# 大規模候補リストを利用したトランスリタレーション

名古屋大学大学院工学研究科教授　佐藤　理史

PROFILE

京都大学大学院工学研究科電気工学第二専攻博士課程研究指導認定退学。博士（工学）。北陸先端科学技術大学院大学、京都大学を経て、2005 年 6 月より名古屋大学大学院工学研究科電子情報システム専攻教授。

## 1 はじめに

トランスリタレーション (transliteration：翻字：字訳：音訳) とは、“Smith”を「スミス」に訳すように、意味ではなく音 ( 発音 ) に基づいて語を翻訳することを指す。英日翻訳では、人名・地名・会社名・商品名などの固有名詞は翻字され、カタカナ表記されるのが普通である。それらに加え、専門用語も翻字される傾向が高まってきているように見受けられる。

手元にある、日本語で書かれた科学技術論文約 68 万本のメタデータを調べてみると、データに含まれる約 16 万種類のキーワードのうち、カタカナのみで書かれたキーワードは全体の 10.6%、一部にカタカナを含むキーワードは全体の 24.6% であり、これらを合わせると全体の約 1/3 に達する。このメタデータは、必ずしもすべての分野をカバーしているわけではないが、テクニカルな分野のドキュメントを翻訳する際、トランスリタレーションの問題は避けて通れないことは間違いない。

我々のグループは、2004 年頃から外国人名を主な対象としてトランスリタレーションの研究を行ってきた。昨年の Japio Year Book では、その技術を利用した外国人名対訳辞書の自動編纂について報告した。本稿では、新たに開発した、大規模候補リストを利用したトランスリタレーションの方式について報告する。

## 2 発想の転換

トランスリタレーションの研究は非常に盛んとは言いがたいが、毎年、かならず新しい論文が出るような分野である。我々が対象とする、英日方向のトランスリタレーション、および、日英方向の逆トランスリタレーション ( カタカナ綴からの原綴の推定 ) においても、これまで各種の方法が提案されてきた。しかしながら、現時点において、確立された方式・システムは存在しない。一見、問題はそれほど難しくは見えないのに、これはいったいどうしたことであろうか。

### 2.1 文字のマッピングによるトランスリタレーション

現在、トランスリタレーションの機械化の主流となっている方式は、実例 ( すなわち、「Smith →スミス」のような翻訳例 ) を大量に集め、機械学習の手法を適用して文字間の対応関係を推定してシステムを構成するという方式である。音をどのくらい明示的に扱うか、どんな学習方式を採用するかなど、多くのバリエーションがあるが、それらを捨象して単純化すれば、その基本は、文字のマッピングによる翻訳である。たとえば、“Smith”から「スミス」を得るためには、「s →ス」、「mi →ミ」、「th →ス」というマッピングを適用すればよい。もちろん、文字に対して音が一意に定まるとは限らないので、競合するマッピングのうち、どのマッピングが起こりやすいかということも知っておく必要があ

る。すなわち、機械学習では、実際に起こり得るマッピングの種類と、そのそれぞれの起こりやすさを学習する。

しかしながら、現実の翻訳例には、次のような現象が見られる。

(1) Michael Jordan → マイケル・ジョーダン
(2) Michael Connelly → マイクル・コナリー
(3) Michael Ende → ミヒャエル・エンデ
(4) Michael Schumacher → ミハエル・シューマッハ

これらの人名のファーストネームは、すべて“Michael”であるが、標準的に用いられる訳はすべて異なる。これらの標準訳のみを正解とするのであれば、上記の方式で高い精度のトランスリタレーション・システムを構成するのは、至難の技である。

## 2.2 既訳を探す

では、文字のマッピング以外に、どのような考え方があるであろうか。

実は、人間の翻訳者が行なう翻訳には、2つのケースが考えられる。以下では、“Barack Obama”の翻訳を例にして、この2つのケースについて説明しよう。

**ケース1**：“Barack Obama”の既訳が存在しない場合

歴史的に考えれば、“Barack Obama”を最初に日本語に訳した人が必ず存在する。彼女は、そのとき、どこを探しても既訳は見つからず、どこかの時点で既訳を見つけることをあきらめ、その日本語訳を作ったのである。既訳が存在しないのであれば、“Barack Obama”の訳として、「バラック・オバマ」を当てようが、「バラク・オバマ」を当てようが、大した違いはない。もちろん、“Barack ～”や“～ Obama”がどう訳されているかを参考にはしただろうが、これらの人とは別人なのであるから、どのような訳を当てるかは、彼女の自由である。

**ケース2**：“Barack Obama”の既訳が存在する場合

ケース1の場合を除けば、理論的には既訳が存在する。既訳が存在する場合、原則としてそれに従うというのが翻訳者の基本戦略である。その背景には、日本語しか読まない読者の存在がある。彼らにとっては、カタカナ表記の日本語訳が、“Barack Obama”という人物を特定する唯一の言語的手がかりとなる。そのため、「バラック・オバマ」と「バラク・オバマ」のように異なる表記は、最悪の場合、別人とみなされる可能性がある。

