京都発! オープンソースなかな漢字変換の変遷

おーぷん万葉
はしもとまさひこ＆京橋ひよわ
2016/7/30 オープンソースカンファレンス 2016 Kyoto

おーぷん万葉

Q. 突然ですが質問です^^

この中で聞いたことがある用語があったら挙手！

- IBus　（アイバス）
- Fcitx　（ファイティックス）
- mozc　（モズク）
- libkkc　（リブカカシ）
- FreeWnn　（フリーウンヌ）
- Anthy　（アンシー）
- Canna　（カンナ）

実はこれらは・・・

Linuxで使われる主な(?)かな漢字変換システムの名称です

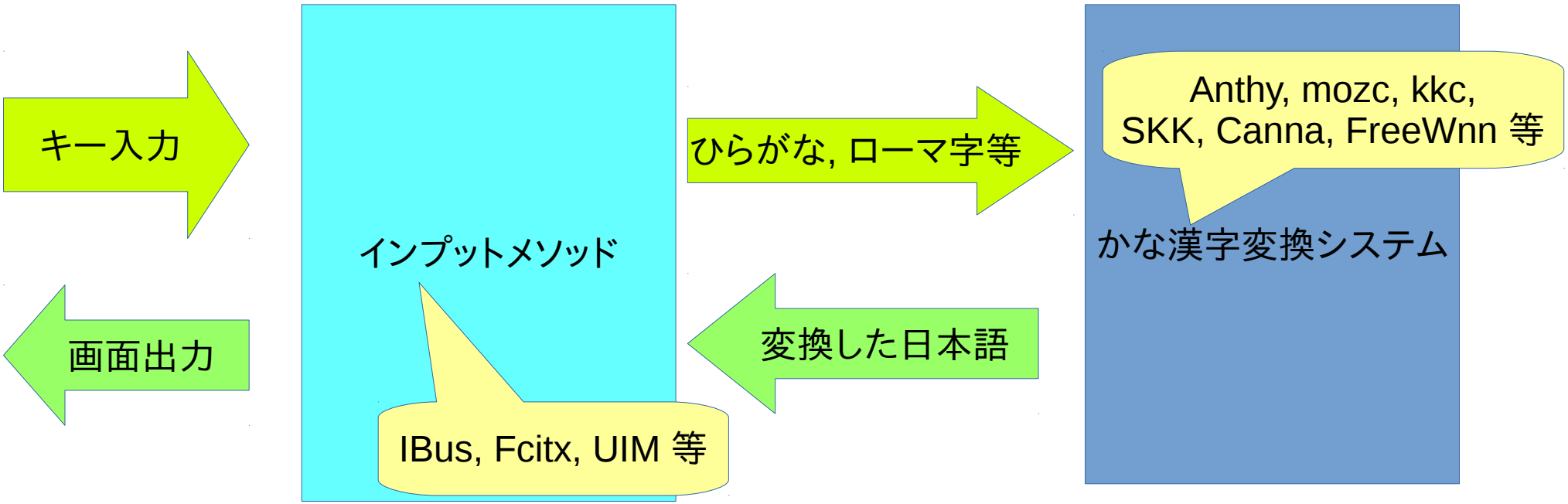
- インプットメソッド
  - IBus
  - Fcitx
- かな漢字変換システム
  - mozc
  - libkkc
  - FreeWnn
  - Anthy
  - Canna

どのくらいご存知でしたか??

