2C4-1

大規模訳語候補集合を利用した専門用語翻訳

Non-Productive Machine Translation of Technical Terms

岡田昌也　佐藤理史
Masaya Okada　Satoshi Sato

名古屋大学大学院工学研究科電子情報システム専攻
Department of Electrical Engineering and Computer Science, Graduate School of Engineering, Nagoya University

This paper proposes a non-productive machine translation framework of Japanese-English technical terms. The non-productive framework was first proposed as non-productive machine transliteration and we adapt it to machine translation of compound terms. In the non-productive framework, it is assumed that a large candidate list including the correct translation is given. Therefore, the translation problem is simplified into the selection problem of the correct entry from the large candidate list. We have developed an efficient algorithm that solves this selection problem. An experimental result shows that our algorithm is fast even if the size of the candidate list is over five millions.

1. はじめに

英語が国際語として定着した現在、英語で記述された情報の量は、他の言語で記述された情報を圧倒的に凌駕している。このため、英語非母国語話者は、情報収集という点において、英語母国語話者に対して大きなハンディキャップを負っており、それを軽減する機能の実現が強く求められている。その有力な方法の一つに、情報の記述に用いられている言語以外の言語で、その情報へのアクセスを可能とする言語横断検索がある。この方法は、情報を見つけ出すという作業を支援するものである。

言語横断検索を実現する標準的な方法は、検索クエリ（検索キーワード）を翻訳するという方法である。たとえば、日本語で英語文書を検索する場合、システムは、まず入力された日本語クエリを英語に翻訳し、その翻訳結果で英語文書を検索する。当然のことながら、その性能は翻訳精度に強く依存することになるが、単に「日本語クエリを英語に翻訳する」という一般的な問題設定では、それほど高い翻訳精度は望めない。

ここで、「検索の対象となる情報源には、あらかじめ検索索引リストが公開されている」という条件を仮定しよう。このような条件下では、**検索クエリの翻訳問題は、検索索引リストの中から適切なものを選ぶという選択問題に帰着できる**（それ以外の訳では情報源を検索できないので、そのような訳を考える必要はない）。翻訳問題を選択問題に帰着して解くというアイディアは、翻訳のサブクラスであるトランスリタレーションにおいて、すでに**非生産型トランスリタレーション**として提案されている [佐藤 10]。

本論文では、このような非生産的な枠組を、専門用語の翻訳に適用する。具体的には、複合語の翻訳に対する非生産型機械翻訳の枠組と、それを実行する効率的なアルゴリズムを示す。

2. 複合語の非生産型機械翻訳

2.1 コンセプト

非生産的な枠組の基本的なコンセプトは、「『すでに誰かによって過去に訳されたことがある』ことを仮定し、それを見つけることに専念する」というものである。より具体的には、正解を含む巨大な候補リストが与えられることを仮定する。

このような枠組を採用する一つの理由は、そのような候補リストが現実に入手可能となってきたからである。たとえば、日本語の専門用語の大多数は、英語の専門用語を翻訳したものである。すなわち、実在する英語の専門用語を網羅的に収集すれば、日本語のほとんどの専門用語の正しい英訳 (原語) は、その中に含まれていることが期待できる。

もう 1 つの理由は、このような枠組が「存在しない用語を出力しない」ことを保証する点にある。複合語の翻訳に広く用いられる要素合成法の 1 つの問題は、実際には存在しない用語を多数出力するという点にある。非生産的枠組は、この問題を回避することができる。

2.2 枠組

非生産型トランスリタレーション [佐藤 10] にならい、複合語の非生産型機械翻訳の枠組を次のように定義する。

与えられるもの

1. 対訳辞書 D
2. ソース側の文字列 (または単語列) x
3. ターゲット側の候補リスト T

求めるもの

$$\dot{\delta}_{min}(x, T, \dot{D}) = \dot{\delta}_{min} = \operatorname{argmin} \operatorname{cost}(\dot{\delta})$$
$$\text{satisfying } \dot{\delta} \in \dot{D}^*,\ \operatorname{src}(\dot{\delta}) = x,\ \operatorname{tgt}(\dot{\delta}) \in T$$

