TypeAny: 言語判別を用いた多言語入力システム

江原　遥[†]・田中久美子[†]

近年，国際化に伴い，多くの言語を頻繁に切り替えて入力する機会が増えている．既存のテキスト入力システムにおいては，言語が切り替わるたびに，ユーザーが手動で，テキスト入力ソフトウェア (IME) を切り替えなければならない点が，ユーザーにとって負担になっていた．この問題を解決するために，本論文では，多言語を入力する際にユーザーの負担を軽減するシステム，TypeAny を提案する．TypeAny は，ユーザーが行うキー入力からユーザーが入力しようとしている言語を判別して，IME の切り替えを自動で行う．これによって，ユーザーが IME を切り替える操作量が減るため，複数の言語をスムーズに切り替えながら入力することが可能になる．本研究では，隠れマルコフモデルを用いて言語の判別をモデル化し，モデルにおける確率を PPM 法を用いて推定することで TypeAny を実装し，その有用性を評価した．その結果，人工的なコーパスにおける 3 言語間の判別において，96.7% の判別精度を得た．また，実際に多言語を含む文書を用いて実験したところ，切り替えに必要な操作の数が，既存の手法に比べて 93% 減少した．

キーワード：テキスト入力システム，多言語，言語判別，PPM, 隠れマルコフモデル

TypeAny: Multilingual Text Entry System based on Language Identification

Yo Ehara[†] and Kumiko Tanaka-Ishii[†]

Computer users increasingly need to produce text written in multiple languages. However, typical computer systems require the user to change the text entry software each time a different language is used. This is cumbersome, especially when the languages change frequently. To solve this problem, we propose TypeAny, a novel multilingual text entry system that identifies the language of the user's key entry and automatically dispatches the input to the appropriate text entry system. This language identification is modeled as a hidden Markov model whose probability is estimated by using the PPM method. When evaluating this method, we obtained language identification accuracy of 96.7% when an appropriate language had to be chosen from among three languages. The number of control actions needed to switch languages was decreased 93% when using TypeAny rather than a conventional method.

Key Words: *Text entry system, Multilingual, Language identification, PPM, Hidden Markov model*

[†] 東京大学大学院情報理工学系研究科, Graduate School of Information Science and Technology, The University of Tokyo

1 はじめに

近年，国際化の流れの中で，多くの言語を頻繁に切り替えて入力することが多くなってきている．例えば，自然言語処理の分野では，“named entity” や “chunking” といった英語の表現が，そのままの形で日本語文中に出現することも多い．このように同一文書内に複数の言語が混在する文書を，本論文では「多言語文書」と呼ぶ．

言語入力には，ユーザーが入力したキー列を，その言語での文字列に変換するために，*input method engine* (IME) と呼ばれるソフトウェアが欠かせない．例えば，日本語のローマ字入力の IME は，ユーザが “tagengo” というキー列を入力した時，これを “多言語” という文字列に変換する役割を担う．IME は，日本語や中国語など，漢字への変換に限定されたものとして捉えられることも多いが，本論文では，以後，単純に，キー列から文字列への変換を担うソフトウェアという意味で用いる．

従来は，入力する言語を切り替えるたびに，この IME をユーザが手動で切り替えていた．しかし，これでは，言語を切り替える際のユーザの負担が大きく，特に言語を切り替え忘れた時に打ち直しの問題が生じていた．

そこで，本論文では，IME の切り替えを自動化してユーザーの負担を軽減する，TypeAny という多言語入力システムを提案する (Ehara and Tanaka-Ishii 2008)．このシステムは，ユーザーのキー入力と IME を仲介し，ユーザーが入力しているキー列から言語を自動的に判別して，IME を切り替える．これによって，IME の切り替えによるユーザーの負担が，大幅に減少すると見込まれる．

2 関連研究

多言語文書を入力する際に必要なキー入力は，以下のように分類できる．この章では，関連研究をこの分類に沿って述べる．

(1) 本文の文字列に対応する入力

(2) IME に対する操作

(3) IME を切り替えるための操作

(1) は，本文の文字列に対応する入力である．例えば，本文が “多言語” であれば，日本語のローマ字入力では，“tagengo” がこれに当たる．1 つの言語に対応する IME は，入力方式やキーボード配列の違いによって，複数存在する場合もある[1]．例えば，日本語には前述のローマ字入力の他に「かな入力」もある．日本語のかな入力の IME では，“多言語” に対応する (1) のキー

[1] 文献によっては，1 つの言語に 1 つの IME を対応付け，入力方式やキーボード配列の違いを別の名称で呼ぶ場合もあるが，本論文では，入力方式やキーボード配列が違えば，IME も異なるものとする．

列は “q:@yb@” であり，ローマ字入力の場合とは異なっている．日本語に限らず，英語でも，Qwerty や Dvorak といったキーボード配列の違いによって，複数の IME がある．

ただし，本論文は，入力方式やキーボード配列の違いについて論じることが目的ではないので，以後，各言語について標準的と思われる入力方式やキーボード配列を 1 つに定め，「日本語の IME」のように呼ぶものとする．

(2) の操作は，仮名漢字変換ソフトウェアに対する変換候補選択などの操作に相当する．例えば，上記の “多言語” の例であれば，(1) だけでは，“多言語” のほかに “他言語” という選択肢も存在する．このとき，両者から “多言語” を選び出すための入力が，(2) である．予測入力を行う場合の，キー入力から候補を予測した変換候補を選択する操作も，(2) に相当する．

(1) と (2) は，単言語で構成される文書を入力する際にも必要となるので，ユーザーインターフェースや自然言語処理の分野で詳細な研究がなされており，(MacKenzie and Tanaka-Ishii 2007) にまとめられている．

