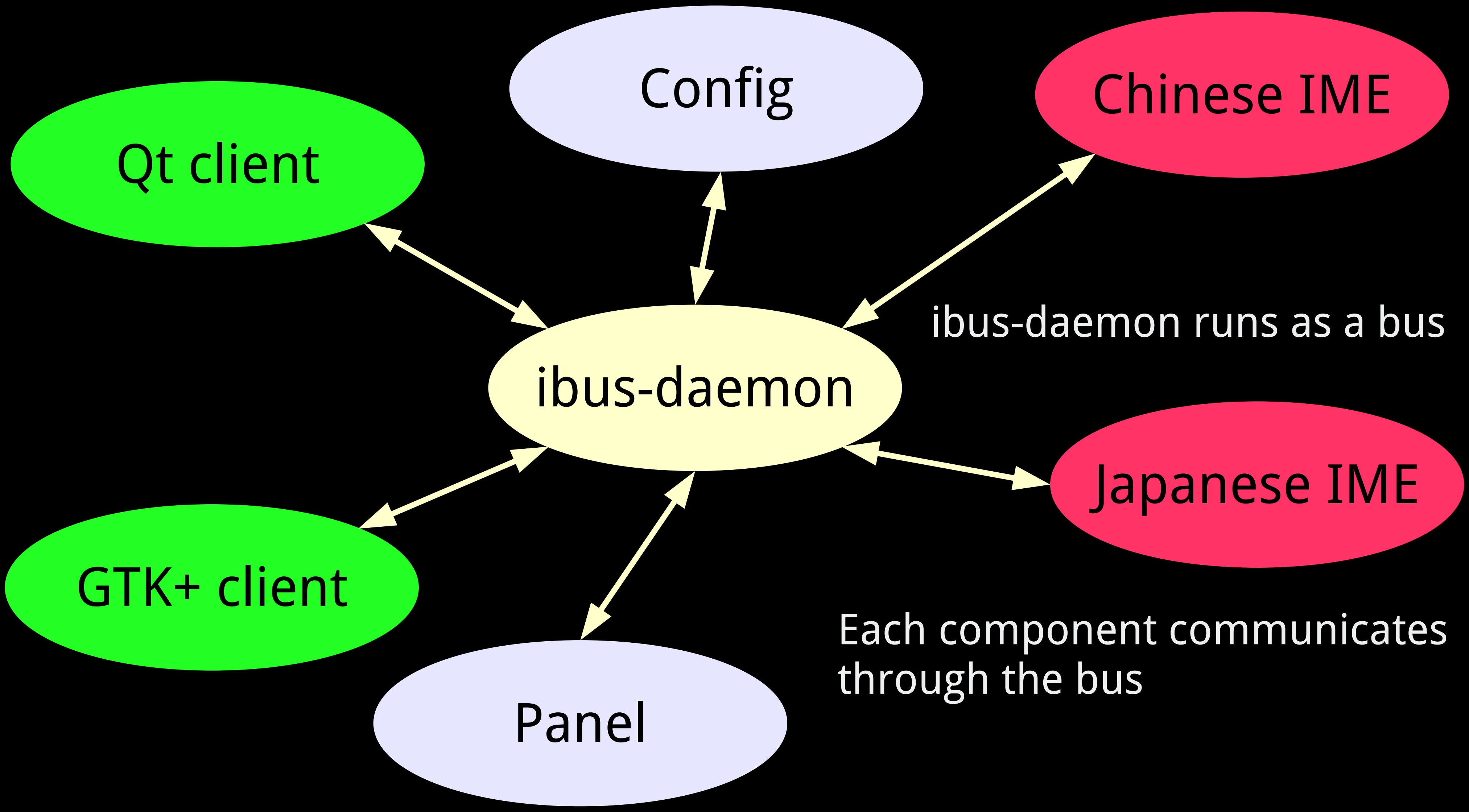
- IM client
  - Gtk/Qt immodule, XIM server
- IM engine (IME)
  - ibus-pinyin, scim-anthy
- IM panel (UI)
- ...

Communicate
with each other

Single process or multi-process

- Single process
  - Fcitx, uim, SCIM
  - Pros: lightweight, easy to add UI component
  - Cons: one component can crash the whole system
- Multi-process
  - IBus
  - Pros: robust against crashes
  - Cons: high latency because of IPC

Multi-process architecture



ibus-daemon runs as a bus

Each component communicates through the bus

Client: Hook into GUI toolkits

- Plugged into GUI toolkits through dynamic modules, called **IM modules**
- Text widgets shall call “hooks” of IM modules

```
static gint
gtk_entry_key_press (GtkWidget   *widget,
                     GdkEventKey *event)
{
  ...
  if (gtk_im_context_filter_keypress (priv->im_context, event))
    {
      ...
      return TRUE;
    }
  ...
```

Actually implemented by IM modules

Algorithm

Recall the example

- Input 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
  - ...

