記号列間の類推方程式を解くアルゴリスムの研究

伊藤 舞子　金 敬訓　　ルパージュ イヴ
早稲田大学大学院 情報生産システム研究科
{wm003@akane, kei2407@ruri, yves.lepage@}.waseda.jp

1　はじめに

1.1　背景

以前、翻訳の考え方 [6] が提唱されて以来、用例翻訳システムの研究が数多く存在する。翻訳精度を向上させるために、類推関係を用いた翻訳システムとして、意味論的に類推を使用した語彙の関係 [7] や、単語構造の関係 [1]、語彙の構造を抽出 [2]、形態論の区分化 [3] する方法など挙げられる。しかし用例翻訳システムでは、ルールを追加することに比例し、データが大容量になる問題が発生する。

この問題に対して、本論文では翻訳速度の向上を目的とした研究を行う。入力文に対する類似文をデータベースから抽出する際、入力文と 2 つの類似文の文字数を考慮したソート手法と、文字の出現回数を考慮したエントロピーによるソート手法を提案する。ソートを行うことによりソートなし手法との速度を比較し、有効であるか検証する。また、以前研究された [4] 結果では複数解候補が存在する場合であっても、1 つのみが選択され出力される問題があるため、候補に挙がる複数解全てを出力とすることも本論文の目的とする。

1.2　類推方程式

類推関係とは，4 つのものの比例関係に基づいた概念であり、A と B の関係は C と D と等しい関係である。これを A:B::C:D と表記する。これらは「A が B であるならば C は D である」と意味する。以下に日本語の例を挙げる。「食べる : 食べます :: 決める : 決めます」や「私は猫が好きです : 私たちは猫を飼っています :: 私は犬が好きです : 私たちは犬を飼っています」。

類推関係を使用する際は、要素の 1 つが未知であり、類推方程式の「最初の要素は二番目の要素と同様に、3 番目の要素は 4 番目の要素と同様の方法で導出する」という定義に基づいて未知の要素を導出する。「食べる : 食べます :: 決める : x」や「私は猫が好きです : 私たちは猫を飼っています :: 私は犬が好きです : x」x は未知の要素であり、求める箇所となる。

記号列間の類似は、類推方程式を用いた用例翻訳システムの重要部分である。この方法は 1984 年長尾 [6] により導入され、2005 年ルパージュら [4] により洗練された。本研究に用いるアプローチは、以下のような例で示される、文と文の類推方程式を導出する 1 つの翻訳システムである。

$$\begin{array}{ccccccccc} \mathrm{A} & : & \mathrm{B} & :: & \mathrm{x} & : & \mathrm{D} \\ & & & & \downarrow & & \\ & & & & \mathrm{C} & & \\ \updownarrow & & \updownarrow & & \updownarrow & & \\ \mathrm{A'} & : & \mathrm{B'} & :: & \mathrm{C'} & : & \mathrm{y} \\ & & & & & & \downarrow \\ & & & & & & \mathrm{D'} \end{array}$$

(A,A′),(B,B′),(C,C′) は翻訳システムの例のデータベースに存在する。

2　類推方程式の制約

類推方程式を解く際、3 つの制約を設ける。まず、導出される解の文字列の長さを求める。次に、文字列の長さに対する使用される文字の出現回数を求める。最後に検出した出現回数の文字の位置を探索する。

2.1　文字列の長さ

文字列の長さは、類推関係にある 3 つの要素から導くことが可能である。3 つの要素を A,B,C と置き、導出される解を D とすると、以下の関係が成り立つ。

$$|\mathrm{A}| + |\mathrm{D}| = |\mathrm{B}| + |\mathrm{C}|$$
$$\Rightarrow |\mathrm{D}| = |\mathrm{B}| + |\mathrm{C}| - |\mathrm{A}|$$