(3) の操作は，IME を切り替える操作である．本論文では，以後この操作を「IME 切り替え操作」と呼ぶ．例えば，英語と日本語を切り替える際の “Alt+半角” などのキー操作が (3) に相当する．IME 切り替え操作は，主に，多言語文書の入力時に言語を切り替える際に発生する．

1 節で述べたように，この IME 切り替え操作を，キー列からユーザーが入力している言語を判別することで，自動化するのが本論文の主旨である．

(3) の操作量を直接扱った既存研究は少ない．著者らが調査した範囲では，論文が公開されているものとしては，(Chen and Lee 2000) が該当するのみである．この論文では，ピンイン[2]と英語が混じったキー列を正しく変換するタスクが述べられている．しかし，この機能はあくまで中国語入力の一環として述べられているにすぎず，本論文で扱うように，言語の種類や数を変更することは考慮されていない．

論文の形でなく，フリーソフトウェアでは，*PUNTO switcher*[3] というロシア語圏のソフトウェアが，ロシア語と英語の間の切り替えの目的で 2001 年より開発されている．当該ウェブサイトの情報によれば，このソフトウェアは 150 万件のダウンロードがあり，一定の成功を収めていると思われる．また，*Keyboard Ninja*[4] というソフトウェアも作成されている．Keyboard Ninja は，ロシア語，英語，フランス語，ドイツ語，イタリア語，スペイン語，ウクライナ語の間での切り替えを行うソフトウェアである．これらのソフトウェアの用いているアルゴリズムや性能評価については，著者らの知る範囲では公開されていない．

[2] ローマ字による中国語の発音表記のこと．中国語入力では，ピンインを入力して漢字に変換するピンイン漢字変換が一般的である．
[3] http://punto.ru/
[4] http://www.intelife.net/ninja/

一方，本研究は，文書の言語判別問題としてとらえることも可能である．この文書の言語判別問題は，次のように分類することが可能である．

(1) ある単言語の文書が与えられ，その文書の言語を判別する問題

(2) 多言語の文書が与えられ，その中の部分が何語であるかを判別する問題

(1) の問題については，OCR を多言語に対応させることを主な目的とした，(Cavnar and Trenkle 1994) の研究が基礎的である．この論文では，ニュースグループへの投稿文書という長いテキストを対象に，文字ベースの N-gram の頻度を用いて文書の言語を判別している．また，(Sibun and Reynar 1996) では，やはり ECI コーパスという長いテキストを対象に，KL 情報量を用いて文書の言語を判別している．どちらの論文でも，言語によって判別精度に差があることと，平均して 90%を超える高い精度が達成できることを報告している．

一方，本研究との関連がより深いものは，(2) の問題である．(2) は，(1) を拡張した問題になっているうえ，多くの場合 (1) より短い部分から言語を判別しなければならないため，(1) より難しい．以下，代表的な (2) に関する研究を 2 つ挙げる．

(Murthy and Kumar 2006) は，小さいサンプルを対象とした言語判別問題に機械学習を用い，高い判別精度で解けることを報告している．しかし，この研究は，一般の文書からのサンプルを対象に 2 言語の間の判別を行うことを目的とし，インドで使用されている言語や文字に特化した素性を用いて判別精度の向上を報告するものである．本研究は，入力中のキー列を対象に 3 言語以上の間の判別も扱い，言語の種類に特に制限は設けていない点で，目的も手法も異なる．

(Alex 2005) は，ドイツ語中に混在する英語を判別する方法について論じている．この論文では，文書中に混在する他の言語を発見するタスクを，*foreign inclusion detection* (FID) と呼び，音声合成 (Text to speech) の分野で研究されてきたことを紹介している (Pfister and Romsdorfer 2003), (Marcadet, Fischer, and Waast-Richard 2005)．近年 Alex は，FID を構文解析の前処理として使用することで，構文解析の精度が向上させられることを示している (Alex, Dubey, and Keller 2007)．この FID のように，他の処理の前処理として言語判別を使用する場合は，高い精度が求められるため，対象言語について大規模なコーパスが入手可能であることが前提となる．一方，本論文の目的では，対象言語の大規模なコーパスが手に入るとは限らないため，FID の手法をそのまま適用することは困難である．

以上の関連研究を踏まえて，次の 3 節で，設計方針を立てる．

3 準備と設計方針

TypeAny に必要な機能を特定するために，図 1 に，日本語，英語，中国語の 3 言語による，多言語文書の例を挙げて説明する．中国語の下には，ピンインを表記した．

既存の手法では，図 1 の文章を入力するためには，日本語から英語への切り替えと戻す操作

ビールは，有名なGuinness beerなどをはじめ，世界中で飲まれ
ている．意外なことに，生産量世界一は中国であり，啤酒と呼ば
pijiu
れ親しまれている．青岛啤酒，燕京啤酒などが有名である．
qingdao pijiu　yanjing pijiu

図 1　日本語，英語，中国語による多言語文書の例

で 2 回，日本語から中国語への切り替えと戻す操作が 2 対あるので 4 回，合計 6 回の IME 切り替え操作が必要となる．このような IME 切り替え操作は，文字種が違う言語間だけでなく，同じアルファベットを使う英語やフランス語の間でも，アクセント記号付きの文字を入力する場合に必要になる．

一方，TypeAny では，キー列から言語を判別し，IME を自動的に切り替える．例えば図 1 では，“bi-ruha,” を日本語，“beer” を英語，“pijiu” を中国語といったように言語を判別してユーザーに提示する．言語判別が間違っていれば，ユーザーが必要に応じて言語を訂正する．このことから，大きく分けて次の 2 つの機能が必要であることがわかる．

(1)　キー列から言語を判別する機能

(2)　言語判別の結果をユーザーに提示し，訂正情報を受け取るユーザーインターフェース

(1) については 4 節で，(2) については 5 節で扱う．

一般に，文書を入力するときには，多くの言語で，文書を区切るための区切りキーを仮定することが可能である．例えば，スペースを用いた分かち書きが，その典型的な例として挙げられ[5]，この場合には，スペースキーを区切りキーとして採用することが可能である．日本語や中国語では，通常，分かち書きは行わないが，漢字に変換する際にスペースキーを打鍵しているので，やはりスペースキーを区切りキーとして使用することが可能である．