特に、既訳が標準訳として定着している場合は、それ以外の表記を採用するのは、ほとんど誤訳である。たとえば、英日翻訳において、ローマ法王の“John Paul II”に、「ジョン・ポール2世」という訳を当てるのは明らかに誤訳である。必ず「ヨハネ・パウロ2世」と訳さなければならない。

以上をまとめると、次のようになる。

(a) 翻訳者が知りたいのは、「既訳・標準訳が存在するか」ということである。
(b) 既訳・標準訳が存在する場合、原則として、新たな訳は作り出さない。つまり、翻訳者自身は、音に基づく翻訳を実行しない。
(c) 既訳が存在しない(あるいは見つからない)場合のみ、翻訳者自身が音に基づく翻訳を実行する。

これまでのトランスリタレーションの機械化の方式は、あたかも、それが初めて訳されるもの(上記のケース1)として、文字の対応関係(マッピング)に基づいて日本語のカタカナ表記を決めようとしていたことに相当する。これに対して、「既訳が全く存在しないということは、頻繁には起こらない」と考え、ケース1を除外してみよう。そうすると、トランスリタレーションは、もはや、文字をマッピングするという問題ではなく、その実体は、既訳を探すという探索問題となるのである。

## 2.3 大規模候補リストを用いる

実は、昨年度報告した外国人名対訳辞書の自動編纂でも、「既訳を探す」という考え方が採用されていたが、それは、サーチエンジンを使うという方法で実装されて

いた。具体的には、“Barack Obama”の既訳を探す場合、まず、これをサーチエンジンで検索し、得られたスニペットの中に現れるすべてのカタカナ語を抽出し、その中から“Barack Obama”のカタカナ訳として可能性があるもののみを残し、最終的に、スニペット中の出現回数を考慮して有望なもののみを出力する。(複数の訳が出力されることもある。)

これに対して、あらかじめ大規模な候補リストを用意しておき、その中から選ぶということはできないか、と考えたのが、新しい方式への出発点となった。たとえば、外国人名の英日翻訳では、大規模日本語コーパスから抽出したすべてのカタカナ語を候補とする。一方、外国人名の日英翻訳では、大規模な英語人名リストを候補リストとする。それらのリストのサイズは、百万件を越えることになるが、その中から少数の有望な候補を現実的な時間内で選択することができれば、いちいちサーチエンジンを引く必要がなくなる。それを実現する枠組およびアルゴリズムが、**非生産型トランスリタレーション**(Non-Productive Machine Transliteration; NPMT)である

## 3 非生産型トランスリタレーション

ここでは、非生産型トランスリタレーション(NPMT)の概要を簡単に説明する。技術的詳細は、文献[1,2]を参照されたい。

いま、翻訳(翻字)したいソース文字列 $s$ と、訳語の大規模候補リスト $T$ が与えられているとする。NPMTが解くべき問題は、「$s$ の翻訳として有望なものを $T$ の中から選ぶ」という問題である。これは、形式的には選択問題となるので、次のような方法で解くことができる。

(1) $s$ と $t$ ($\in T$) の組に対してコストを定義する。このコストは、翻訳対として有望なものほど小さくなるように設計する。

(2) すべての可能な組に対してコストを計算し、最も小さなコストをとる組のターゲット側 $t$ を出力する。(一つに定まるとは限らない。)

NPMTの基本的な骨格はこれだけであり、後は、これをいかにして高速に計算するかである。

候補リストのサイズが100万件以上だとすると、すべての可能な組(この数は、候補リストのサイズと等しい)に対してコストを計算するのは、かなり厳しい。そこで、コストが $c$ 以下の解だけを探すことにし、かつ、このコストの上限値 $c$ を、0から順次増やしていく方法を採用する。これを**反復型コスト束縛探索**と呼ぶ。

これを実現するために、コスト0の部分対応(文字のマッピング)をあらかじめ定めておく。たとえば、「mi →ミ」、「mi →マイ」などの典型的な対応をコスト0の対応として設定する。

コスト0の解は、コスト0の部分対応だけでカバーされる翻字対である。これを求めるために、まず、与えられた入力 $s$ の前方から、部分対応を利用して、考えられる出力の前方部分を作っていく。たとえば、“Michael Ende”が入力の場合、前方の2文字“mi”に対して、「ミ」や「マイ」などの候補が得られる。このとき、得られたそれぞれの候補が $T$ のいずれかの要素の前方文字列となっているかどうかを調べ、そうなっていなければ、その候補を削除する(なぜならば、その先を続けても、得られる出力側の文字列が $T$ の要素となることはないからである)。これを**プレフィックス・フィルタリング**と呼ぶ。このフィルタリングは、理論的にも実際的にも高速に実行できるので、無駄な探索を早めにカットすることができる。

