入力支援機能を統合した多言語入力システムUniversal Text Input

鈴木久美, Pallavi Choudhury, Chris Quirk,
Chris Wendt, Colin Yu, Abdulaziz Mohammed, Vikram Dendi
Microsoft Research
One Microsoft Way
Redmond WA 98052, USA
{hisamis, pallavic, chrisq, christw, colinyu,abdulsm, vikramde}@microsoft.com

概要

この論文では、どのキーボードからでも何語の文字でも発音に従って簡単に入力ができる多言語入力システム Universal Text Input を紹介する。近年、発音に基づいた文字入力システムが日本語や中国語にとどまらず用いられるようになってきた。また、企業の多国籍化などで、非母国語話者がテキスト入力をする機会も増えてきている。そのような背景を踏まえ、本プロジェクトでは入力時のスペル訂正機能も備えた多言語入力システムを構築し、ウェブサービスとして公開している。2012年1月現在、7言語(アラビア語、中国語、英語、フランス語、ギリシャ語、日本語、ロシア語)をサポートしており、今後も言語・機能ともに充実させていく予定である。

1　はじめに

本論文では、Universal Text Inputプロジェクトを紹介する。Universal Text Inputは多言語入力を支援するウェブサービスで、昨年秋から公開されており、[1]誰にでも使いやすい普遍的な入力サービスの開発を目指している。図1にホームページの画面を示す。ソーシャルメディアなどのインフラが発達し、そのコメントやツイートなどを通して外国語のテキストを目にする機会がこれまでになく増えている現在、機械翻訳を利用してそれらを消費できるようにすることは今後の言語処理技術の重要な方向性のひとつだと言えるだろう。と同時に、そのようなテキストの入力を容易にする技術が発達することで、ネット上での言語の多様性をサポートすることができるのではないか。本プロジェクトはそのような動機に基づいている。

本プロジェクトはまだ立ち上がってから日が浅いが、現段階では以下の三点を中心に研究・開発を進めている。

(1) 文字変換によるテキスト入力: 発音により文字入力する方法は、これまで日本語ではかな漢字変換、中国語ではピンイン入力として長い歴史があり広く使われてきたが、ここ数年でアラビア語やインドの諸語など、日中以外

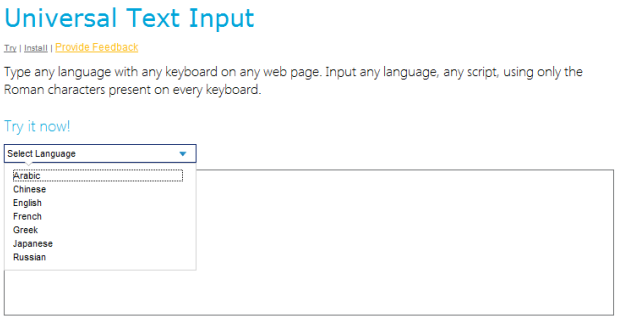

図1: Universal Text Input ホームページ画面

の非ローマ字語圏でも使われるようになってきている。[2]また、韓国語やタイ語などは、ハングルやタイ文字に特化したキーボードがあり、慣れると高速に入力ができるそうだが、慣れていないユーザーにはお手上げであり、新しいキーボード配列をマスターすることなくこれらの言語を入力できるようになれば、外国語学習者やそのようなキーボードが手元にない環境で非常に有益である。

(2) 入力ミスを想定したテキスト入力: 入力システムが誰にでも使いやすくなるためには、入力ミスを想定したものでなければならない。入力ミスには、手の動きによるものや文字列の頻度に左右されるもの、また発音や綴りがうろ覚えであるためにおこるものなど、さまざまなタイプがあり、これらをよりよくサポートするためには、ユーザーデータの収集・モデル化が不可欠である。

(3) ウェブサービスとしてのテキスト入力: Universal Text Inputがウェブサービスとして構築されている理由は、サーバーの豊富なリソースが使えること、アップデートが随時だという利点のほかに、ユーザーフィードバックデータの収集とその活用をプロジェクトの中心に据えていることが理由である。使えば使うほどよくなる入力システムいう考え方は以前から提唱されており[15]、Universal Text Inputはそのような方向性を大規模・多言語システムで追及していこうとする試みでもある。