そこで，TypeAny では，区切りキーから次の区切りキーまでに挟まれるキー列を「トークン」と定義する．トークンを単位として入力を行い，トークンの属すべき言語[6]を判別する．例えば，図 1 におけるトークンとしては，“bi-ruha,”，“beer”，“pijiu” が挙げられる．トークンは単語とは限らず，連文節変換を用いる場合などは，単語よりも長い単位となる場合もある．

トークンによっては，そもそも，その属すべき言語が曖昧である場合がある．例えば，“sushi” というトークンは，英語としての “sushi” にも日本語の “寿司” に対する入力にもとらえることが可能である．借用語の多くにこのような曖昧性がある．また，ヨーロッパの多くの言語で使

[5] スペースを用いた分かち書きは，英語やフランス語などアルファベットを用いる言語に限らず，字種の異なるアラビア語，ヘブライ語，韓国語などでも行われている一般的な習慣である．

[6] 厳密には，ユーザーが入力しようとしているキーボード配列 (Qwerty, Dvorak, Azerty など) も同時に判別している．

用されるアクセント記号はしばしば省かれることがあり，この場合にも曖昧性が生じる場合がある．例えば，“fur” は通常は英単語であるが，“für” というドイツ語のウムラウト記号が省かれた形としても，使用されることがあり，英語とドイツ語の間で曖昧性が生じる．

ただし，このような曖昧性は，実用上は必ずしも問題とならない場合もある．ユーザーは，例え言語判別に失敗していても，最終的に入力したトークンが正しい文字列に変換されていれば問題とは認識しないと考えられる．例えば，上記の後者の例である “fur” は，英語とドイツ語の間で曖昧性があるものの，どちらに判別されたとしても，最終的には “fur” に変換されるため，問題を生じない．一方，上記の前者の例である “sushi” では，英語と判別された場合は “sushi” と変換され，日本語と判別された場合は “寿司” などに変換されるため，ユーザーの観点からは問題を生じる．以上のように，この曖昧性が問題となるか否かを判定することは，個々の言語に対する具体的な知識を必要とするため難しい．そこで本論文では，6 節で示すように，単純に言語判別の精度を用いて評価を行った．

トークンの属すべき言語は，事前に用意する学習コーパスが多くなるほど明確に判別することが可能になるが，その分，TypeAny が対応可能な言語は限られてくる．TypeAny は入力システムであるため，多くの言語に対応可能であることが望ましいと考えた．そこで，より多くの言語に対応を優先する設計方針を立て，言語判別を 4 節で述べるように設計した．また，その言語判別を用いるユーザーインターフェースを 5 節で述べるように設計した．

4 言語判別

4.1 言語判別モデル

言語判別の確率モデルには，隠れマルコフモデル (HMM) を用いた．隠れ状態を言語として，隠れている言語からトークンが記号列として観測されるとする．

ここでの目的は，$\mathrm{P}(l_1^m, t_1^m)$ を最大にするような $\hat{l}_1^m$ を求めることである[7]．ただし，$l \in L$ は言語集合 L 中の言語であり，t はトークンである．HMM では，$\mathrm{P}(l_1^m, t_1^m)$ を数式 (1) のようにして最大化する．

$$
\begin{aligned}
\hat{l}_1^m &= \operatorname*{argmax}_{l_1^m \in L} \mathrm{P}(l_1^m, t_1^m) \\
&= \operatorname*{argmax}_{l_1^m \in L} \mathrm{P}(t_1^m | l_1^m) \mathrm{P}(l_1^m) \\
&\approx \operatorname*{argmax}_{l_1^m \in L} \left(\prod_{i=1}^{m} P(t_i | l_i) \right) \left(\prod_{i=1}^{m} P(l_i | l_{i-k}^{i-1}) \right) \qquad (1)
\end{aligned}
$$

[7] ここで，t_u^v は，$v \geq u$ のとき $v-u+1$ 個の要素からなる列 $t_u^v = (t_u, t_{u+1}, \ldots, t_v)$ を表し，$v < u$ のとき空列 $t_u^v = ()$ を表す．

数式 (1) では，第一項を $\mathrm{P}(t_1^m|l_1^m) \approx \prod_{i=1}^{m} P(t_i|l_i)$ のように，また，第二項を $\mathrm{P}(l_i|l_1^{i-1}) \approx P(l_i|l_{i-k}^{i-1})$ のように，近似した．ここで，第一項は出力確率であり，第二項は遷移確率である．

4.2 出力確率の推定

出力確率 $P(t_i|l_i)$ は，ある 1 つの言語 l_i からトークン t_i が出力される確率である．トークンを単語とみなせば，この確率は単純に言語 l_i における単語の出現確率であり，その推定手法は自然言語処理の分野において，よく研究されている．

$P(t_i|l_i)$ の推定するには，言語 l_i のコーパスが必要となる．十分に大規模な言語 l_i のコーパスを用いれば，$P(t_i|l_i)$ は，単純にコーパス中にトークン t_i が出現した頻度で近似することが可能である．しかし，この方法は，入力可能な言語を大規模なコーパスを入手することが可能な言語に限定してしまうため，5 節の最後で述べた，より多くの言語に対応するという方針に反する．