FreeWnnとAnthyは京都発祥の
かな漢字変換システムですね

インプットメソッドとかな漢字変換

Linuxではこんな感じで変換しています



サーバー・クライアント型をイメージしやすい？
※実際、ちょっと古めのかな漢字変換はサーバ型でした！
（FreeWnn, Cannaなど）

おーぶん万葉

さて、本日のアジェンダ

- 第1部：　「かな漢字変換の仕組み」をざっくりと。
  担当）　はしもとまさひこ　15分
  ～かんたんな概要編～

- 第2部：　「日本語入力のこれから」
  担当）　京橋ひよわ　30分
  ～一歩踏み込んだ考察編～

おーぷん万葉

京都発! オープンソースなかな漢字変換の変遷

第一部:
「かな漢字変換の仕組み」をざっくりと。

おーぷん万葉　はしもとまさひこ
2016/7/30 オープンソースカンファレンス 2016 Kyoto

おーぷん万葉

簡単に自己紹介

- 東海道らぐ関東案内人
  - 東海道らぐ = **Tokaido Linux User Group**
  - 本日12時からアトリウムにてLT大会があります！
    こちらもぜひご参加ください!!!
- ちびぎーこ保護者会（別名：日本openSUSEユーザ会）の人
- 最近は日本語入力についてもいろいろやってます
  - かな漢字変換「Genji」開発中！

「かな漢字変換の仕組み」をざっくりと。

アジェンダ

- **Anthy**以前：
  - 1987年： FreeWnn
  - 1990年： Canna
  - 2001年： Anthy
- **今時の**かな漢字変換
  - 2010年： Mozc
  - 2013年： libkkc

- かな漢字変換のこれから　～**Genji**の挑戦～

1. FreeWnn ～since 1987～

- OSC京都と行ったらFreeWnnですね!!!
  - FreeWnnに縁のある方々:
    - 吉田智子先生（本年度OSCアワード・OSC京都実行委員長）
    - 山下康成氏（本年度OSCアワード）

- ワークステーションでも連文節変換を!

  …というのを目指して開発されました

Wnnの名前の由来

Watashino Namaeha Nakanodesu

↑これを一発で変換できるようにしたい!!!

おーぶん万葉

Wnnの大文節変換と小文節変換

- 大文節:
  - 複数の小文節から大文節ができている単語

    第三十回、大魔人、通学路、大代表

- 小文節:
  - 辞書にある単語

    私、昔、足し算、代表

どちらの変換を使うか
ユーザーが選択できます

おーぶん万葉

Wnnの評価値計算式

- 小文節の評価値＝

  自立語部分の頻度 × 頻度重み[2] ＋
  小文節の長さ × 小文節長重み[45] ＋
  自立語部分の読みの長さ × 自立語長重み[5] ＋
  自立語部分の今使ったよビット × 最新使用状況重み[80] ＋
  自立語部分の属する辞書の辞書優先度 × 辞書優先度重み[5]

- 大文節の評価値＝

  大文節を構成する小文節の評価値の平均 × 小文節の評価値重み[1] ＋
  大文節の（読みの）長さ × 大文節長重み[40] ＋
  小文節数 × 小文節数重み[0]

ちょっと複雑ですね…^^;;

2. Canna ～since 1990～

- NECが開発していた
  - 今はMITライセンスとなりオープンソース化

- 逐次変換機能（今でゆうところのライブ変換）が実装された

Cannaの文節長決定方法：2文節最長一致法

連続する2文節が最長となるものを選択する

- 例文：　「きょうはあひるやきです」

今日	歯	あ	昼	焼きで	酢

6文節

今日は	あ	昼	焼きです

4文節

今日は	家鴨	焼きです

3文節

今日は	あひる焼きです

2文節
2文節で一番長くなるのでこれを選択

おーぶん万葉

ポイント：
文節 = 自立語（名詞･動詞等） + 付属語（助詞等）

3. Anthy ～since 2001～

- 京都を代表するもうひとつのかな漢字変換
  - 京大マイコンクラブにて開発が始まる
  - 平成13年度未踏ソフトウェア創造事業として採択

- 機械学習による識別モデルに基づいた変換
  - OSSでは統計的手法の先駆けとも言える存在

すみません・・・

本日はAnthyの詳細な説明を省略いたします^^;