文字列の長さの関係を可視化したものを図 1 に示す。

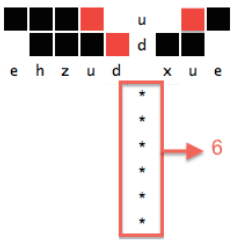

図 1: 文字列の長さ

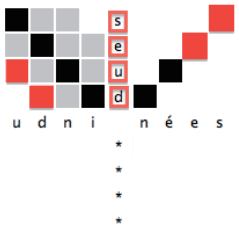

図 2: 文字の検索範囲

2.2 文字の出現回数

文字の出現頻度は第 2 章 1 節と同様、類推関係のある 3 つの要素から導くことが以下の関係から可能である。

$$|\mathrm{A}|_a + |\mathrm{D}|_a = |\mathrm{B}|_a + |\mathrm{C}|_a, \forall a$$

表 1 に一例の計算結果を示す。表内の数字は、各文字列内の文字の出現回数を表す。

表 1: 文字の出現頻度

	A	B	C	D	A+D	B+C
d	1	1	0	0	1	1
e	0	1	1	2	2	2
h	0	1	0	1	1	1
u	1	1	1	1	2	2
x	0	0	1	1	1	1
z	0	1	0	1	1	1

2.3 文字の位置

文字の出現頻度により導出された文字が文字の位置を示すには、3 つの要素の文字が関係している。3 つの要素 A,B,C は、A の文字が B と C の文字に含まれていることが最低条件となる。文字と文字の最短検索範囲として | 長い文字 | ー | 短い文字 | +1 が成り立つ。しかし、全ての類推関係が最短検索範囲を満たすとは限らないため、その際検索幅を拡張する必要がある。以下の図 2 に例を示す。

3 つの要素が $du : duzhe :: nées : D$ の場合、黒いセルで示している最短検索範囲内では、A の文字が全て存在していないため、灰色のセルで示している検索の幅を拡張し、A の文字を全て満たすことを行う。検索の幅を拡張する際、AB の文字間だけではなく AC の文字間も拡張するため、検索の幅が広がることにより文字の探索を行う計算時間も増加する問題がある。この問題に対し、今回最短検索範囲内で得られる幅においては拡張しないという方法をとる。4 つの要素のある類推関係から AB,AC,DB,DC の検索範囲を検証したところ、AB と DC は同じ検索幅であることが実験により証明され、また同様に AC と DB の関係も同じ検索幅であることが判明した。以下にいくつか実例を挙げる。

表 2: 検索範囲の双方の幅サイズ

$A : B :: C : D$	AB=DC	AC=DB
sprechen : *er spräche* :: *nehmen* : *er nähme*	1=1	1=1
fliehen : *er floh* :: *:schließen* : *er schloß*	3=3	1=1
huzila : *huzAl* :: *Sudi'a* : *SudA'*	0=0	0=0
aslama : *arsala* :: *muslim* : *mursil*	1=1	1=1
du : *duzhe* :: *xue* : *xuezhe*	0=0	0=0
dues : *indu* :: *nées* : *inné*	2=2	0=0
recevoir : *j'ai reçu* :: *percevoir* : *j'ai perçu*	0=0	1=1

今回証明されたこの方法を用いることで、探索時間を大幅に削ることは可能である。

2.3.1 文字数によるソート

検索範囲が決定し、文字の出現回数と検索範囲の適合を検証する。その際、検索範囲より文字数の少ない昇順にソートを行う。文字数が少なければ、文字の位置の決定も速くなると考えられるためである。実際に一例として $du : duzhe :: xue : xuezhe$ を挙げる。以

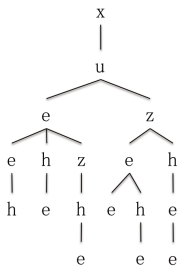

図 3: 文字数によるソート前

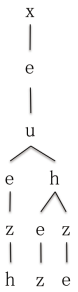

図 4: 文字数によるソート後

下は文字数によるソート前図 3 とソート後図 4 である。
文字数によるソート前は 17 分岐であるのに対し、ソート後は 10 分岐となる結果が得られる。

2.3.2 エントロピーによるソート

エントロピーによるソートも、 第 2 章 3 節 1 項の文字数によるソートと同様、検索範囲が決定し、文字の出現回数と検索範囲の適合を検証する。その際、検索範囲よりエントロピーの数値が低い昇順にソートを行う。エントロピーとは、乱雑さを意味する。乱雑である数値が低いものほど解に確証を得る確率が高いため、文字の位置の決定も高速であると考えられる。先ほどの例と同様の $du : duzhe :: xue : xuezhe$ を挙げる。以下の例はエントロピーによるソート後の結果である。右の数値は各文字がその文字の位置である確率のエントロピーを表す。