そこで，本研究では，トークンを入力したキーの列と捉え，キー列に関するスムージングを行うことで，$P(t_i|l_i)$ を計算する方法を採用した．まず，トークン t_i の長さを $|t_i|$ とし，トークン t_i をキー列 $c_1^{|t_i|} = (c_1, c_2, c_3, \ldots, c_{|t_i|})$ として捉える．すなわち，$t_i = c_1^{|t_i|}$ とする．例えば，t_i="pijiu" の場合，$|t_i| = 5$ で，$t_i = c_1^5 =$ ('p', 'i', 'j', 'o', 'u') となる．次に，この $c_1^{|t_i|}$ について，最大 n_{max} までの $n+1$-gram 確率を計算することで，スムージングを行い，$P(t_i|l_i)$ を次のように計算する．

$$P(t_i|l_i) = P(c_1^{|t_i|}|l_i) \approx \prod_{r=1}^{|t_i|} P(c_r|c_{r-n_{max}+1}^{r-1}, l_i) \tag{2}$$

このスムージングの手法としては，さまざまなものが提案されているが，本研究では，*Prediction by Partial Matching* (PPM) という手法を採用した．この PPM は，データ圧縮の分野で最初に提案され，後に自然言語処理に応用された手法である (Bell, Clear, and Witten 1990), (Teahan, McNab, Wen, and Witten 2000), (Tanaka-Ishii 2006).

PPM は，データ圧縮の分野で提案されたため，学習を動的に行いながら判別を行うことが，可能なように設計されているという特徴がある．この特徴によって，ユーザーが誤判別を訂正した場合，瞬時にその情報を確率値にフィードバックして次の判別に利用することが可能となる．この点が，TypeAny のような入力システムに適した特徴であると考えたので採用した．以下，PPM の詳細について説明する．

PPM は，$c_1^{r-n_{max}}$ までの頻度情報をもとに，現在の文脈 $c_{r-n_{max}+1}^{r-1}$ の次にキー c_r がくる確率 $P(c_r|c_{r-n_{max}+1}^{r-1})$ を推定する．

$$P(c_r|c_{r-n_{max}+1}^{r-1}) = \sum_{n=-1}^{n_{max}-1} w_n p_n(c_r) \tag{3}$$

ここで $p_n(c_r)$ は，次のように，長さ n の文脈にキー c_r が続く $n+1$-gram 確率を表す．X_n, x_n は，それぞれ，c_1^{r-n-1} 中の c_{r-n}^{r-1}, c_{r-n}^{r} の頻度とする．

$$p_n(c_r) = \frac{x_n}{X_n}$$

数式 (3) において，重み w_n は，基本的には，長い $n+1$-gram 確率を重く，短い $n+1$-gram を軽く重みづけるのが望ましい．ただし，重みが偏りすぎることも精度を悪化させる．PPM では，この重み w_n を，簡単な計算で適切に設定するために，エスケープ確率 e_n という概念を導入して，次のように計算する．

$$\begin{aligned} e_{-1} &= 0 \\ w_n &= (1-e_n)\prod_{n'=n+1}^{n_{cont}} e_{n'} \quad (-1 \le n < n_{cont}) \\ w_{n_{cont}} &= (1-e_{n_{cont}}) \end{aligned} \tag{4}$$

ただし，n_{cont} は，$X_n \neq 0$ を満たす $n_{max}-1$ 以下で最大の n である．エスケープ確率 e_n は，現在の文脈に一度も続かなかったキーに割り当てる確率を表す．すなわち，現在の長さ n の文脈 c_{r-n+1}^{r-1} に一度も続かなかった新しいキー（これを「エスケープ」と呼ぶ）が，エスケープ確率 e_n で現れると考える．反対に，現在の長さ n の文脈 c_{r-n+1}^{r-1} に続いたことのあるキーは，エスケープ確率 e_n を新しいキーに割り当てた分を減らし，単純な頻度に $1-e_n$ 倍をした確率で出現すると考える．

このエスケープ確率をどのように定義するかによって，PPM は，PPMA，PPMB，PPMC のように分類される．その中でも，本研究では，基礎的かつ比較的性能が高いとされる PPMC を用いた[8](Bell et al. 1990)．

PPMC では，エスケープ確率を次のように計算する．ただし，q_n は，c_1^{r-n-1} 中の，c_{r-n}^{r-1} のあとに続くキーの異なり数である．

$$e_n = \frac{q_n}{X_n + q_n} \tag{5}$$

数式 (5) から，PPMC では，次のキー c_r の確率 $P(c_r|c_{r-n_{max}+1}^{r-1})$ は，キー列の n-gram の頻度 X_n と n-gram の後に続くキーの異なり数 q_n が分かれば，推定することが可能であることがわかる．n-gram の頻度と n-gram の異なり数は単純な加算によって学習中に更新することが容易であるため，PPMC は動的に学習することに適している．

[8] PPM は大規模圧縮においてこそ性能の差が問題となるが，入力応用上はどの PPM を用いても，大きな差にはつながらないとの報告もある (Tanaka-Ishii 2006)．

4.3 遷移確率の推定

ここでは，数式 (1) の第 2 項である，k_{max}-gram までの，言語 $k+1$-gram による文脈を考慮した遷移確率 $P(l_m|l_{m-k_{max}+1}^{m-1})$ を推定する手法について述べる．この遷移確率は，大量の多言語文書から学習することが可能であるが，そのような大量の多言語文書は，通常，入手することが難しい．ユーザーが過去に確定した言語列 l_1^{m-1} を正解とみなし，l_1^{m-1} から動的に遷移確率を推定することが可能であれば，この学習データの入手の問題を回避することが可能となる．

この方法は，4.2 節と同様で，学習データが少量であることを，利用中のユーザーからの情報を動的に利用して補い，精度を向上させることが狙いである．したがって，遷移確率の推定方法には，4.2 節と同様，PPM を用いた．具体的には，数式 (3) における c_r を l_m と読み替えることで，遷移確率 $P(l_m|l_{m-k_{max}+1}^{m-1})$ を分解して推定した．