[1] http://labs.microsofttranslator.com/uime/

[2] その例としてGoogle Transliterate (http://www.google.com/transliterate/)、Microsoft Maren (http://www.microsoft.com/middleeast/egypt/cmic/maren/)、ILIT (http://specials.msn.co.in/ilit/Hindi.aspx)、Quillpad (http://quillpad.in/)などがあげられる。

本稿では、まず「文字列変換としてのテキスト入力」という側面からUniversal Text Input が文字種変換とスペル訂正をどのように扱っているかを次節で説明する。また、ウェブサービスとしてのUniversal Text Inputを Amazon Mechanical Turkによるデータ収集に利用した一例も3節で紹介する。

2 文字列変換としてのテキスト入力

2.1 関連研究

文字変換による入力タスクとスペル訂正タスクは、どちらも文字列変換タスクとして互いに似通っている。文字列変換にはほかにもさまざまなタスクがあり、言語処理研究の対象となってきた。スペル訂正タスクは1960年代から研究されているタスクだが、その難しい点は、訂正前と訂正後のペアデータが自然状態では普通存在しないため、実データの入手が困難なことである。このため、たとえばthereとtheirなど綴りの似ている語をコンテキストによってどちらが妥当か判定する曖昧性解消タスク(e.g. [7])や、辞書にない単語を辞書にある単語に変換・訂正するタスク(e.g. [1])がよく用いられる。近年では、データが豊富にある検索クエリの書き換えログ利用したスペル修正もさかんに研究されている(e.g. [6])。また、ウェブでの頻度を使用してペアデータを使わずにスペル訂正を行う手法も提案されている[13].

スペル訂正とならんで先行研究の多い文字列変換タスクは翻字 (transliteration) である。なかでも、ある文字列を別の文字種を使った別の言語へ変換するタスク、例えばカタカナ語「スパゲッティ」を英語 spaghetti に逆変換するタスク種のはカタカナ語のみならず広く研究されている[10]。このタスクでは、コンテキストによる曖昧性解消はあまり問題になることはなく、その点が日中のテキスト入力タスクとの最大の相違点でもある。

これらに共通するアプローチとしてノイズのある通信路モデルが広く用いられてきた(e.g. [8,9])。入力文字列をS、最適な出力文字列解をT*とすると、ベイズの法則により

$$T^* = \text{argmax}\, P(T|S) = \text{argmax}\, P(T)P(S|T)$$

となる。ここで$P(T)$は言語モデル確率であり、出力のもっともらしさを評価する。$P(S|T)$は出力Tからノイズのある通信路によって入力Sが生成される確率であり、タスクごとにエラーモデル、翻字モデル、かな漢字変換モデルなどと呼ばれる。また、ノイズのある通信路モデルの拡張にあたる線形モデルも近年これらのタスクで共通して使われている(e.g., [4,12])。線形モデルでは、出力の事後確率$P(T|S)$を直接以下のモデルを用いて推定する。

$$P(T|S) = \frac{1}{Z(S,T)} exp \sum_i \lambda_i h_i(S,T)$$

$Z(.)$は正規化項、h_i は素性関数、λ_i は素性関数の重みである。線形モデルは、ノイズのある通信路モデルよりも柔軟な素性設計が可能なため、近年の統計的機械翻訳も含めて幅広く使用されている。上述のタスクにおいても、ノイズのある通信路モデルを凌ぐ結果が報告されている。

2.2 文字列変換としての入力

統計的かな漢字変換やピンイン入力もまた、ノイズのある通信路モデルの枠内で研究・実装されてきた(e.g.,[3,17])。Universal Text Inputもそのシンプルな拡張版となっている。たとえばUniversal Text Inputでサポートしている言語のひとつであるフランス語の入力を考えてみよう。ここでのタスクはアクセントのない文字列Sを正しくアクセントのついた文字列Tに変換することである。