出力

1. $y = \operatorname{tgt}(\dot{\delta}_{min}) \in T$
2. $c_{min} = \operatorname{cost}(\dot{\delta}_{min})$

ソース側の言語およびターゲット側の言語において、何をプリミティブとみなすかは、それぞれの言語に応じて定めるものとする。たとえば、日本語では文字をプリミティブとみなし、日本語の複合語は文字列とみなす。これに対して、英語では単語をプリミティブとみなし、英語の複合語は単語列とみなす。

対訳辞書 D は、2 言語間の訳語対の集合である。たとえば、日英対訳辞書では、その要素である訳語対 d は、

連絡先: 岡田昌也，名古屋大学大学院工学研究科電子情報システム専攻，名古屋市千種区不老町 C3–1(631), okada@sslab.nuee.nagoya-u.ac.jp

⟨ 機械翻訳, machine translation⟩ のように、日本語の文字列と英語の単語列のペアとなる。

本枠組のアイディアの中核は、訳語対 d の列 $\delta(\in D^*)$ である。δ は訳語対の列であると同時に、複合語全体の訳語対、および、その部分対応 (アライメント) を表現している。たとえば、

$$\begin{aligned} d_1 &= \langle \text{機械翻訳}, \text{machine translation} \rangle \\ d_2 &= \langle \text{システム}, \text{system} \rangle \end{aligned}$$

とすれば、訳語対の列 $\delta = d_1 d_2$ は、

$$\begin{aligned} \delta &= d_1 d_2 \\ &= \langle \text{機械翻訳}, \text{machine translation} \rangle \langle \text{システム}, \text{system} \rangle \\ &= \langle \text{機械翻訳システム}, \text{machine translation system} \rangle \end{aligned}$$

という 3 つの情報を同時に表している。このことから、D^* は、D によって生成可能な複合語の訳語対の集合を表していることがわかる。なお、以下では、$\mathrm{src}(\delta)$ と $\mathrm{tgt}(\delta)$ は、それぞれ、複合語の訳語対のソース側、ターゲット側を表すものとする[*1]。

$\dot{D}$ は拡張された対訳辞書 (後述する) を表し、$\dot{\delta}(\in \dot{D}^*)$ は、その要素の列を表す。$\dot{\delta}$ に対して、そのアライメントの善し悪しを反映したコストを定義する。この枠組は、与えられた入力 x に対し、2 つの条件

$$\text{(ソース側条件)} \qquad \mathrm{src}(\dot{\delta}) = x \tag{1}$$

$$\text{(ターゲット側条件)} \qquad \mathrm{tgt}(\dot{\delta}) \in T \tag{2}$$

を満たすアライメント $\dot{\delta}$ のうち、最小コストをとるもの $(\dot{\delta}_{min})$ を見つけるという枠組であり、そのターゲット側 $y = \mathrm{tgt}(\dot{\delta}_{min})$ が、求める翻訳となる。

2.3 単純なアルゴリズム

ここでは、まず、対訳辞書 D の拡張を考えず、D が生成可能な複合語の訳語対のみを対象としよう。このとき、上記の 2 つの条件を満たす δ は次のように求めることができる。

まず、ソース側条件を満たす δ を求める。これは、次の方法を、x のあらゆる可能な分割に対して適用することで求まる (要素合成法と呼ばれる方法と同一である)。

1. 入力 x をいくつかの部分に分割する。分割結果を $x_1 x_2 \cdots x_n$ とする。
2. 各 x_i に対して、$\mathrm{src}(d_i) = x_i$ を満たす $d_i(\in D)$ を得る。
3. $\delta = d_1 d_2 \cdots d_n$ である。

ステップ 2 で、ある x_i に対して複数の d_i が存在した場合は、ステップ 3 ですべての組合せを作成する。逆に、ある x_i に対して、一つも d_i が見つからなかった場合は、δ は一つも求まらないことになる。