4.2 節で述べた出力確率の推定の場合との違いの一つは，遷移確率は，出力確率ほど出現位置の離れた要素に依存しない，すなわち，長距離依存性が小さいことである．これは，次のように考えれば直感的に理解することが可能である．たとえば，言語 3-gram を考えた場合，英語，フランス語，日本語のトークンがこの順番で何回も出現する文書は，まれであると推測される．したがって，通常は，遷移確率の最大文脈長 k_{max} を，出力確率の最大文脈長 n_{max} より小さく取り，$k_{max} \leq n_{max}$ としてよい．

ただし，実用上は，これらの最大文脈長の値はある程度の大きさがあれば十分であり，これらの値を細かく調整する必要性は乏しい．その理由は，数式 (3) のように，PPM では文脈の長さごとに文脈の重要度 w_n が自動的に決定されるためである．本研究では，特別な事情がない場合は $k_{max} = n_{max} = 5$ とした．

5 ユーザーインターフェース

ここでは，前節で述べた言語を判別する手法を，ユーザーインターフェースに組み込む方法について説明する．システムの構造を，図 2 に示す．

TypeAny は，図 2 に示すように，ユーザーのキー入力と IME の間に立って両者を仲介する．まず，ユーザーが入力したキー列を，クライアントが受け取り，クライアントはそのキー列をサーバーに送る．サーバーでは，サーバー内の「言語判別モジュール」がキー列からユーザーが入力しようとしている言語を判別して，対応する IME に送る．IME では，キー列を文字列に変換して，クライアントに送り返す．この中の言語判別モジュールに，前節で述べた言語判別手法を実装し，組み込んだ．

フランス語やドイツ語などヨーロッパ系の言語では，キー列に対して文字列が一意に定まるので，IME は，単純な置き換えですむ．たとえば，ドイツ語の IME では，日本語のキーボードで “@” に対応するキーを，ドイツ語の “ü” に置き換えている．一方，日本語や中国語では，

キー列に対して文字列が一意に定まらないので，IME がユーザーに候補を提示して選択してもらう必要がある．この処理には，既存のかな漢字変換／ピンイン漢字変換のシステムをそのまま用いればよい．日本語の IME には，Anthy[9]を用い，中国語の IME には，単純なピンイン漢字変換を自作した．

図 3 に，TypeAny を用いて図 1 に示す文章の入力例を示す[10]．各ステップにおいて，白黒反

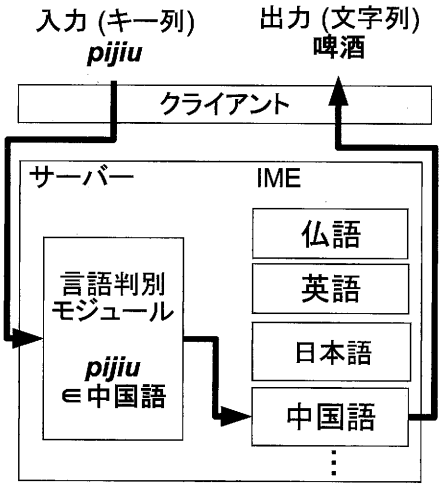

図 2　TypeAny の構造

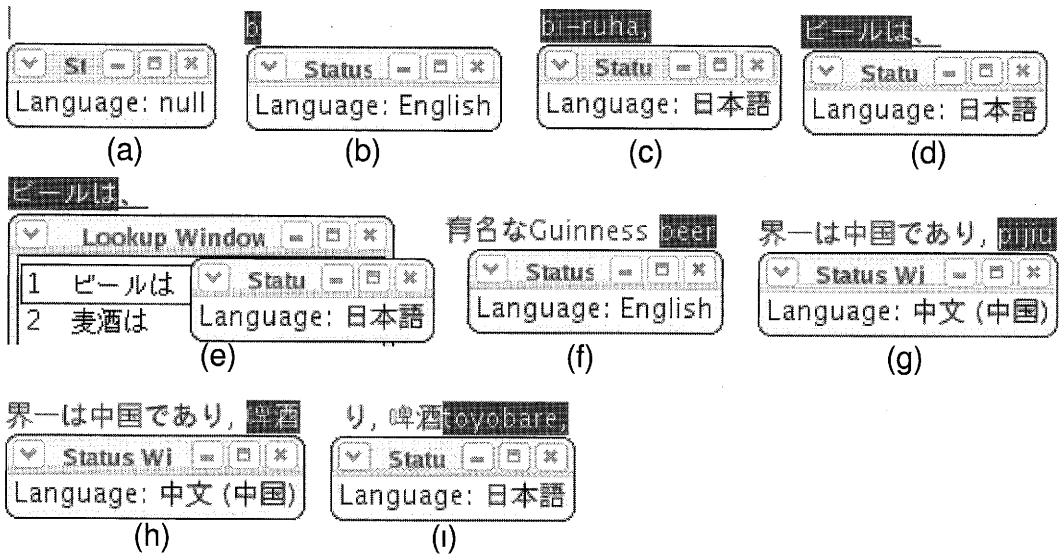

図 3　TypeAny を用いた入力操作

[9] http://anthy.sourceforge.jp/
[10] ここでは，Qwerty 配列上での入力を仮定している．日本語はローマ字入力，中国語はピンイン入力とする．

転されているところが，ユーザーが入力中の部分である．言語の判別は，反転部分のキー列に対して行われ，その結果が *Locale Window* に表示される．以下，各ステップを説明する．

(a) 初期の状態では，どの言語も選択されていない．

(b) キーを押すごとに反転部分のキー列（トークン）から言語が判別され，*Locale Window* に表示される．“bi-ruha,” の “b” を打鍵した時点では，英語と判別されていることがわかる．