Easy

Character conversion 1:1

Sentence conversion 1:N

Not so easy

Sentence conversion

- Rule based approach
  - Pros: easy to tune to each case
  - Cons: need to maintain thousands of rules
- Statistics based approach
  - Pros: maintenance cost is low
  - Cons: difficult to tune to corner cases

Statistics based (2-gram)

1. Split an input sequence into all possible sub sequences
   - 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

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

Statistics based (2-gram)

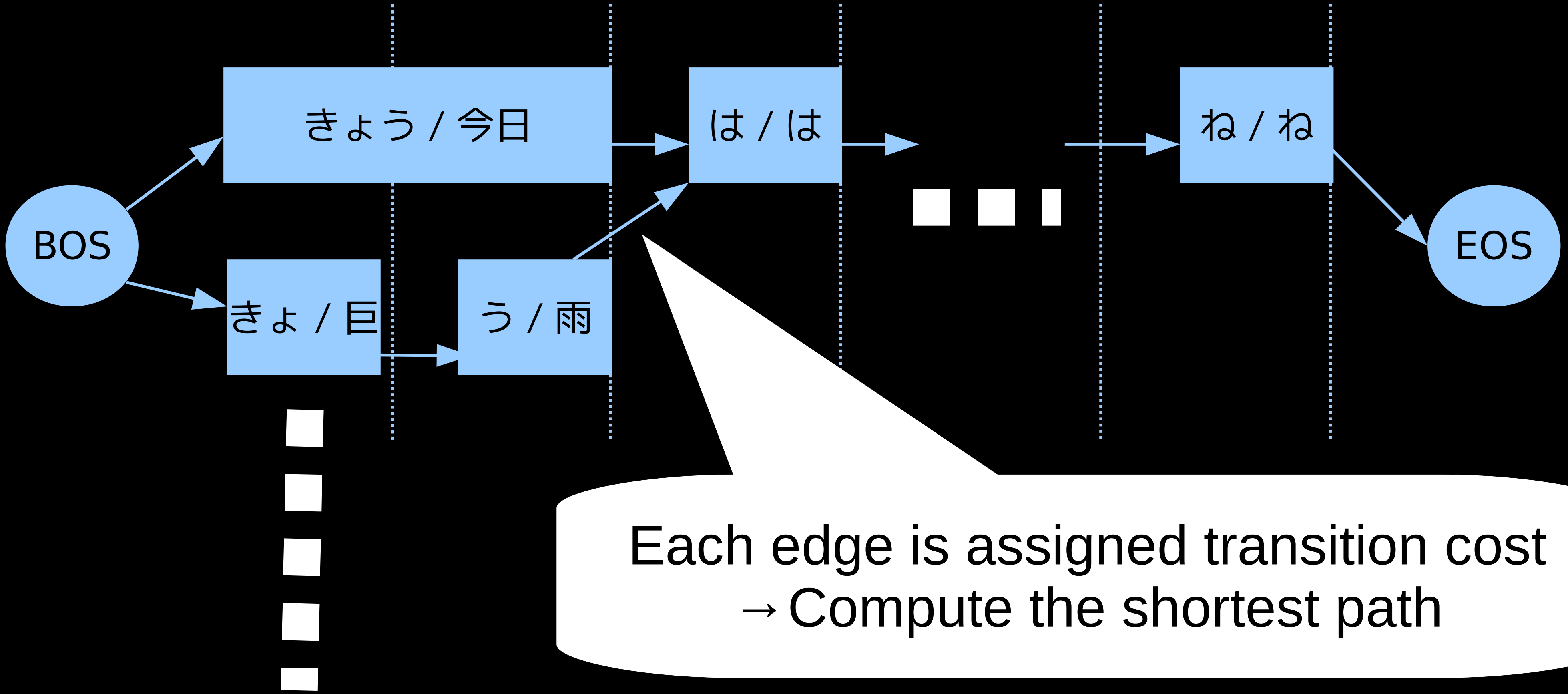
2.Lookup words for each sub sequence

- 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

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

Statistics based (2-gram)