S: un eleve va a l'ecole
T: un élève va à l'école
（英訳）a student goes to school

これは、本当はアクセントをつけなければならないのに誤ってつけ忘れたとみれば、スペル訂正タスクと見ることができる。Universal Text Inputでは線形モデルを採用し、エラーモデルから得られる素性と言語モデル素性を使っている。エラーモデルは有限状態トランスデューサを用いて実装されており、実行時の速度を考慮して、辞書と文字変換ルール[3]から、頻出する文字列に関して変換候補を準備しておく。実行時には、ビームサーチデコーダを使用して、エラーモデル、文字ベースの言語モデル、付与したアクセント数素性を考慮した線形モデルに基づいてNベスト解を出力し、さらに単語ベースの言語モデルで再ランキングして候補を出力している。モデルとトレーニングの詳細はChoudhuryら[5]に譲るが、このようなアクセント付与タスクは、SとTのペアデータ作成が容易であるため（SはTから単純にアクセントを取り除いたものである）、モデルの構築と評価が容易にできる。Europarlの議事録[4]と、オンライン機械翻訳システムのユーザログでこのタスクを評価したところ、不正解率は文字のレベルで0.2%ほどであり、言語モデルを使わないシステム（つまりエラーモデルのみを使用）でも不正解率は0.5%以下であった[5]。後者のモデルを使用した場合、一文字当たりの計算時間は0.6msであり、入力システムしての使用に十分耐えうるものである。

以上、フランス語は入力文字と出力文字のマッピングが非常に単純である例であるが、表記に非ローマ字を使用する言語ではこのマッピングが一筋縄ではいかないことが多い。Universal Text Inputは言語非依存のシステムを目指しているが、このマッピングの部分においてのみ

[3] フランス語の場合、文字変換ルールはa→$\grave{a}$のようにアクセント付与ルールのみだが、文字変換を扱う言語の場合には文字種変換ルールが用いられる。

[4] http://www.statmt.org/wmt09

[5] 我々の使用したトレーニングデータではアクセントのついた文字の頻度は全体の3.64%であったので、これがベースラインの不正解率にあたる。

言語ごとに特殊なケースを扱うことにならざるを得ない。たとえばヒンディー語にはschwa deletionとよばれる現象がある。これは発音されない(つまりローマ字入力でタイプされない)母音aが転写される現象である。

S: devanagari/devnagri
T: देवनागरी

の例では、*T*を一文字ずつ音写するとde.va.nā.ga.rī(文字の境界をピリオドで表示)となるが、実際の発音はdevnāgrīであり、したがってこれをローマ字で入力する場合devanagari/devnagri が両方考えられ(実際に両方使われている)、後者の場合には母音を挿入しなければならない。デーヴァーナーガリー文字は音節文字であり、基本的に一文字は日本語のかなのように子音と母音のコンビネーションを表すが、日本語と違ってヒンディーでは実際には閉音節(子音で終わる音節)があるため、発音と表記のミスマッチがおこるのである。また、デーヴァーナーガリーでは短母音・長母音の区別(例:*a* vs. *ā*, *i* vs. *ī*)があるがこの区別も発音上されないことが多く、ユーザは単に*a*,*i*とタイプするため、この点でも曖昧性解消が必要になる。ヒンディーでのローマ字入力のバリエーションはSowmyaら[11]に詳しい。現在、ローマ字入力によるヒンディーのデータをユーザーから取集し評価する実験が進行中である。

2.3 スペル訂正を考慮した文字列変換

スペル訂正と文字列変換を同時に考慮したモデルはノイズのある通信回路モデルの枠内で可能である。上述の

$$T^* = \operatorname{argmax} P(T|S) = \operatorname{argmax} P(T)P(S|T)$$

で、スペル訂正を考えない入力システムの場合、*P*(*S*|*T*)は入力と出力のペアが辞書にあれば1、なければ0である。したがって変換候補のランキングは言語モデルのみにより決定される。スペル訂正を考慮するには、このエラーモデルにスペル訂正操作を考慮すればよい。例えば日本語のローマ字入力を考えたとき、*P*(*T*|*S*)は次のように書き換えられる:

$$\begin{aligned} & P(T|S) \\ &= \sum_C P(T|C)P(C|S) \\ &= \sum_C P(C|T)P(T)P(C|S) \end{aligned}$$

