言語の自動判別を利用した多言語入力

江原 遙　　田中久美子
東京大学工学部計数工学科　東京大学情報理工学系研究科
niam@bk.main.jp　　kumiko@kumish.net

1　はじめに

近年，計算機環境でもローカリゼーションが進み，多言語を扱うための環境が普及してきている．

多言語を入力するためのインターフェースには，一般にキーボードのキー列を対象言語の文字列に変換する Input Method Engine (IME) と呼ばれるソフトウェアを用いる．IME は各言語ごとに作られているため，ユーザーが入力言語を切り替えたい場合は，IME を切り替える必要がある．IME の切り替えは，通常，専用の特殊なキー操作やマウス操作などにより行われる．

ところが，言語の**混ぜ書き**を行う場合には，IME の切り替え作業が頻繁に必要になり，入力作業が煩雑になる．これが，入力ミスや集中を阻害する原因となる．

そこで，本研究ではユーザーが入力しているキー列の属する言語を実時間で判別し，入力 IME を自動選択することで IME の切り替え作業なしに入力することが可能なインターフェースを提案する．

2　既存研究

多言語入力インターフェースの既存研究の中で，本稿のように言語の切り替え作業そのものを省略しようとする試みは筆者らの知る範囲ではまだ少ない．英語と中国語の 2 言語の間においてのみ，Chen et al. [4] が隠れマルコフモデルを用いて混ぜ書きが可能なインターフェースとして Modeless Pinyin Input を提案している．しかし，この研究はあくまで中国語入力をスムーズに行うための一環として行われており，より多くの言語間での混ぜ書きについては触れられていない．

入力インターフェースを離れ，一般的な言語判別問題を扱った研究としては，ニュースグループの投稿を対象に言語判別を行った Cavnar et al. [2] の研究が基礎的である．N-gram による比較的単純な方法を用いて，各言語 20KByte〜120KByte ほどの学習データで，300Byte 以下の文書を 14 言語に分類する場合において，95%程度の高い精度を得られると報告している．

入力インターフェースではないが言語の混ぜ書きに関する研究としては，Alex [1] はドイツ語中に混在する英語を判別する手法を提案し，文献 [2] のアルゴリズムより高い性能を得たとしている．しかしこの研究は豊富なレキシコンや World Wide Web との単純な照合によるアプローチと機械学習によるアプローチの精度を比較することに主眼があるので，本稿のように任意の多言語を扱う場合とは目的も手法も異なる．

3　インターフェース

3.1　ユーザーインターフェース

本節ではインターフェースの使用方法について説明する．

まず，起動すると図 1a のようになる．ここで，例えば “satousan” と打つと未確定の文字列がテキストエリア中に反転表示される．このとき，一打鍵ごとに言語が判別され，図の Current locale:で始まるウィンドウにその結果が表示される．図 1b から “satou” のわずか 5 文字で日本語と判別されていることが確認できる．このように反転表示された文字列は一つの言語に属するという仮定の上で自動的に言語判別され，最適な言語が選択される．

“satousan” と打ち終わった状態（図 1c）でもやはり日本語と判別されていることがわかる．**変換キー**を押すと図 1d のように日本語の文字列に変換される．他の変換候補を選びたい場合にはもう一度変換キーを押すと図 2 のようにルックアップウィンドウが表示されるので，ここから入力したい文字列を選択して確定すればよい．