次に、こうして得られた δ の中から、ターゲット側条件を満たすものだけを抜き出す。以上により、ソース側条件とターゲット側条件の両者を満たす δ の集合が得られる。

2.4 動的計画法とプレフィックス・フィルタリング

実際に使用するアルゴリズムは、上記の説明とは異なり、動的計画法 (Dynamic Programming, DP) とプレフィックス・フィルタリングを用いたアルゴリズム [佐藤 10] である。

プレフィックス・フィルタリングは、ターゲット側条件のチェックを早期に行ない、探索空間の枝刈りを行なう手法である。ターゲット側条件は、解となるアライメント δ が、条件 $\mathrm{tgt}(\delta) \in T$ を満たすことを要請する。いま、$\delta = d_1 d_2 \cdots d_n$ とするとき、もしその δ が解であるならば、そのプレフィックス (前方部分列) $d_1 d_2 \cdots d_i (i < n)$ は、T のいずれかの要素 t のプレフィックスと必ず一致する。逆に、$d_1 d_2 \cdots d_i (i < n)$ と一致するプレフィクスを持つ $t(\in T)$ が存在しなければ、その後ろにどんな $d(\in D)$ を付加しても、解とはなり得ない。そこで、δ を前から成長させていく過程で、随時、そのターゲット側が T のいずれかの要素のプレフィックスと一致するかどうかを調べ、一致しない場合に、そこから先の探索を中止する。これがプレフィックス・フィルタリングである。

アルゴリズムを説明するために、次の記法を導入する。まず、x の $i+1$ 文字目から k 文字目までの部分文字列を、x_i^k と記述する。次に、x の長さ k のプレフィックス x_0^k に対して、次の条件を満たす訳語対列 $\zeta_k(\in D^*)$ を考える。

$$\mathrm{src}(\zeta_k) = x_0^k \tag{3}$$

先に述べたように、この ζ_k の後ろにいくつかの $d(\in D)$ を連結したものが解となるためには、ζ_k は次の条件を満たさなければならない。

$$\mathrm{tgt}(\zeta_k) \in \mathrm{prefix}(T) \tag{4}$$

ここで、$\mathrm{prefix}(T)$ は、T のすべての要素のすべてのプレフィックスからなる集合を表すものとする。

採用するアルゴリズムは、このような ζ_k を $k = 0, 1, \cdots, |x|$ の順に求めていくというものである。ここで、同じ計算を 1 回しか行なわないようにするために、動的計画法を用いる。ζ_k は、以下の式によって計算される (ここでは、プレフィックス・フィルタリングを陽に記述していないが、ζ_k は、すべて、式 (4) の条件を満たす必要がある)。

$$\zeta_0 = \epsilon \tag{5}$$

$$\zeta_k = \zeta_i + d \quad \text{satisfying } \mathrm{src}(d) = x_i^k,\ 0 \le i \le k-1 \tag{6}$$

ここで、ϵ は空列を表す。動的計画法を用いれば、式 (6) の右辺第 1 項は、すでに行なわれた計算によって求められている。右辺第 2 項の d は、対訳辞書 D に登録されている対訳対のうち、ソース側が x_i^k となっているものを表す。“+” の記号は、訳語対の連結を意味する。

こうして得られた $\zeta_{|x|}$ のうち、$\zeta_{|x|} \in T$ を満たすもののみを抜き出せば、求める解集合が得られる。

3. 日英翻訳への適用

前節の枠組とアルゴリズムは、任意の言語対に対して適用可能である。その一方で、辞書の拡張については何も規定していない。ここでは、この枠組を複合語の日英翻訳に適用する場合の、辞書の拡張法について述べる。

3.1 対訳辞書の拡張

一般に、完全な (網羅的な) 対訳辞書を作成することは、非常に困難である。そこで、対訳辞書の不備を補うために、与えられた辞書 D に対して、それを拡張した辞書 $\dot{D}$ を考える。以下の説明では、話の都合上、与えられた対訳辞書を D_0、拡張辞書を $D_i (i > 0)$ と表記する。