D[0] = { x:$\frac{1}{1}$ } 0.00
D[5] = { e:$\frac{2}{2}$ } 0.00
D[1] = { u:$\frac{2}{3}$, x:$\frac{1}{3}$ } 0.92
D[4] = { e:$\frac{2}{4}$, h:$\frac{1}{4}$, u:$\frac{1}{4}$ } 1.50
D[2] = { e:$\frac{1}{5}$, u:$\frac{2}{5}$, x:$\frac{1}{5}$, z:$\frac{1}{5}$ } 1.92
D[3] = { e:$\frac{1}{5}$, h:$\frac{1}{5}$, u:$\frac{1}{5}$, x:$\frac{1}{5}$, z:$\frac{1}{5}$} 2.32

3 実験

今回の実験では類似文字列から解を生成するまでの時間の計測を行った。その際、複数解生成された場合は複数解全て出力し、単数解の場合は 1 つのみを出力し、解が存在しない場合は解なしと出力するものとする。実験はソートなし手法とソートあり手法の速度比較、文字数に基づいたソート手法とエントロピーに基づいたソート手法の速度比較、類推方程式の長さによる速度比較実験を行った。今回実験に使用したデータは 3 種類のデータを用いた。データ 1 は実在している単語の例、データ 2 は単純な文字列で作成した形式的なデータの例、データ 3 は類推方程式として成り立たない例を用いた。

3.1 実験データ

データ 1 は実在している単語の例をデータを用いた。このデータは 12 カ国語 (ドイツ語、フランス語、アラビア語、中国語、日本語など) より 944 の類推関係のあるデータを生成した。またデータを生成する際、 1 つの類推関係から他の等しい 7 つの類推関係が生成できる。これは、中央要素を変更したり（ $A : B :: C : D$ の類推関係における B と C の要素の代替）、類似の対照（:: の印を基準とした両側の入れ換え）によって基本的な属性が含まれている。以下のようにして、次の 8 つの類推関係が表現出来る [5]：

$A : B :: C : D$ $C : A :: D : B$
$A : C :: B : D$ $C : D :: A : B$
$B : A :: D : C$ $D : B :: C : A$
$B : D :: A : C$ $D : C :: B : A$

この類推関係の性質を用い、118 のデータから他の 7 つの類推関係を生成する。以下に実例を示す。ドイツ語の例： $setze : setzte :: lachen : lachte$ 、日本語の例： 食べる : 食べます :: 認める : 認めます 、要素代替したもの： るべ食 : すまべ食 :: るめ認 : すまめ認
データ 2 は単純な文字列で生成した形式的なデータの例である。このデータも上記の類推関係の性質に基づき、39 のデータより生成したため 312 のデータを使用した。以下に実例を示す。 $a : aa :: aaa : aaaa$ 、$ab : aabb :: aaabbb : aaaabbbb$
データ 3 は類推方程式として成り立たない関係の 48 のデータである。以下に実例を示す。 $a : ab :: c : bc$ 、 $b : b :: ba : bb$

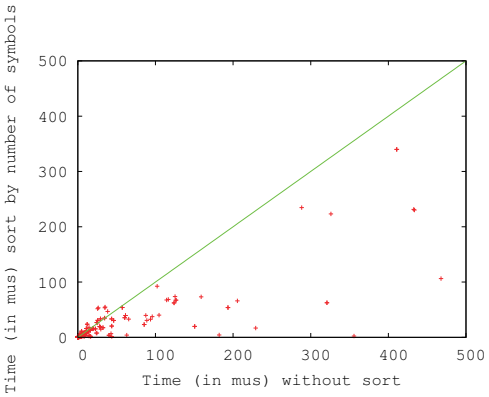

図 5: 文字数によるソート (データ 1)