次にこのまま “speaks” と入力すると図 1e のように英語と判定される．英語の場合は変換候補は一つしかない

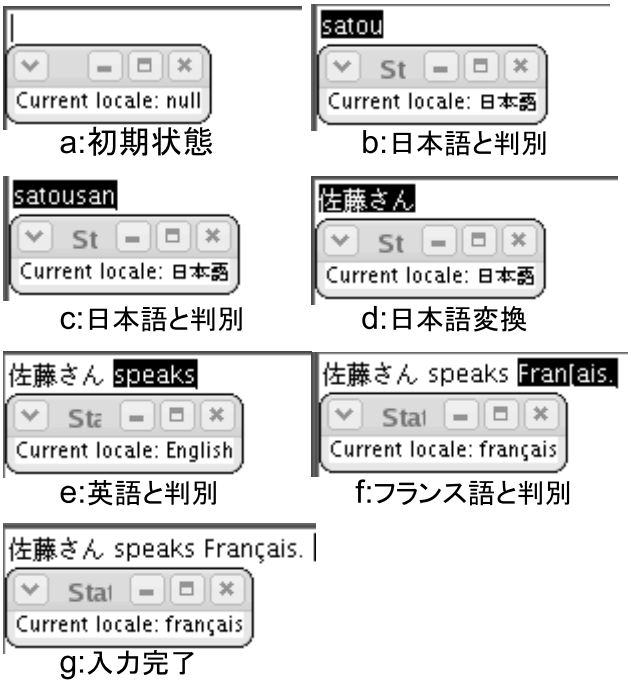

図 1: 入力過程図

ので変換キーを押すと自動的にスペースが入力された上で確定される．

同様に “Fran[ais.”[1]と入力すると今度はフランス語と判定され（図 1f），全体で “佐藤さん speaks Français.” という文字列が図 1g のように入力される．他の言語についても，変換の必要な言語 (例：中国語) は日本語と同様に，変換の不要な西欧諸語一般は英語と同様に入力することが可能である．

このインターフェースは全体としてかな漢字変換などの IME を用いたインターフェースと入力手順がほぼ同一であるため，既存の IME を使用したことのある者であれば 10 分程度の練習で本研究のインターフェースも習得することが可能である．

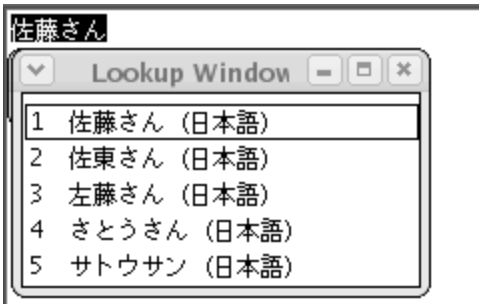

図 2: ルックアップウィンドウ

[1]フランス語のカナダ式入力では日本語キーボードの “[” を打鍵すると “ç” が表示されるのでこのように打鍵する．

3.2 システムの構成

図 3 にシステムの構成を示す．本システムは言語の

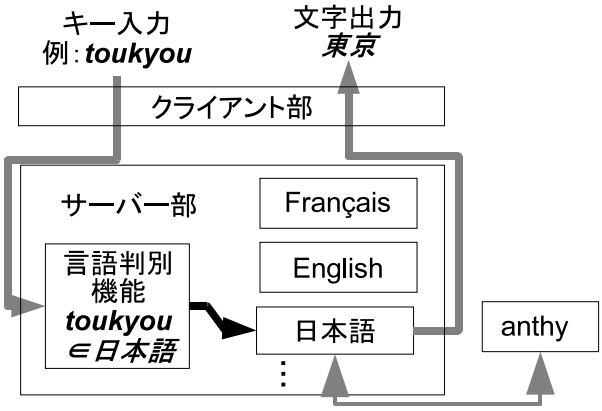

図 3: システム概観

判別とキー列から文字列への変換を行う**サーバー部**と，ユーザーにインターフェースを提供する**クライアント部**からなる．さらにクライアント部は Java 上で動作する IME として実装されており，アプリケーションのソースコードを書き換えることなしに使用することができる．

クライアント部とサーバー部の間は独自の通信プロトコルでソケット通信を行っている．サーバー部は内部に言語変換機能を持っており，英語や仏語などキー列から文字列への変換が単純な置き換えですむ場合は変換を自ら行う．しかし日本語は特に複雑であるため，サーバー部はさらに外部のかな漢字変換サーバーに変換作業を委託する[2]．このようなサーバー部の実装の詳細はクライアント部には完全に遮蔽されており，クライアント部はサーバー部にすべての処理を委託すればよいようになっている．

4 言語判別の定式化

本インターフェースでは，ユーザーは変換と確定をこの順序で繰り返しながら入力を行う．このとき，一つ前の確定から次の確定に至るまでの間に入力された，言語判別の対象となるキー列をチャンクと呼ぶことにする．本インターフェースでは，チャンクが属する言語は 1 言語と仮定されている．過去のチャンクを並べていき，チャンク列を次のように定義する．