(c) しかし，“bi-ruha,” まで打鍵すると，正しく日本語と判別される．

(d) 現在のトークンは日本語と判別されているので，デリミタとなるスペースキーを打鍵すると，日本語の IME を通じて日本語の文字列への変換が行われる．

(e) 日本語のように，キー列文字列への変換候補が複数ある場合は，さらにスペースキーを打鍵することによって，通常のかな漢字変換を行うことが可能である．

(f) “beer” というトークンが，正しく英語と判別されている．

(g) “pijiu” というトークンが，正しく中国語と判別されている．

(h) (g) でスペースキーを打鍵すると，日本語のかな漢字変換と同様に，中国語のピンイン漢字変換が行われる．

(i) その後の “toyobare,” というトークンも，正しく日本語と判別されている．

このように，TypeAny を用いることで，ユーザーは，言語の誤判別が発生しない限り IME 切り替え操作を行う必要がなく，ユーザーの負担は大幅に軽減される．

トークンの言語を判別した結果が，ユーザーの望むものと異なる場合は，「誤判別」となる．誤判別時の処理を，図 4 を用いて説明する．

言語判別の結果は常に *Locale Window* に表示されるので，誤判別の場合を含め，ユーザーはその結果を常に把握することが可能である．したがって，誤判別の場合でも，TAB キーを押すことで，ユーザーは IME を手動で簡単に切り替えることが可能である．例えば，図 3 の “pijiu” は正しく中国語と判別されているが，図 4(a) のように，間違って日本語と判別されていたと仮定する．ここで，ユーザーが TAB キーを押すと，IME が図 4(b) のように中国語に切り替わる．

TypeAny における誤判別は，次の 2 種類に分類される．

誤判別 1: 言語が切り替わるべき時に，言語が切り替わらなかったか本来の言語とは違う言語

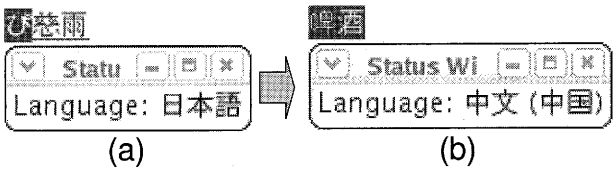

図 4　誤判別時の操作

に切り替わった場合.

誤判別 2: 言語が切り替わるべきでない時に，言語が切り替わってしまう場合.

既存手法では，言語を切り替えるごとにIME切り替え操作を行わなければならなかったのに対し，TypeAnyでは，IME切り替え操作は言語判別を失敗したときのみ必要になる．**誤判別 1**は，TypeAnyが言語判別を間違えた場合であってもIME切り替え操作回数が増える原因とはならない．一方，**誤判別 2**は，特に多言語コーパスの大半が1言語から構成されているような場合において，既存手法と比較した場合のIME切り替え操作回数を増加させてしまう可能性がある．したがって，TypeAnyの有効性は，言語を切り替える点での自動判別によるIME切り替え操作の減少量と，言語を切り替えるべきない点での**誤判別 2**によるIME切り替え操作の増加量とのトレードオフによって決まる．このトレードオフについては，6.2節で論じる．

6 評価

TypeAnyを2つの観点から評価した．まず，人工的に作成した多言語なコーパスを用いて，言語判別の精度を測定した．次に，実際の多言語文書を入力した場合の，打鍵数の減少量を測定した．

6.1 言語判別精度による評価実験

ここでは，言語判別の精度を測定した．言語判別の精度を測定するためには，実際に多言語を含む十分な量の正解コーパスがあることが望ましい．しかし，そのようなコーパスは，通常，入手することは難しい．

そこで，本論文では，人工的に2言語から8言語の多言語コーパスを作成して，言語判別の精度を測定した．最初に，単言語のコーパスを収集した．日本語コーパスには毎日新聞2004年度版，中国語コーパスには北京大学コーパス，その他，英語，フランス語，ドイツ語，エストニア語，フィンランド語，トルコ語のコーパスには，Leipzig corpora (Biemann, Heyer, Quasthoff, and Richter 2007) を用いた．各言語の文書は，事前にキー列に変換した[11].

混合率による判別精度の変化をみるため，テストセットを2つ作成した．テスト1では，どの言語の出現確率も等確率であるような多項分布から生成した．テスト2では，メインとなる言語が90%を占め，残りの10%は残りの言語が均等に配分されるような多項分布から生成した．実際の多言語文書では，メインとなる言語が存在するので，テスト2の方がより現実的な状況に即している．

[11] 日本語の文書は，MeCab (http://mecab.sourceforge.net/) を用いて読みに変換した．また，日本語の“し”がキー列としては“shi”にも“si”にもなるように，文字に対して複数のキー列が対応する場合は，事前に定めた確率を用いて対応するどのキー列にも文字が変換される可能性が残るようにした．

出力確率と遷移確率は，4節で述べたPPMCによって推定する．実際のシステムでは，どちらの確率も動的に学習されるのであるが，今は判別精度を評価することが目的であるため，事前に準備した訓練コーパスを用い，テスト中は動的な学習を行わない．4.2節の数式(4)にあるように，出力確率は$n_{max}=5$で学習した．また，今回は多項分布から人工的にコーパスを生成しているため，遷移確率は$k_{max}=1$とした．

評価は，生成した多言語コーパス上での10回交差検定を用いた．訓練コーパスのサイズは100 Kbyte，テストデータは11 KByteとした．出力確率は事前に各言語の訓練コーパスを用いて学習し，遷移確率は約2000トークンを用いて学習した．

テスト1，テスト2の両者の結果を，それぞれ，図5と図6に示す．横軸は言語数を表し，縦軸は判別精度を表す．各言語数ごとに，全ての言語の組み合わせについて言語セットを作成した．各言語セットごとに10回交差検定を用いて判別精度を測定した後，全ての言語セットについて判別精度を平均した値をプロットした．凡例中の「PPM」は遷移確率もPPMを用いて学習させた場合，「ML」は出力確率のみを用いて判別させた場合（最尤推定に相当），「Baseline」は最も頻度の高い1言語を常に正解として返す場合である．