4. **Mozc** ～since 2010～

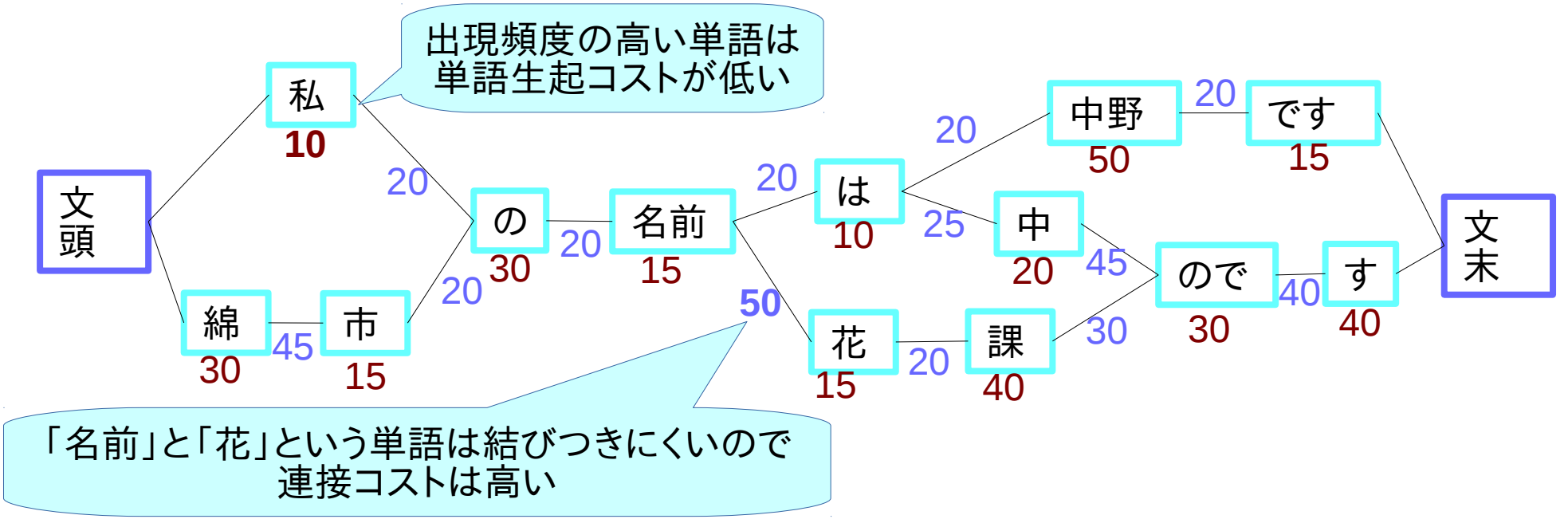
- Google日本語入力のオープンソース版
- 現在のLinux日本語入力のデファクトスタンダード
  - 最近どのディストリでも標準ですよね…

- 形態素解析を用いたかな漢字変換
  - コスト最小法を採用

コスト最小法 とは

注：ちょっと（かなり？）端折って説明します^^;

例：「わたしのなまえはなかのです」を変換する場合（コストの値はテキトーです^^;）



- 単語生起コスト：　単語の出現優先度を表したコスト
- 連接コスト：　単語と単語の結びつきやすさを表したコスト

→ 全て足して、合計値が最も低いルートが候補になります

コスト最小法も完璧ではない？

Mozcで変換できないものもある…

「にわにはにわにわとりがいる」

→「庭には庭鶏がいる」になってしまう

庭	には	庭	鶏	が	いる

庭	には	二	羽	鶏	が	いる

正解の変換のほうが単語数が多いため
単語生起コストがどうしても高くなる
→N文節最長一致の延長線上になってしまってる!?