ここで*C*は*S*をスペル訂正した後のローマ字文字列であり、たとえば「入力」と入力したいのに*nyuuryou*とタイプミスしてしまったとき、*S=nyuuryou*, *C=nyuuryoku*である。上の式で*P*(*C*|*T*)はスペルミスを考慮しない場合のかな漢字変換確率、*P*(*T*)は言語モデル確率にあたり、*P*(*C*|*S*)がスペル訂正モデル確率にあたる。このようなモデルによる、スペル訂正を包括したピンイン入力が提案されている。例えば、Chenら[3]では中国語の音節構造に基づいたスペル訂正モデル確率*P*(*C*|*S*)をユーザログから学習している(が、収集されたログに関しての詳細情報はない)。最近

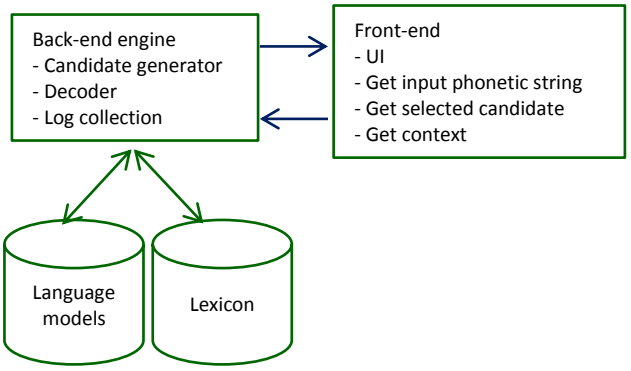

図2: ウェブサービスの概観

では、Zhengら[14]が、スペル訂正モデル確率にピンイン入力の誤変換ログから抽出した文字ベースの編集距離を人手で重みづけして使用したモデルを提案している。スペル訂正タスク自体、文字列変換タスクであるので、ノイズのある通信回路モデルや線形モデルの枠内で高性能の変換器を作ることができる。困難なのはモデル構築のためのペアデータの収集で、*S*も*C*もローマ字文字列であるために、通常のテキストとしては存在しない。そこで、Universal Text Inputではウェブサービスを通じてフィードバックデータを収集し、スペル訂正モデルの構築など、テキスト入力の改良に結びつけていくことをサービスの重要な機能として位置づけている。次節では、ウェブサービスとしてのインフラを活用して行ったデータ収集の具体例について述べる。

3 ウェブサービスを利用したデータ収集

図2はウェブサービスの概観である。サービスはクライアント側のユーザーインターフェースと、変換を行うサーバー側のバックエンドから成り立っている。ユーザーインターフェースがユーザーの入力をバックエンドに送り、候補のリストを受け取って表示する。この際、変換に付随する情報、たとえば変換した文字の周辺情報やユーザーが実際に変換した文字列なども送付することができる。サーバー側のバックエンドは、辞書と言語モデルは言語ごとに異なるが、変換エンジンは全言語共通である。

ウェブ上のアプリケーションであることの利点は、奥野ら[15]にもあるように大規模なデータソースを利用できること、サーバーサイドのハードウェアリソースが利用できること、新しい辞書や機能を素早くリリースすることできることなど様々あるが、Universal Text Inputという多言語・汎用目的のアプリケーションであることを念頭に置いたとき、インストールやダウンロードなしで使えるテキスト入力システムであることの利点が非常に重要である。このことにより、手間もかからず、ユーザーの設定に何の変更も加えることなく、入力への敷居を下げることができる。

またウェブサービスであることから、ユーザーフィードバックデータの収集も容易である。馬場ら[16]は、アマゾンのクラウドサービスAmazon Mechanical Turk (MTurk)を