どの言語も等確率で出現するテスト1（図5）では，PPMとMLの精度が非常に近くなっている．これは，PPMが遷移確率を学習を通して，どの言語も等確率で出現していることを学習したためであり，遷移確率の学習を無限に行えば，理論的にはPPMの精度とMLの精度は一致すると考えられる．

一方，テスト2（図6）では，PPMがMLより明らかに高い精度を示している．PPMは遷移確率を学習することで，主となる言語が90%を占めていることを学習する一方，MLでは遷移確率を学習しないため，このような結果となる．また，先行研究では調査がなされていない3言語以上の場合では，MLはベースラインより下がってしまうことがわかった．この結果から，各言語について少量のコーパスしか入手できない場合でも，単純にMLを使って言語を最尤推

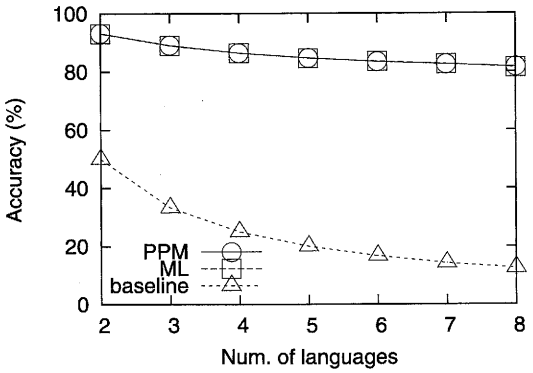

図 5　言語判別精度実験テスト1

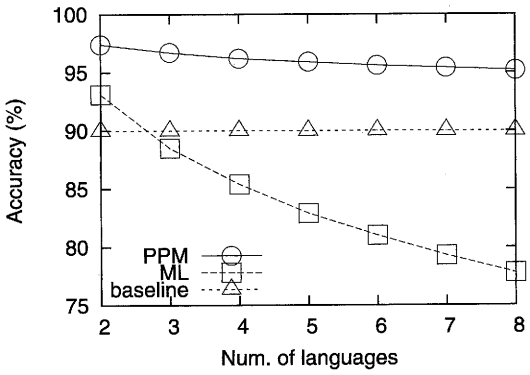

図 6　言語判別精度実験テスト2

定するのではなく，遷移確率をPPMを用いて推定することによって，言語判別の精度を向上させることが可能であると考えられる．

また，言語セットごとに，判別精度に差がみられることも注目に値する．例えば，綴りが似通った単語の多い英語とフランス語の両方を含む言語セットでは，他の言語セットよりも精度が落ちる傾向が見られる．実際に，テスト2で，90%が英語，5%がフランス語，5%がドイツ語であるような，ある言語セットでの判別精度は，3言語の言語セット全体の平均より低い94.4%であった．一方，90%が英語，5%がフィンランド語，5%がトルコ語であるような，ある言語セットでの判別精度は，3言語の言語セット全体の平均より高い97.5%であった．

6.2 IME切り替え操作回数による評価実験

次に，TypeAnyの実用性を評価するため，実際の多言語文書を入力した場合に，IME切り替え操作回数が減少する量を測定した．5節で述べたように，キー操作は3種に分類されるが，TypeAnyが関わるのはIME切り替え操作回数のみであるため，これを測定した．

この実験のために，2種類の多言語文書をWebから取得した．各文書の詳細を表1に示す．両文書とも，主となる言語は英語であり，文書1では日本語が，文書2では日本語と中国語のトークンが混在している．文書2では，文書の大半98.9%が英語のトークンである点が特徴的である．

各文書を，既存手法を用いて入力した場合と，TypeAnyを用いて入力した場合の，それぞれのIME切り替え操作回数を比較した．既存手法では，言語が切り替わるたびにIME切り替え操作を行わなければならない．一方，TypeAnyでは，言語判別に失敗した場合のみIME切り替え操作を行えばよい．誤判別時に目的のIMEに切り替えるための操作は，既存手法と同じく，1回で行うことができるものとした．

出力確率は，前述の評価で用いた各言語100 Kbyteのコーパスより学習させた．遷移確率については，事前の学習は行わない．すなわち，実験開始時点では，TypeAnyはどの言語が入力されるか分からず，各言語が一様に入力されるものと想定している．具体的には，TypeAnyは，

表1 IME切り替え操作回数による評価実験で使用した文書

	文書1	文書2
他言語のトークン	286	55
トークン合計	1725	5100
他言語のトークン比率	16.6%	1.1%
言語	英，日	英，日，中
内容	英語母国語者向けの日本語旅行会話集	豆腐の解説
取得元	Wikitravel (http://en.wikitravel.org/)	Wikipedia (http://en.wikipedia.org/)

表 2　文書入力に必要な IME 切り替え操作回数

		文書 1	文書 2
既存手法による IME 切り替え操作回数		572 (100%)	110 (100%)
TypeAny 使用時	**誤判別 1**	2.8%	3.6%
	誤判別 2	1.6%	2.7%
誤判別合計		4.4%	6.3%
IME 切り替え操作回数の減少量		95.6%	93.6%

この実験の開始時点で，文書 1 では英語と日本語を一様に，文書 2 では英語と日本語と中国語を一様に，ユーザが入力するものと想定している．この実験は，実際に多言語文書を入力する場合に，IME 切り替え操作回数が減少する量を測定することが目的であるため，実験中は，出力確率も遷移確率も PPMC を用いて動的に学習されるようにした．特に遷移確率の学習も行っているため，この実験では TypeAny は，入力されているテキストにおける各言語の比率についても学習していく．考慮される文脈の長さについては，$n_{max} = k_{max} = 5$ とした．

実験の結果を，表 2 に示す．実験の初期段階では，文書 2 において，英語であるべき文書 2 中の “tofu” が日本語として判別されてしまう誤判別が起こった．この誤判別は，借用語の曖昧性が原因であり，5 節で予想された結果である．実際，TypeAny は “tofu” が英語と判別されるべきであることを PPMC を用いて学習したため，“tofu” による誤判別は実験の初期段階にとどまり，実験の後半では発生しなかった．