5. libkkc ～since 2013～

- Fedoraの標準かな漢字変換

- N-gramによるかな漢字変換

  ＝ビッグデータ（巨大コーパス）を十分に活かせる

  その反面、辞書が大きくなりすぎてしまう危険性…

libkkcの変換用データ

-1.114728 ぬいぐるみ/ぬいぐるみ 」/」 ていど/程度
-0.667107 ぬいぐるみ/ぬいぐるみ 」/」 と/と
-0.643911 ぬいぐるみ/ぬいぐるみ うらない/占い を/を
-0.740726 ぬいぐるみ/ぬいぐるみ たすう/多数 を/を
-0.454970 ぬいぐるみ/ぬいぐるみ だ/だ が/が
-0.814252 ぬいぐるみ/ぬいぐるみ で/で わりお/ワリオ
-1.110465 ぬいぐるみ/ぬいぐるみ は/は かのじょ/彼女
-0.802579 ぬいぐるみ/ぬいぐるみ やら/やら しゃしん/写真
-1.626115 ぬいぐるみ/ぬいぐるみ を/を 「/「
-1.708439 ぬいぐるみ/ぬいぐるみ を/を せいさく/製作
-1.631926 ぬいぐるみ/ぬいぐるみ を/を つく/作
-1.713616 ぬいぐるみ/ぬいぐるみ を/を なげつけ/投げつけ
-1.681401 ぬいぐるみ/ぬいぐるみ を/を のこ/残
-1.713256 ぬいぐるみ/ぬいぐるみ を/を もちこ/持ち込
-1.714574 ぬいぐるみ/ぬいぐるみ を/を よご/汚
-1.716680 ぬいぐるみ/ぬいぐるみ を/を りんぐ/リング
-0.813648 ぬいぐるみ/ぬいぐるみ （/（ じょん/ジョン
-0.803737 ぬいぐるみ/ヌイグルミ の/の こと/こと

単語単位ではなく
単語の組み合わせ単位で
変換用データとして
登録されていますね

libkkcの場合は
3単語ずつの組み合わせ
＝ **3-gram** ＝ **tri-gram**

なんだかノイズデータっぽいのもいるのですが…

そして、**Genji**の挑戦・・・

本日のおーぷん万葉ブースでは…

かな漢字変換「Genji」のデモを展示しています

- 現在は「文節単位のN-gram」という手法を用いています
  - 基本はN-gramですが、各々のデータは文節をまたぎません
  - ゆくゆくは文節間の連携も視野に入れた統計モデルにします
    その手法はこのあとのひよわさんの話の中で…

おーぷん万葉

ご清聴、ありがとうございました。

次は、京橋ひよわさんの第二部です！

日本語入力のこれから

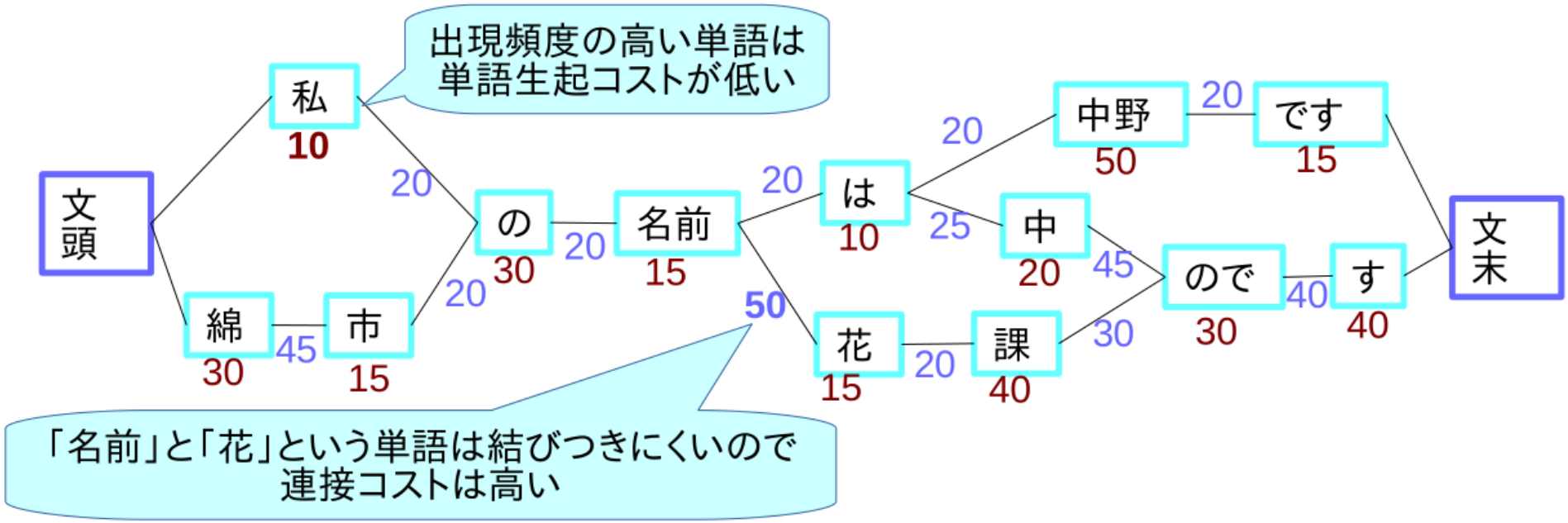
OSC京都 2016 おーぷん万葉プロジェクト
京橋 ひよわ (@Khiyowa)

