Real-World Language-Independent Early-Termination Analogy-Solver
言語に依存しない早期終了型類推解決手法

Yves LEPAGE, 飯田 仁
ATR 音声翻訳通信研究所
619-0288 京都府精華町相楽郡光台 2-2
{lepage, iida}@itl.atr.co.jp

はじめに

ソシュールは少壮文法学派の考えをより明確にして、類推とは、通時的な音声変化に影響されることになり乱れる言語上の形態的配列を適切な配列として捉える共時的な現象であるとまとめている [Saussure 16]。そして、「類推は、モデルとその規則的模倣を予想する。類推形とは、一個以上の他の形態を見習い、一定の規則にしたがって作られた形態である。」[小林英夫訳] と述べている。例えば、昔のラテン語の「honos」が「honor」になった理由は「orator」に関するモデルと等しくする為だった。

$$oratorem : orator = honorem : honor$$

一般的に類推の現象は三つのものから予測して一つのものを産出する事である。x を未知数として次の関係が得られる。

$$oratorem : orator = honorem : \mathtt{x} \quad \Rightarrow \quad \mathtt{x} = honor$$

アリストテレスの定義ではギリシャ語の語源を理解するのと同じ関係があることになった[1]。「ＡのＢに対する関係はＣのＤに対するようなものだ」と言う意味を「 Ａ：Ｂ ＝ Ｃ：Ｄ 」という比例式になぞって記述する。

1 情報処理

1.1 人工知能にある類推

２０世紀初めの物理学の有名な比較「電子の原子核に対する関係は惑星の太陽に対するようなもの」がアリストテレスの意味での類推だと言える。

類推に基礎をおく陰喩は多い [Steinhart 94]。例えば、「電子は太陽系と同じだ」。最近の人工知能の研究では陰喩のことを「類推」と呼ぶ傾向が強い [Gentner 83]。本稿では本来の意味での類推について議論する [Hoffman 95]。

[1] ギリシャ語でανα - は再び、 - λογιαは論理、議論、関係のこと。

1.2 単語間の類推

単語間だけの類推を対象に考えただけでも従来の研究について、いくつの問題点を指摘することができる。

- 一つは類推が起こる要素は全て同じ区域である点だ。これは人工知能研究と大きく異なる [Hall 89]。人工知能研究では意味的な区域をいくつかの間にマッピングする。
- もう一つは類推が起こる要素を等しくするための変化が何通りかある可能性がある点だ。例えば、「 自由：不自由な ＝ 用意：不用意な 」には挿入が二つある。これは類推に関する他の言語処理研究と大きく違う。例えば、[Nagao 84] や [Yvon 94] では複数変化の問題を同時に解くことができない。二つの要素間の変化が単純に一つの場合しか考察していない。

1.3 本研究の目的

本研究の目的は単語にしか起こらない類推を解決するアルゴリズムを提案することである。

- データの区域を文字列とする。
- 複数変化を可能とする。

「 Ａ：Ｂ ＝ Ｃ：Ｄ 」の類推を解くためにＡとＢやＡとＣの共通部分を探してその部分を一定の順番に並べて、その全体をＤをする。一つ目の問題は共通部分を探すことである。二つ目の問題はその部分を正しい順番に集めることである。

2 前置き

2.1 最大共通部分列

[Wagner & Fischer 74] のアルゴリズムで最大共通部分列を探すことができる。その手法では編集距離の行列を計算する。編集距離とは２つの文字列を同一にする為に必要な最小の編集操作（削除、挿入、置換）のコストである。

以下の再帰的関数で編集距離を計算する。ここで、a や b は文字を示して、Ａや Ｂ は文字列を示す。ε は空である。

$$
\begin{aligned}
\mathrm{dist}(a,\varepsilon) &= \mathrm{dist}(\varepsilon,b) = \mathrm{dist}(a,b) = 1 \\
\mathrm{dist}(a.\mathrm{A},b.\mathrm{B}) &= \min\left(\begin{array}{l} \mathrm{dist}(a,\varepsilon)+\mathrm{dist}(\mathrm{A},b.\mathrm{B}), \\ \mathrm{dist}(a,b)+\mathrm{dist}(\mathrm{A},\mathrm{B}), \\ \mathrm{dist}(\varepsilon,b)+\mathrm{dist}(a.\mathrm{A},\mathrm{B}) \end{array}\right)
\end{aligned}
$$

例えば、以下の行列では $like$ と $unlike$ や $like$ と $known$ の距離を一番右下の行列要素が示す。丸を書いた数の線に沿って最大共通部分列を表す。

	u	n	l	i	k	e		k	n	o	w	n
l	1	2	2	3	4	5	l	1	2	3	4	5
i	2	2	3	2	3	4	i	2	2	3	4	5
k	3	3	3	3	2	3	k	2	3	3	4	5
e	4	4	4	4	3	2	e	3	3	4	4	5