拡張 1 付属語「の」の類を削除した訳語対の追加

「格子間領域」の正解訳は、“interstitial region” である。ここで、D_0 には、訳語対 ⟨ 格子間の, interstitial⟩ と、

*1 これらの関数は、必要に応じて区切り記号 (スペース等) の挿入を行なうものとする

⟨ 領域, region⟩ しかないとすると、「格子間領域」から “interstitial region” を生成することができない。

上記の現象は、日本語の複合語において、「の」のような付属語が削除されうることに起因する。このような付属語として、次のようなものがある。

$$F_A = \{\ の, 的, する, 的な, な, 性の, 性, \} \tag{7}$$

そこで、このような付属語を末尾に持つ訳語対 $d(\in D_0)$ に対して、それらの付属語を削除した訳語対を仮想的に追加する。これは、次のように辞書を拡張することに相当する。

$$D_1 = D_0 \ \cup\ \{\langle \alpha, e\rangle \mid \langle \alpha a, e\rangle \in D_0, a \in F_A\} \tag{8}$$

ここで、α は任意の日本語文字を表す。

拡張 2 接尾辞「性」の類に対する削除可能規則の追加

「境界性人格」の正解訳は、“borderline personality” である。ここで、D_0 には、訳語対 ⟨ 境界, borderline⟩ と、⟨ 人格, personality⟩ しかないとすると、「境界性人格」から “borderline personality” を生成することはできない。

このような現象は、英語の名詞連続において、日本語では前方の名詞がしばしば形容詞的に訳されることによって起こる。

そこで、名詞を形容詞化するような接尾辞に対し、削除可能であることを表す訳語対を仮想的に追加する。たとえば、「性」に対して、訳語対 ⟨ 性, ϵ⟩ を追加する。このような付属語として、次のようなものがある。

$$F_B = \{\ 性, 症, の, 術, 法, 型, 的, 化\ \} \tag{9}$$

辞書の拡張は、次のように表現できる。

$$D_2 = D_0 \ \cup\ \{\langle b, \epsilon\rangle \mid b \in F_B\} \tag{10}$$

拡張 3 拡張 1 と拡張 2 の同時拡張

上記の 2 つの拡張を同時に行なうことも考えられる。すなわち、

$$D_3 = D_1 \ \cup\ D_2 \tag{11}$$

これらの拡張は、いずれも対訳辞書に新たな訳語対を追加することに相当するため、それらが生成しうる複合語の訳語対の集合 D_i^* は D_0^* より拡大する。別の言い方をすれば、より多くの複合語が翻訳可能となる。しかし、その一方で、正解訳以外の解を出力してしまう危険性も高くなる。

そこで、使用する辞書 D_i に優先順位を定義し、優先順位の高いものから順に使用することにする。優先順位は、次のように定義する。

$$D_0 > D_1 > D_2 > D_3 \tag{12}$$

すなわち、まず、D_0 を用いて翻訳を実行し、結果が 1 つも得られなかった場合のみ、D_1 を用いた翻訳を試みる、というように進める。対訳辞書 D_i を用いた場合に得られた解をコスト i の解と定義すれば、2.2 節に示した枠組にうまく収まる。

4. 実験と検討

本節では、既知の日英専門用語対 (テストペア) を用い、2 節および 3 節で提案した方法を実験的に評価する。日英という方向を選んだのは、1 節で述べたように、本手法が想定している応用が日英の言語横断検索の実現であるからである。まず、本方法の実行に必要な対訳辞書 D と、候補リスト T の作成、および、実験に用いるテストペアの収集について述べる。次に、これらを用いた実験とその結果について述べる。