3.Construct a graph of ~N' + 2 nodes



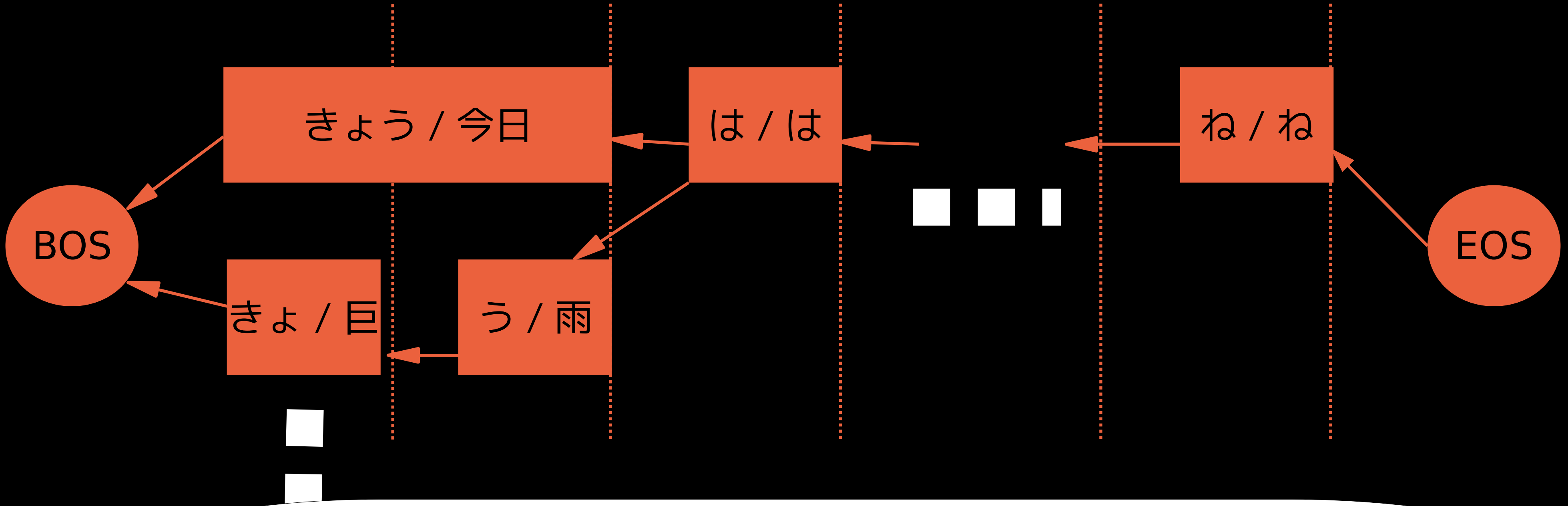
Each edge is assigned transition cost
→Compute the shortest path

Transition cost

- Obtained from **language model**
- Language model construction
  1. Count occurrences of words in corpus
  2. Count occurrences of word pairs in corpus
  3. Compute probability of each occurrence
  4. Smoothing
- Some free software tools are available
  - MITLM, Palmkit

Statistics based (2-gram)

4.Find other candidates



A* (A-star) search from BOS, using cumulative cost saved on each node

Algorithms

- Finding the most likely sentence
  - Compute the shortest path with Viterbi algorithm
  - http://en.wikipedia.org/wiki/Viterbi_algorithm
- Finding the other possible sentences
  - Backward A* search from EOS
  - http://en.wikipedia.org/wiki/A*_search_algorithm
- Changing word boundary
  - Add constraints when building the graph

Algorithm improvements

- Use 3-gram rather than 2-gram
  - Accuracy will be improved
  - Possible, but a bit too complex
- Compress language models
  - Use succinct data structure

Future

Challenges I

- Using context information
  - Language detection
    - Switch IME automatically based on document
  - Predictive input
    - Use preceding N words to predict the next input
    - Similar algorithm as statistic sentence conversion
  - How to retrieve context information?
    - GTK+: surrounding-text, input purposes and hints

Challenges II

- Support for restricted input environment
  - Mobile input
    - Maybe the complex SBIM algorithm is not suitable
    - Maybe predictive input is good enough
    - Approximate matching for mistyped words
  - Accessibility
    - Integrate with scanning mode
      - visual input with a single key

Recent development topics of IBus

- Improve performance of initial startup
- Switch to binary based cache
    - Currently it is in XML
- Detect newly installed IME
    - Currently user needs to restart ibus-daemon
- Use XInput2 raw events to utilize long press
- Support GSettings in proper way
    - Currently it directly uses Dconf library

Hints on development

- Concentrate on “plumbing” rather than UI
  - Desktop UI trends are rapidly changing
- Make the core algorithm reusable
  - Turn it into a library
    - libpinyin, libhangul, m17n-lib, libskk, libchewing
  - Allow access from various programming languages
  - Be flexible about IM framework mushrooms
    - IBus, Fcitx, uim, SCIM, HIME, ...

Summary

- Three aspects of modern IM are presented
  - UI elements of IM
  - Single process vs multi-process
  - Rule based or statistic based algorithm
- There are still rooms for improvement

Questions?

Contact:
ueno@fedoraproject.org