入力されるキーの種類の集合を K，打鍵数を n，ユーザが入力したキー列を $k_1^n = k_1, k_2, \ldots, k_n (k_i \in K)$ とおく．チャンク列を c_1^m とおくと，そのキー列との関係は，

[2]外部のかな漢字変換サーバーには anthy を用いた．
http://anthy.sourceforge.jp/

$c_i = k_{s(i)}^{s(i+1)-1}$ $(s(i) < s(i+1), s(1) = 1, s(m+1) - 1 = n)$ となる．言語の集合を L とし，チャンク c_i が属する言語を ℓ_i とする．目的は，c_1^m と ℓ_1^{m-1} が与えられたときに，ℓ_m の推定値 $\hat{\ell}_m$ を求めることである．ベイズの定理を用いて展開すると $\hat{\ell}_m$ は次のように表すことができる．

$$\begin{aligned}\hat{\ell}_m &= \operatorname*{argmax}_{\ell_m \in L} \mathrm{P}(\ell_m | \ell_1^{m-1}, c_1^m) \\ &= \operatorname*{argmax}_{\ell_m \in L} \mathrm{P}(\ell_1^{m-1}, \ell_m | c_1^m) \\ &= \operatorname*{argmax}_{\ell_m \in L} \mathrm{P}(\ell_1^m) \mathrm{P}(c_1^m | \ell_1^m) \end{aligned} \quad (1)$$

ここで，式 (1) によれば，ℓ_m を推定するためには事前確率 $\mathrm{P}(\ell_1^m)$ と尤度 $\mathrm{P}(c_1^m | \ell_1^m)$ をそれぞれ推定しなければならない．事前確率の推定は言語に関する 2-gram による近似を用いて $\mathrm{P}(\ell_1^m) \approx \prod_{i=1}^m \mathrm{P}(\ell_i | \ell_{i-1})$ とし，尤度は $\mathrm{P}(c_1^m | \ell_1^m) \approx \prod_{i=1}^m \mathrm{P}(c_i | \ell_i)$ のように近似する．以上をまとめると式 (2) を得る．

$$\hat{\ell}_m \approx \operatorname*{argmax}_{\ell_m \in L} \prod_{i=1}^{m} \mathrm{P}(\ell_i | \ell_{i-1}) \, \mathrm{P}(c_i | \ell_i) \quad (2)$$

ここで，式 (2) は各言語 ℓ_i での tagging の問題とみなすことが可能である．

4.1 事前確率の近似値の算出方法

事前確率 $\mathrm{P}(\ell_i | \ell_{i-1})$ の近似値を求めるためには，実際に言語が混ぜ書きされたコーパスが必要になるが，そのようなコーパスはまだ少なく，特に多言語について用意するのは難しい．そこで，ヒューリスティクスとして，文書のほとんどは単一の言語でかかれるがわずかに他の言語が混在する場合を想定し，事前確率を設定した．

4.2 尤度の近似値の算出方法

尤度 $\mathrm{P}(c_i | \ell_i)$ の算出方法を述べる．直接 $\mathrm{P}(c_i | \ell_i)$ を近似してもよいが，ここでは次のような手法を考えた．$\mathrm{P}(c_i) = \sum_{\ell \in L} \mathrm{P}(c_i | \ell) \mathrm{P}(\ell)$ が成り立つ．右辺で ℓ_i に注目すると $P(c_i | \ell_i)$ は，

$$\mathrm{P}(c_i | \ell_i) = \frac{\mathrm{P}(c_i) - \sum_{\substack{\ell \in L \\ \ell \neq \ell_i}} \mathrm{P}(c_i | \ell) \mathrm{P}(\ell)}{\mathrm{P}(\ell_i)} \quad (3)$$

である[3]．$\mathrm{P}(c_i)$ は全言語 L 中に出現する c_i から推定する．この手法では ℓ_i 以外の言語におけるキー列の発生確率の情報が加わるので，ℓ_i 以外の言語で発生しにくいチャンクは相対的により重く重みづけられ全体として精度が向上すると考えられる．