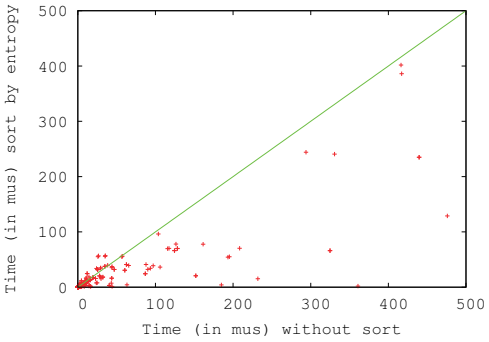

図 6: エントロピーによるソート (データ 1)

3.2 実験結果

ソートなしの速度とソートありの速度比較を行った結果を図 5 と図 6 に示す。図は対角線を境目とし、対角線より上に存在する印はソートなし手法の結果が良く、対角線より下に存在する印はソートある手法の結果が良いと判断されたものである。　今回はデータ 1 である実在している単語の例の結果のみを掲載しているが、データ 2、データ 3 もソートあり手法に印が多くみられる同様の傾向がみられた。

類推方程式の長さは A:B::C:?→ $|A|+|B|+|C|$ で示される、A,B,C の文字列の長さによる比較を行った。

図 7 に結果を示す。縦軸がデータ 1 を用いたエントロピーによるソートを行った時間、横軸が類推方程式の文字列の長さである。　文字列の長さ 20 を超えたあたりから時間が増加している傾向がみられる。これはアルファベットは 26 文字で成り立っているため、文字の出現頻度が高いものが影響していることを現している。

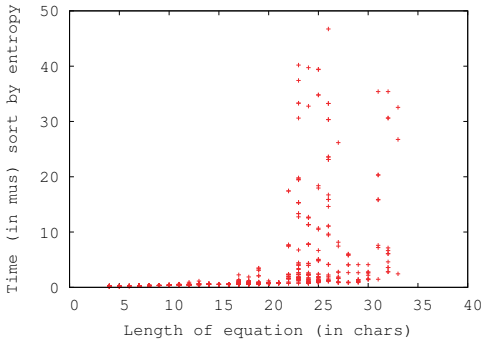

図 7: 類推方程式の長さ (データ 1)

4　おわりに

全体データの 50%以上のデータより速度向上がみられ、時間は平均的に 17%の時間削減であった。ソートあり手法においては文字数によるソート、エントロピーによるソートどちらが優れているか判断は出来かねない結果となった。考えられる原因として、より多くの実験データを生成し速度の比較を行うことを目的としたため、実験データの文字列の長さに偏りがあったと考えられる。今後の課題として、今回単語単位で類推関係が成り立っている例を用いたため、文字列の長い例を用いる必要があると考えられる。

参考文献

[1] Vincent Claveau and Marie Claude L'homme. Structuring terminology using analogy-based machine learning. 2005.

[2] Nabil Hathout. Acquisition of morphological families and derivational series from a machine readable dictionary. *The Computing Research Repository*, 2009.

[3] Jean-Francois Lavallee and Philippe Langlais. Unsupervised morphology acquisition by formal analogy. *In Lecture Notes in Computer Science*, 2010.

[4] Y. Lepage and E. Denoual. Purest ever example-based machine translation: detailed presentation and assessment. *Machine Translation*, (251-282), 2005.

[5] Yves Lepage. Analogy and formal languages. *Electronic notes in theoretical computer science*, pages 180–191, apr 2004.

[6] Makoto Nagao. A framework of a mechanical translation between japanese and english by analogy principle. *Artificial and Human Intelligence*, pages 173–180, 1984.

[7] Peter D. Turney and M.L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 2005.