結果として，TypeAny を用いた場合，両文書とも，既存手法と比較して 93%を超える IME 切り替え操作回数の減少が認められた．特に，文書 2 で IME 切り替え操作回数が減少したことは，重要な知見である．5 節で述べたように，TypeAny の有用性は，**誤判別 2** に依存する．文書 2 は英語が 98.9%と大半を占めるにも関わらず，文書 1 と比較して，誤判別 2 は僅かしか増加していない．この結果は，5 節の最後で述べた懸念を払拭するものである．すなわち，入力頻度の少ない言語において，IME 切り替え操作回数を増加させる**誤判別 2** が起こった場合でも，訂正を繰り返し行うことで TypeAny が学習し，以後の**誤判別 2** を防ぐことが可能である．この実験では，文書 2 において最大 3 言語間の判別を行ったが，より多くの言語をサポートした場合でも，現実的には同様にして**誤判別 2** を防ぐことが可能であると考えられる．以上より，これらの結果は，TypeAny が既存手法と比較して有用であることを示唆している．

7　結論

TypeAny は，ユーザーが入力したキー列から言語を判別して，IME を自動的に切り替えるこ

とで，多言語入力におけるユーザーの負担を軽減するシステムである．言語判別は，隠れマルコフモデルとしてモデル化した．事前に各言語の少量の学習コーパスのみを用意し，出力確率も遷移確率も入力に伴い動的に学習させることで，多くの言語に容易に対応することを優先した．これを達成するため，PPM 法を用いた．

評価実験の結果，現実的な，1 つの言語が 90%を占める 3 言語からなる多言語文書において，96.7%の判別精度を得た．また，実際に多言語文書を入力した場合，既存手法と比較して IME 切り替え回数が 93%減少した．これらの結果より，TypeAny を用いることで多言語文書を効率的に入力することが可能であることが示唆された．

今後の課題としては，識別モデルを用いて精度を向上することや，IME を頻繁に切り替える必要のある語学教材の作成を容易にするシステムとして語学教育分野に応用すること，携帯端末など計算機の性能に制限がある状況でも幅広く利用可能にすることなどが挙げられる．

参考文献

Alex, B. (2005). "An Unsupervised System for Identifying English Inclusions in German Text." In *Proceedings of the ACL Student Research Workshop*, pp. 133–138 Ann Arbor, Michigan. Association for Computational Linguistics.

Alex, B., Dubey, A., and Keller, F. (2007). "Using Foreign Inclusion Detection to Improve Parsing Performance." In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pp. 151–160.

Bell, T. C., Clear, J. G., and Witten, I. H. (1990). *Text Compression.* Prentice-Hall, New Jersey.

Biemann, C., Heyer, G., Quasthoff, U., and Richter, M. (2007). "The Leipzig Corpora Collection—Monolingual corpora of standard size." In *Proceedings of Corpus Linguistics 2007* Birmingham, United Kingdom.

Cavnar, W. B. and Trenkle, J. M. (1994). "N-Gram-Based Text Categorization." In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR '94)*, pp. 161–175 Las Vegas, NV, USA.

Chen, Z. and Lee, K.-F. (2000). "A New Statistical Approach To Chinese Pinyin Input." In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*, pp. 241–247 Hong Kong. Association for Computational Linguistics.

Ehara, Y. and Tanaka-Ishii, K. (2008). "Multilingual Text Entry using Automatic Language Detection." In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008)*, pp. 441–448 Hyderabad, India.

MacKenzie, I. S. and Tanaka-Ishii, K. (2007). *Text Entry Systems: Mobility, Accessibility, Universality (Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Marcadet, J.-C., Fischer, V., and Waast-Richard, C. (2005). "A Transformation-Based Learning Approach to Language Identification for Mixed-Lingual Text-to-Speech Synthesis." In *Proceedings of Interspeech 2005*, pp. 2249–2252 Lisbon, Portugal.

Murthy, K. N. and Kumar, G. B. (2006). "Language identification from small text samples." *Journal of Quantitative Linguistics*, **13** (1), pp. 57–80.

Pfister, B. and Romsdorfer, H. (2003). "Mixed-Lingual Text Analysis for Polyglot TTS Synthesis." In *Proceedings of Eurospeech 2003*, pp. 2037–2040 Geneva, Switzerland.

Sibun, P. and Reynar, J. C. (1996). "Language Identification: Examining the Issues." In *Proceedings of the 5th Symposium on Document Analysis and Information Retrieval (SDAIR '96)*, pp. 125–135 Las Vegas, NV, USA.

Tanaka-Ishii, K. (2006). "Word-based Text Entry Techniques Using Adaptive Language Models." *Journal of Natural Language Engineering*, **13** (1), pp. 51–74.

Teahan, W. J., McNab, R., Wen, Y., and Witten, I. H. (2000). "A compression-based algorithm for Chinese word segmentation." *Computational Linguistics*, **26** (3), pp. 375–393.

略歴

江原　遥：2007年東京大学工学部計数工学科卒業．現在，東京大学大学院情報理工学系研究科創造情報学専攻修士課程学生．自然言語処理，また，その教育への応用に興味を持つ．言語処理学会学生会員．ehara@r.dl.itc.u-tokyo.ac.jp

田中久美子：現在，東京大学大学院情報理工学系研究科准教授．言語処理学会，ソフトウエア科学会，情報処理学会会員．専門は計算言語学，自然言語処理．言語に内在する数理情報論的構造に関する研究に興味を持つ．kumiko@i.u-tokyo.ac.jp

(2008 年 4 月 21 日　受付)
(2008 年 7 月 10 日　再受付)
(2008 年 7 月 11 日　採録)