さて，$\mathrm{P}(c_i | \ell_i)$ の導出過程で使われる ℓ_i 以外の言語 ℓ に対する $\mathrm{P}(c_i | \ell)$ を求めるためには，チャンク c_i が言語 ℓ 中で出現する確率を推定する必要がある．しかし，この確率をコーパス中の c_i の頻度から直接最尤推定することはコーパスのサイズが小さく難しいことが多い．そこで，c_i がキー列であることを利用し以下のようにキー列の N-gram 確率で尤度を近似する．

$$\begin{aligned}\mathrm{P}(c_i | \ell) &= \mathrm{P}(k_{s(i)}^{s(i+1)-1} | \ell) \\ &\approx \prod_{j=s(i)}^{s(i+1)-1} \mathrm{P}(k_j | k_{j-N+1}^{j-1}, \ell) \end{aligned} \quad (4)$$

次に $\mathrm{P}(k_j | k_{j-N+1}^{j-1}, \ell)$ の計算手法について説明する．N-gram 確率の最尤推定値 $\hat{P}_{ML}(k_j | k_{j-N+1}^{j-1}, \ell)$ は，言語 ℓ の学習コーパス中の k_{j-N+1}^{j} の頻度を k_{j-N+1}^{j-1} の頻度で割って求める．しかし，最尤推定値 $\hat{P}_{ML}$ をそのまま推定値として使用するとゼロ頻度問題がおきる．そこで，$\mathrm{P}(k_j | k_{j-N+1}^{j-1}, \ell)$ の推定法として線形補間法によるスムージングを行って求める．

線形補間法では，未知や高次の N-gram 確率を，低次の $\hat{P}_{ML}$ の線形和を用いて推定する [3]．すなわち，k_{j-N+1}^{j} に対して $0 < \lambda_{k_{j-N+1}^{j}} < 1$ を定め，

$$\begin{aligned}\hat{P}_{LI}(k_j | k_{j-N+1}^{j-1}, \ell) &= \lambda_{k_{j-N+1}^{j}} \hat{P}_{ML}(k_j | k_{j-N+1}^{j-1}, \ell) \\ &\quad + (1 - \lambda_{k_{j-N+1}^{j}}) \hat{P}_{LI}(k_j | k_{j-N+2}^{j-1}, \ell) \end{aligned} \quad (5)$$

とする．$\lambda_{k_{j-N+1}^{j}}$ の推定方法には様々な方法が提案されており，**kneser-ney-mod** [3] などを試したが精度などを比較検討し高次の N-gram を指数的に重く重み付けする簡単な方法を採用した．この $\hat{P}_{LI}(k_j | k_{j-N+1}^{j-1}, \ell)$ を式 (4) の $\mathrm{P}(k_j | k_{j-N+1}^{j-1}, \ell)$ として $\mathrm{P}(c_i | \ell)$ を推定する．

5 評価

5.1 コーパス

本稿における学習には，各言語の文字に変換されたコーパスではなく，変換前のキー列のコーパスを使用する必要がある．しかし，そのようなキー列のコーパスは十分な量がないため，自然文のコーパスをキー列に変換する処理を施して学習コーパスとして利用した．

日本語のコーパスは毎日新聞の 2004 年前半期の記事を利用し，MeCab [4]を用いてローマ字化した．中国語

[3]通常は，このような計算は行わず直接式 4 を用いて $\mathrm{P}(c_i | \ell_i)$ を推定する．しかし，5 節に述べる実験では脚注 7 に述べるように精度の向上が見られたため，この方法を採用した．