0. もくじ

- 変換辞書のこれまでとこれから
- 変換処理のこれまでとこれから
- オープンソースのかな漢字変換の課題

1-1. 辞書のこれまで(コスト最小法)

- 統計的かな漢字変換の辞書(コスト最小法)を作る
  - 先ほどのこれ「ビタビアルゴリズム(動的計画法)」



1-1. コスト最小法

- コストをつける
  - 現在、コストは多くが自動でつけられている
    - ✓CRF(条件付き確率場)という手法を使い、機械学習器で学習させて
      コストをつけている
  - ソフトによっては、学習用データを用意する部分と
    学習済みの辞書を洗練する部分は人手
    - ✓某社では学習用コーパスは人手で作っている
    - ✓98%程度の精度でも、残り2%によく使う言い回しが含まれていたりするので、その部分については追加学習が必要
    - ✓WebからデータをとっているGoogle 日本語入力も、
      当初と比べると変換効率は向上しており、おそらく辞書の洗練を行うようになっている…?

1-1. コスト最小法

- 学習用データの作成
  - 生の文に対して情報を付与していく
    ✓私の名前は中野です

```
BOS(文頭)
私　　名詞,代名詞,一般,*,*,*,私,ワタシ,ワタシ
の　　助詞,連体化,*,*,*,*,の,ノ,ノ
名前　名詞,一般,*,*,*,*,名前,ナマエ,ナマエ
は　　助詞,係助詞,*,*,*,*,は,ハ,ワ
中野　名詞,固有名詞,地域,一般,*,*,中野,ナカノ,ナカノ
です　助動詞,*,*,*,特殊・デス,基本形,です,デス,デス
EOS(文末)
```

  - 文に対して、何らかの情報(この場合は品詞情報)を付与したものを「コーパス」と呼んでいる

1-1. コスト最小法

- 品詞を自動で求める
  - 私　の　名前　は　中野　です　。
    代名　格助　名　係助　固名　助動

- 単純に分類するわけにはいかない
  - 「の」の品詞は
    ✓格助詞、終助詞、間投助詞、並立*助*詞、*準体助*詞のどれ?
  - 前後の情報も利用して判断する必要がある
    ✓すべての可能性を確率として保持し、最大値を求める
    ✓最近では、CRF(条件付確率場)モデルが使われる

1-1. コスト最小法

- 確率を使って次の品詞を当てに行く
  - この「確率」が高いものはコストを小さく、
    逆に確率が低いものにはコストを大きくする
  - 辞書を使うときは、このコスト情報を動的計画法(ビタビ)で計算

- 現在の統計的かな漢字変換では品詞が重要
  - 内部的には600個ほどの品詞を持つ製品も

- 形態素解析と表裏一体
  - Google 日本語入力では、形態素解析器(MeCab)を使って
    コーパスを作っている
  - ジャストシステムでは、ATOKの技術を生かして
    商用の形態素解析器(JMAT)を作っている

1-2. 変換辞書のこれから(係り受け編)