表 1: 対訳辞書 D_0 の作成

	訳語対の数
ステップ (1) 終了時	1,623,809
部分対応辞書	16,897
対訳辞書 D_0	1,633,359

4.1 対訳辞書の作成

本実験で使用する対訳辞書の作成には、英和辞書『英辞郎』[*2] ver.79 を利用した。『英辞郎』は、多くの複合語の訳語対を収録している。しかし、その一方で、複合語の部分対応を表す訳語対は、かならずしも収録されているとは限らない。そこで、このような収録されていない訳語対を取り出して部分対応辞書 ([外池 07], [藤井 00]) を作成し、これを対訳辞書に追加する。

以下に、具体的な部分対応辞書の作成手順を示す。

(1) **エントリを展開する**

一般に、『英辞郎』では、1 つの見出し語に対して複数の訳語が定義されている。ここでは、これらを展開し、1 つの見出し語に対して 1 つの訳語を対応させた訳語対を生成する。以降、これを、$\langle j, e\rangle$ と記述する。j は日本語、e は英語を表すものとする。

(2) **部分対応数 2 の訳語対を抽出する**

生成した訳語対の集合のうち、j の形態素数と、e の単語数が、ともに 2 である訳語対のみを抽出する。ここで、日本語側の形態素数は、日本語側の文字列 j を、形態素解析器 MeCab [*3] で形態素解析することで求める。

(3) **複合語の部分対応を表す訳語対を生成する**

抽出した訳語対のそれぞれに対して、j の 1 番目の形態素と e の 1 番目の単語、j の 2 番目の形態素と e の 2 番目の単語をそれぞれ対応させ、訳語対を 2 つ生成する。

(4) **多数生成された訳語対を抽出する**

こうして生成された訳語対のうち、同一の訳語対が k 個以上生成されたものを、部分対応辞書に登録する。

このような手順で作成した部分対応辞書の訳語対の数を表 1 に示す。ここでは、上記の (4) の k として、5 を用いた。

最終的に、上記の手順で作成した部分対応辞書に、ステップ (1) で作成した訳語対を合わせ、対訳辞書 D_0 を作成した。表 1 において、D_0 の訳語対の数が、ステップ (1) 終了時と部分対応辞書の和より少ないのは、重複する訳語対が存在するためである。

4.2 候補リストの作成

本実験では、ターゲット側の候補リスト T として、Wikipedia 英語版[*4] の見出し語リスト (5,457,971 件) を用いた。

4.3 テストペアの収集

テストペアの収集には、新旧の『英辞郎』の差分を利用した。まず、新しい版である『英辞郎』ver.116 から、対訳辞書 D_0 の作成に利用した『英辞郎』ver.79 には含まれない専門用語の訳語対を抽出した。次に、抽出した訳語対 $\langle s, t\rangle$ のうち、次の条件を満たすもののみをテストペアとして採用した。

*3 http://mecab.sourceforge.net/
*4 http://en.wikipedia.org/wiki/

表 2: 再現率と精度

i	総出力数 (a+b)	正解 (a)	不正解 (b)	再現率 (%) ($\frac{a}{647}$)	精度 (%) ($\frac{a}{a+b}$)
0	342	214	128	33.1	62.6
1	392	252	140	38.9	64.3
2	486	296	190	45.7	60.9
3	488	298	190	46.1	61.1

表 3: 得られた結果の分類

i	Perfect	Ambiguous	False	None
0	156	58	27	0
1	34	4	7	0
2	31	13	9	0
3	2	0	0	0
—				306
Total	223 (34.5%)	75 (11.6%)	43 (6.6%)	306 (47.3%)

条件 1 t は、作成した候補リスト T に含まれる
条件 2 t は、2 単語以上で構成されている
条件 3 s は、漢字と平仮名のみから構成されている

条件 1 は、非生産型機械翻訳の仮定 (入力 s の正解訳 t は T に含まれる) に基づく。条件 2 は、対象を複合語と限定するための条件である。条件 3 は、カタカナ語を除外することを意味する[*5]。

最終的に採用したテストペアの数は、647 対である。

4.4 実験

2 節および 3 節で提案した方法を Ruby を用いて実装した。但し、プレフィクス・フィルタリングの実装には、sary[*6]を利用した。実験には、iMac(Core 2 Duo 2.16GHz) を用いた。