[4]http://mecab.sourceforge.jp/

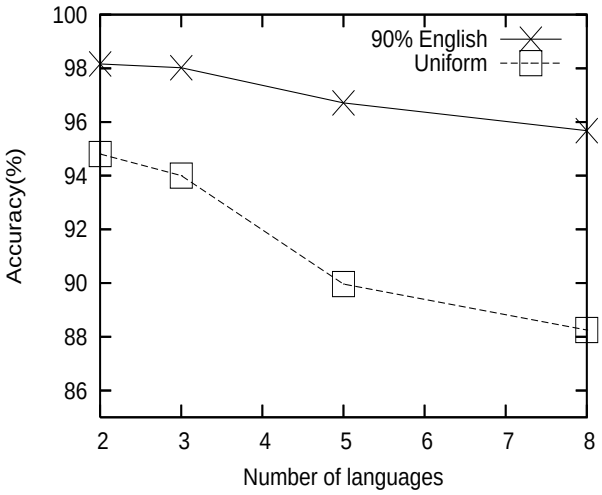

図 4: 言語数に対する精度

のコーパスは，北京大学コーパスを pinyin に機械的に変換したものを用いた．英語，フランス語，ドイツ語，フィンランド語，エストニア語，トルコ語のコーパスには Leipzig Corpora [5]を用いた．学習コーパスのサイズは全言語において約 100KByte とした．

また，キーボード上のキー配置はフランス語がカナダ式である以外はそれぞれの国の本土で一般的に使用されているものを用いた．

5.2 結果

多言語が混ぜ書きされた文章は多いものの正解率を計測するためにタグ付けされたものは非常に少ない．そこで，複数の言語のコーパスから文章部分をランダムに取得してくることにより，人工的に文単位で混ぜ書きされたコーパスを作成し評価実験に利用した．具体的には言語 ℓ のコーパス C_ℓ から文単位でキー列 k_1^n を取得し確率的に区切って[6]チャンク列 c_1^m とし言語を判別させた．

また，たとえ多言語文書であっても現実的には，１つのメインとなる言語がありそのほかの言語が部分的に混じっている文書が非常に多いと考えられる．そこで，人工的に作成するコーパスも，英語が 9 割の確率で発生する場合 (ケース 1) と，比較のために全言語が同じ確率で発生する場合 (ケース 2) について実験を行った．

図 4 に，言語の数に対する提案する言語モデルを用いて言語判別をした際の精度を示す．精度は全チャンク数に対して正しく言語が判別されたチャンク数の割合である．実線はケース 1 の場合，破線はケース 2 の場合である．n 言語の実験は，一つは英語とし，残りの $n-1$ 言語はランダムに選んだ．8 言語の場合以外は，そのような入力 5 セットずつ 2000 文 (約 200KByte) に対して実験を行った[7]．

ケース 1 で 2,3 言語の場合が一般的に最も使用されると思われる．この場合の精度は約 98%であり，十分によい精度が得られている．ケース 2 で 8 言語の場合は，最悪の場合であり精度の下限とみなすことができるが，この場合でも 88.3%と比較的高い精度が得られている．

ケース 1 でもケース 2 でも高い精度が得られているが，この理由の一つは，この評価実験が文単位の混ぜ書きであるためである．文中で数単語だけ他言語が混在する現象も多く存在するので，このような場合における性能評価が今後の課題となる．

参考文献

[1] B. Alex. An unsupervised system for identifying English inclusions in German text. In *Proceedings of the ACL Student Research Workshop*, pp. 133–138, Ann Arbor, Michigan, June 2005.

[2] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 161–175, Las Vegas, April 1994.

[3] S. F. Chen and Joshua T. Goodman. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University, 1998.

[4] Z. Chen and Kai-Fu Lee. A new statistical approach to chinese input. In *The 38th Annual Meeting of the Association for Computer Linguistics*, pp. 241–247, Hong Kong, October 2000.

[5] http://corpora.informatik.uni-leipzig.de/download.html

[6] 英語のように分かち書きする言語は分割部分で区切った．また日本語や中国語のように分かち書きをしない言語では，文ごとに子音をランダムに一つ選びその子音の前後で区切った．

[7] ここで，式 (3) の計算によって判別精度が向上するか，ケース 1 で 4 言語のときに実験した．その結果，直接式 4 を用いて $\mathrm{P}(c_i|\ell_i)$ を推定した場合と比較して，言語判別の精度が 0.8%程度向上した．ただし，式 (3) 中の $\mathrm{P}(\ell), \mathrm{P}(\ell_i)$ は一様とした．