上記のような処理をソース文字列 $s$ の前方から少しずつ延ばしていくと、その過程で、コスト0の解となり得ない候補は淘汰され、コスト0の解となり得るものだけが、少しずつ延びていく。たとえば、“Michael”まで進んだところでは、「マイケル」「マイクル」「ミハエル」「ミヒャエル」という候補はいずれも残るが、“Michael Ende”まで進むと、リスト $T$ に含まれないもの(「マイケル・エンデ」や「マイクル・エンデ」)は淘汰され、$T$ に含まれる「ミヒャエル・エンデ」や「ミ

ハエル・エンデ」だけが残ることになる。(「ミハエル・エンデ」という訳は少数ながら存在する。)

人名の発音は、かならずしも標準的な発音規則に従うとは限らない。言い換えるならば、コスト 0 の部分対応でカバーできるとは限らない。コスト 0 の解が見つからなかった場合は、コスト 1 の解、すなわち、ソース側またはターゲット側で 1 文字だけ対応しない文字が存在することを許すという条件下で、上記の処理を実行する。コストの上限値 $c$ を上げると、探索空間は急速に拡大して計算時間が長くなるので、現在の実装では、$c$=1 または 2 が現実的な上限となっている。

**表 1** に、実際に作成した英日・日英 NMPT システムの、外国人名の英日方向の実行例を示す。本システムでは、日本語側の文字として、カタカナではなく、新たに設計したローマ字を採用している。この実行例は、小さな候補リスト (58,083 件 ) に対して実行したものであるが、150 万件の候補リストを用いた日英方向の実験でも、コスト 0 の解を見つけるのに必要な時間は平均 140msec であり、十分に実用的な速度で動作することが確認されている。

## 4 おわりに

本稿では、非生産型トランスリタレーションと呼ぶ、新しいトランスリタレーションの方式について述べた。この方式は、訳語を大規模な候補リストの中から選ぶという点に特徴がある。本稿では外国人名に対する適用例を述べたが、この方式は、それ以外の一般のトランスリタレーションにもそのまま適用可能である。

これまで、我々は外国人名を中心にトランスリタレーションの研究を行なってきたが、現在、これに加え、他のクラスの固有名詞や専門用語も対象に含めることを検討している。同時に、訳語を大規模な候補リストの中から選ぶという考えを、トランスリタレーションではなく、一般のトランスレーション ( 翻訳 ) に適用することも試みている [3]。これらの研究の一つのゴールとして、我々は、「英語ウィキペディアを日本語で引く機能を実現する」ことを掲げている。すなわち、英語ウィキペディアの全見出し語 ( 現在、約 764 万件 ) を候補リストとして用いたトランスリタレーションおよびトランスレーションの実現を目指している。

[ 参考文献 ]

[1] 佐藤理史 . 大規模候補リストを利用したトランスリタレーション . 言語処理学会第 16 回年次大会論文集 , pp.900-903, 2010.

[2] Satoshi Sato. Non-Productive Machine Transliteration. Riao-2010 (9th international conference on Adaptivity, Personalization and Fusion of Heterogeneous Information), Paris, 2010.

[3] 岡田昌也 , 佐藤理史 . 大規模訳語候補集合を利用した専門用語翻訳 . 2010 年度人工知能学会全国大会 ( 第 24 回 ) 論文集 , 2C4-1, 2010.

| 入力 $s$ | コスト | ローマ字出力 $t$ | カタカナ出力 | 時間 (sec) |
|---|---|---|---|---|
| Edmund Weiss | 0 | etomu0to@vaisu | エトムント・ヴァイス | 0.0104 |
| Ludwig Thuille | 0 | ru=tovihi@tuire | ルートヴィヒ・トゥイレ | 0.0378 |
| Javier Vázquez | 0 | habia=@basukesu | ハビアー・バスケス | 0.0640 |
| Leonid Brezhnev | 1 | reoni=do@bure3inefu | レオニード・ブレジネフ | 0.1821 |
| Jean-Jacques Nattiez | 1 | 3a0-3a!ku@natie | ジャン=ジャック・ナティエ | 0.5858 |
| Amitabh Bachchan | 2 | amita=bu@ba!ca0 | アミターブ・バッチャン | 2.6999 |
| Pavlo Skoropadskyi | 3 | pa=veru@sukoropa=2ukii | パーヴェル・スコロパーツキイ | 9.3214 |
| Miguel Bernal Jiménez | | ( 正解を出力できず ) | ( ミゲル・ベルナル=ヒメネス ) | 14.8818 |

表 1　実行例 ( 英日方向 )