図3: MTurk上でのHIT画面の例

使用し、Universal Text Inputを用いて二か国語（英語・日本語）でタイプミスのログを収集した。MTurkはHIT (Human Intelligence Task) と呼ばれるタスクを依頼者 (Requestor) がデザインし、アマゾンのクラウドインフラを介してユーザーがタスクを行い、完成したタスクにつき依頼者が代金を払う仕組みであり、NLPでも重要なデータ収集のインフラとして位置づけられるようになってきている[2]。馬場らでは、画像に説明文をつけるタスクと、画像中の人物あるいは動物にセリフをつけるという二種のタスクを作成し、それぞれについてユーザーのキーストロークを収集した。図3に画像説明タスクのHIT画面の一例をあげる。テキストボックス内では当該言語の入力がUniversal Text Inputサービスを呼びだすことでサポートされており、それ以外の方法では入力できなくなっている。このサービスを通じてユーザーのキーストロークが、Universal Text Inputのバックエンドのログ収集部に送られる。ログにはバックスペースキーを使用することで入力を訂正した跡が残っており、そこから訂正前後の文字列を抽出することが可能である。[6] ログの収集と分析はまだ始まったばかりであるが、将来的には継続的なモデル構築とアップデートに役立てていきたいと考えている。

4 おわりに

本稿では、多言語入力システム Universal Text Inputを紹介した。本プロジェクトは進行形のプロジェクトであり、現在は文字変換による入力、スペル訂正、ウェブサービスの活用の三点に焦点を当てているが、今後も新しい言語・機能ともに充実させていく予定である。

参考文献

[1] Brill, E., and R. C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL*.

[2] Callison-Burch, C., M. Dredze. 2010. Creating Speech and Language Data With Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

[3] Chen, Z., and K. F. Lee. 2000. A new statistical approach to Chinese Pinyin input. In *ACL*.

[4] Cherry, C. and H. Suzuki. 2009. Discriminative substring decoding for transliteration. In *EMNLP*.

[5] Choudhury, P., H. Suzuki and C. Quirk. 2011. From *pecher* to *pêcher*... or *pécher*: Simplifying French Input by Accent Prediction. In *Proceedings of the IJCNLP 2011 Workshop on Advances in Text Input Methods*.

[6] Cucerzan, S., and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*.

[7] Jones, M., and J. Martin. 1997. Contextual spelling correction using latent semantic analysis. In *ANLC*.

[8] Kernighan, M., K. Church, and W. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING*.

[9] Knight, K. and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

[10] Li, H., Kumaran, A., Zhang, M., Pervouchine, V. (2009). Report of NEWS 2009 Machine Transliteration Shared Task. In *Proceedings of ACL-IJCNLP 2009 Named Entities Workshop.*

[11] Sowmya V. B., M. Choudhury, K. Bali, T. Dasgupta and A. Basu. 2010. Resource Creation for Training and Testing of Transliteration Systems for Indian Languages. In *Proceedings of LREC*.

[12] Sun, X., J. Gao, D. Micol and C. Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of ACL*.

[13] Whitelaw, C., B. Hutchinson, G. Y. Chung, and G. Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *ACL*.

[14] Zheng, Y., C. Li and M. Sun. 2011b. CHIME: An efficient error-tolerant Chinese pinyin input method. In *Proceedings of IJCAI*.

[15] 奥野陽, 萩原将文. 2009. インターネットを用いた日本語入力システム. 情報処理学会研究報告自然言語処理（SIGNL-190）.

[16] 馬場雪乃, 鈴木久美. 2012. Amazon Mechanical Turkを利用したキーストロークログからのスペルミスの収集と分析. 言語処理学会第18回年次大会.

[17] 森信介、土屋雅稔、山地治、長尾真. 1998. 確率的モデルによる仮名漢字変換. 情報処理学会 自然言語処理研究会, NL-125.

[6] スペルミスのみならず、単に気が変わって別の文字列を入力したケースも含まれる。ログからのデータ収集の詳細については[16]を参照。