- ところで…

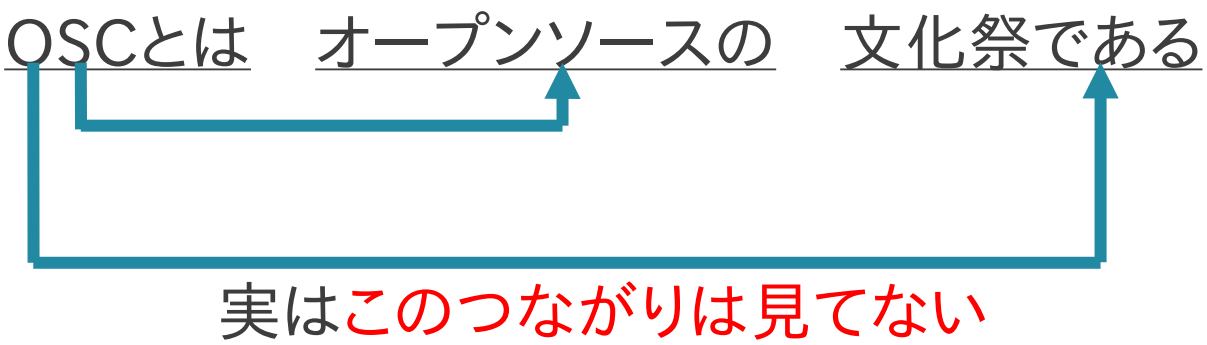
```
BOS(文頭)
私　　名詞,代名詞,一般,*,*,*,私,ワタシ,ワタシ
の　　助詞,連体化,*,*,*,*,の,ノ,ノ
名前　名詞,一般,*,*,*,*,名前,ナマエ,ナマエ
は　　助詞,係助詞,*,*,*,*,は,ハ,ワ
中野　名詞,固有名詞,地域,一般,*,*,中野,ナカノ,ナカノ
です　助動詞,*,*,*,特殊・デス,基本形,です,デス,デス
EOS(文末)
```

  - 文頭に名詞が来ている文が多い
  - 代名詞の次には助詞の連帯化が来る確率が高い…

- 実は直前直後の関係しか考慮してない

1-2. 変換辞書のこれから(係り受け編)

- 実は直前直後の関係しか考慮してない



- 言語って係り受けの関係があるしそれも見ようよ
  - って言ってるのが@hashimomさん

1-2. 変換辞書のこれから(n-gram編)

- 逆に
  - 品詞情報って処理には使ってるけど結果には出てこない
  - 実は要らないんじゃないの?
    - ✓日本語入力の研究をしてる森先生(京都大学)

- n-gramを利用
- 「私の名前は中野です」の文字2-gram
  - 「私の」「の名」「名前」「前は」「は中」「中野」「野で」「です」
- 「私 の 名前 は 中野 です」の単語2-gram
  - 「私の」「の名前」「名前は」「は中野」「中野です」

1-2. 変換辞書のこれから(n-gram編)

- 「文をn文字/n単語分切り出した組」の
  確率を使って辞書を作る
  - 膨大な文書データが必要
  - でも現代なら手に入る…?

- 未知語の処理をどうするか
  - ユーザの入力を使う
  - ユーザは読みを入力し、区切り位置を決定して確定する
  - この読みと区切り位置を使い、学習情報を辞書に追加
  - 学習結果を収集し、ユーザに反映
- 「キーロガー」まがいにもなりかねないので、
  収集については検討が必要

2-1. 変換処理のこれまで

- 漢直(漢字直接入力)の時代
  - 和文タイプライター(戦前, 1915ぐらい)
  - ペンタッチ式ワードプロセッサ(～80年代)
    - ✓2000文字ほどの盤面から漢字を選択せねばならず、熟練の技術を必要とした。「タイピスト」という職業が存在した。
  - 連想方式(～80年代)
    - ✓仮名を2文字入力することで漢字を1文字入力できる方式
    - ✓「リキ」→「力」という音読みのものや、「ミラ」→「鏡」という英語読みに近いものが混在
    - ✓メーカーによってもバラバラ
    - ✓これも素人が使えるものではなかった。

2-1. 変換処理のこれまで

- かな漢字変換という発想
  - 単漢字変換(‘78年)
    - ✓読みと区切り位置を入力して漢字に変換する方式
    - ✓「SKK」は現在も一定数の利用者がいる。
    - ✓OASYSに付属していた「OAK」の後継「Japanist」はサポートが続いており、Windows 10用ドライバもある(2003で開発は終了)
      - –「親指シフト」用に開発されたもの
  - 連文節変換(80年代後半～)
    - ✓読みを入力すると区切り位置も自動判定して変換
    - ✓「VJE」「WX」「ATOK」「松茸」などのソフトウェアが変換精度やUIを競った
    - ✓スペースキーで変換してEnterで確定、という操作は
      ジャストシステムが考案したとされる
    - ✓WX2/WX3は後のMicrosoft IMEとなった