作成したプログラムにテストペアのソース側 (日本語側) を入力として与え、どのような出力が得られるかを調べた。647 件の入力に対して、総計で 488 件の出力が得られた。計算に要した時間は、総計で 16.84 秒、1 入力あたりの平均計算時間は 26.03ms であった。

テストペアのターゲット側を正解とした場合の再現率と精度を表 2 に示す。この表の i は解のコストを意味し、それぞれの行は、コスト i 以下の解に対する値を表す。

表 3 に、それぞれの入力に対してどのような出力が得られたかを示す。この表の記号は、以下のことを表す。

Perfect: 正解訳のみを出力
Ambiguous: 正解訳の他に、他の訳語 (不正解訳) を出力
False: 不正解訳のみを出力
None: 出力なし

また、i は解のコストを意味し、それぞれの行は、コスト i の解に対する値を示す。

*5 カタカナ語の日英翻訳は、非生産型トランスリタレーション [佐藤 10] で実現できるため、今回の実験からは除外した。

*6 http://sary.sourceforge.net/

4.5 検討

プログラムの実行速度は、1 つの複合語の翻訳に要する計算時間が平均 26.03ms であった。このことから、500 万件を越える候補リスト T に対して、本方式が実用的な速度で動作することが確かめられた。

その一方で、再現率および精度は不十分であった。表 2 より、対訳辞書の拡張により、再現率が 33.1%から 46.1% に向上していることが確認できる。すなわち、対訳辞書の拡張は有効に機能している。しかしながら、再現率は最大でも 46.1%であり、半数以上の入力に対して正しい訳を出力することはできなかった。これは、辞書の拡張によっても、その訳語対の不足を十分に補えなかったことを示している。再現率の改善には、対訳辞書 D_0 の、抜本的な強化が必要であると言える。

精度は、辞書の拡張につれて低下すると考えられる。しかしながら、辞書の拡張によって、再現率が 13.0%向上したの対し、精度は 1.5%しか低下しなかった。これは、現在の辞書の拡張が妥当であり、かつ、順次拡張する方法が有効に機能していることを示している。

表 3 からは、306 件 (47.3%) の入力に対して、出力が全く得られていないことがわかる。このことからも、今回用いた対訳辞書 D_0 のカバレジが、大幅に不足していることがわかる。

Ambiguous は、出力される訳の数が数個であれば、実用上大きな問題とはならない。なぜならば、我々が想定する応用では、これらの訳を Wiki の曖昧さ回避ページのような形でユーザーに提示し、その中から 1 つを選んでもらえばよいからである。これに対して、False は深刻なエラーである。対訳辞書の拡張によって、このようなエラーが発生してしまうことは避け難いが、本実験では、拡張しない場合 (D_0 を使用) においても、このようなエラーが少なからず発生している。この原因も対訳辞書 D_0 のカバレジが不足している点にあると考えられる。

5. おわりに

本論文では、非生産的な枠組を専門用語の日英翻訳に適用する方法について述べた。実験の結果、提案した方法は、500 万件以上の候補リストに対しても、実用的な速度で動作することが確認できた。しかしなら、再現率および精度は不十分であり、対訳辞書の強化が必要であることが明らかとなった。今後は、対訳辞書の強化と、対訳辞書のさらなる拡張方法の検討を行なっていく予定である。

謝辞 本研究は、科学研究費補助金挑戦的萌芽研究 (課題番号 22650047) の支援を受けている。

参考文献

[佐藤 10] 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 回年次大会発表論文集, pp. 900-903 (2010).

[外池 07] 外池昌嗣, 宇津呂武仁, 佐藤理史: ウェブから収集した専門分野コーパスと要素合成法を用いた専門用語訳語推定, 言語処理学会, Vol. 14, No. 2, pp. 33-68 (2007).

[藤井 00] 藤井敦, 石川徹也: 技術文章を対象とした言語横断検索のための複合語翻訳, 情報処理学会論文誌, Vol. 41, No. 4, pp. 1038-1045 (2000).