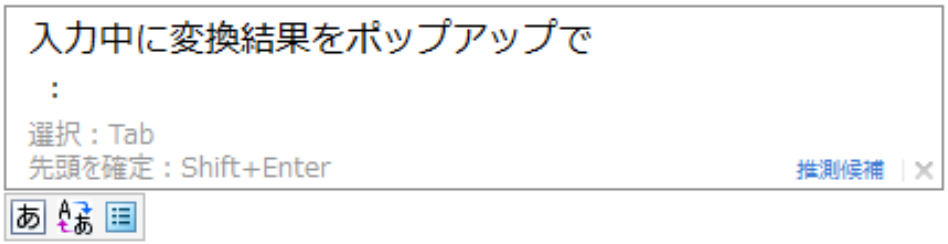
2-1. 変換処理のこれまで

- ところで
  - 連文節変換は未変換/変換中の文がメモリに載る
    - ✓あまり入力されるとメモリが不足する
  - 変換の演算をするのに時間がかかってしまう
- 入力文字数を制限する
  - 未確定では一定以上の入力ができなくなる
    - ✓KAREN(富士ソフト)など
- 自動変換
  - ある程度入力されたら自動で変換していく
    - ✓句読点や、一定文字数ごとなどの条件で自動で変換を行う
      - –いつ頃からかはわからないが、90年11月発売の書院にはあった
    - ✓最近のパソコンではメモリ不足や処理落ちの心配はないものの、設定で自動変換にできる

2-1. 変換処理のこれまで

- 近年
  - サジェスト(Google日本語入力) / 推測変換(ATOK)
    - ✓入力中に変換結果をポップアップで表示

✓にゅうりょくちゅうにへんかんけっかをぽっぷあっぷで



    - ✓入力中にリアルタイムで計算を行っている
  - ライブ変換(ことえり)
    - ✓入力した端から自動で変換していく
    - ✓同じく、入力中にリアルタイムで計算を行っていると考えられる
  - サジェストは早すぎると邪魔/遅すぎると役立たず
    - ✓データをどう持つか

2-1. 変換処理のこれまで

- 以下を参照
  https://www.google.co.jp/ime/comic/small_32.html

2-2. 変換処理のこれから

- ライブ変換の拡張
  - 入力モード(英数/かな)などの切り替えも不要にする
    ✓というのがGenjiの実装目標

- コンピュータの処理速度の向上と
  辞書のデータ構造の工夫によって可能となった
  - キーボードの入力はコンピュータから見ると非常に遅い
    ✓その間に変換処理を走らせることはさほど難しいわけではない
  - 探しやすいデータ構造も大切

3. オープンソースかな漢字変換の課題

- 辞書を作る
  - データ構造はいいものがある
  - 手法もある
  - オープンデータのコーパスがない
    - ✓著作権が絡むため、好き勝手にデータを公開できない
    - ✓自由に手に入り、自由に提供できる良質なコーパスを整備する必要
    - ✓これもおーぷん万葉の活動の目標
- 普及(かな漢字変換に限らずOSS全般にもいえる)
  - 使う人がいないと開発者も集まらない
    - ✓作っても需要がない?
    - ✓Genjiも現状、@hashimomさんが一人でやってる
  - *nixのかな漢字変換が新しい方式をひっさげてくる割に消えていくのはもしかしてこれが原因?

まとめ

- 変換辞書
  - 係り受けを考慮したかな漢字変換、
    ビッグデータで辞書を作る方法が検討されている
- 変換処理
  - ライブ変換を拡張し、入力モードも不要になる
    ような、かな漢字変換が検討されている
- 課題
  - コーパスの整備とOSSの普及

参考

- 日本語入力を支える技術
  - 2012 徳永 拓之 編 技術評論社
- コミック: Google 日本語入力ができるまで
  - https://www.google.co.jp/ime/comic/
- ジャストシステムの形態素解析技術
  - http://www.slideshare.net/JSUXDesign/jtt1
- ジャストシステムの形態素解析技術 その2 機械学習編
  - http://www.slideshare.net/JSUXDesign/2-51367666