$\mathrm{dist}(like, unlike) = 2$。 $\mathrm{dist}(like, known) = 5$。

2.2　類似度

この文字列に対して、最大共通部分列の長さを類似度と呼ぶ。すなわちＡの長さと、ＡからＢを生成する為に削除や置換された文字数との差である。削除や置換された文字数は数学的な距離ではないので $\mathrm{pdist}(\mathrm{A},\mathrm{B})$ で表す。

$$\mathrm{sim}(\mathrm{A},\mathrm{B}) = |\ \mathrm{A}\ | - \mathrm{pdist}(\mathrm{A},\mathrm{B})$$

ＡになくてＢやＣで挿入された文字が類推の結果であるので類似度には数えられていない。

$$
\begin{aligned}
\mathrm{dist}(\varepsilon,b) &= 0 \\
\mathrm{dist}(a,\varepsilon) &= \mathrm{dist}(a,b) = 1 \\
\mathrm{dist}(a.\mathrm{A},b.\mathrm{B}) &= min\left(\begin{array}{l} \mathrm{dist}(a,\varepsilon)+\mathrm{dist}(\mathrm{A},b.\mathrm{B}) \\ \mathrm{dist}(a,b)+\mathrm{dist}(\mathrm{A},\mathrm{B}), \end{array}\right)
\end{aligned}
$$

この結果、編集距離と同じ様に行列で計算できる。

	l	i	k	e
u	1	1	1	1
n	2	2	2	2
l	2	2	2	2
i	3	2	2	2
k	4	3	2	2
e	5	4	3	2

	u	n	l	i	k	e
l	1	1	0	0	0	0
i	2	2	1	0	0	0
k	3	3	2	1	0	0
e	4	4	3	2	1	0

$\mathrm{pdist}(unlike, like) = 2$。 $\mathrm{pdist}(like, unlike) = 0$。

これで類推が解ける為Ａを基準にして以下の様に「 $like : unlike = unknown$: x 」の行列を書く。

n	w	o	n	k		u	n	l	i	k	e
1	1	1	1	1	l	1	1	0	0	0	0
2	2	2	2	2	i	2	2	1	0	0	0
2	2	2	2	2	k	3	3	2	1	0	0
3	3	3	3	3	e	4	4	3	2	1	0

2.3　重ね合わせの制約

類推ではＡの最初の要素が基準である。したがってＡにある文字がＢやＣで存在するはずだ。そうでなければ類推を説明できない。

類推が解ける為には、Ａにある文字が全てＢやＣにも存在しないといけない。すなわちＡとＢやＣの類似度の和よりＡの長さの方が小さくないといけない。$\mathrm{sim}(\mathrm{A},\mathrm{B}) + \mathrm{sim}(\mathrm{A},\mathrm{C}) \geq |\ \mathrm{A}\ |$, 同様に

$$|\ \mathrm{A}\ | \geq \mathrm{pdist}(\mathrm{A},\mathrm{B}) + \mathrm{pdist}(\mathrm{A},\mathrm{C})$$

Ａの長さが二つの pdist の和より大きくなれば、あるＡに存在する部分が三つの文字列に共通であると言える。従ってＤの類推の結果にも存在するはずである。これらの事実を、等式で書くと、次のようになり、それは類推の関係を示す制約になる。

$$
\begin{aligned}
|\ \mathrm{A}\ | &= \mathrm{pdist}(\mathrm{A},\mathrm{B}) + \mathrm{pdist}(\mathrm{A},\mathrm{C}) \\
&+ \mathrm{common}(\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D})
\end{aligned}
$$

3　アルゴリズムの概略

本アルゴリズムではあらかじめＡＢ間およびＡＣ間の pdist の行列を計算する。Ａの最後の文字からＡの最初の文字まで走査する。同じ行で二つの行列の最大共通文字列部分を同時に沿って以上の制約を検査する。

以下のアルゴリズムで「可能」とは制約を検査しながら行列の中に最大共通文字列の線に沿って進む事が出来ることを意味する。もちろん方向は三つあるので、それぞれが対角線、水平線、垂直線と表す。

```
func 類推解決(A, B, C)
  while B や C で可能
    if 対角線で両方の行列でも可能 then
      if a = b = c then
        D ← D.a
      else if a = b then
        D ← D.c
      else if a = c then
        D ← D.b
      end if
    else if 対角線で A と B の行列だけで
        可能 then
      if A と C の行列で水平線に可能 then
        D ← D.c
      else A と B でだけ進む
```

```
      else if 対角線でＡとＣの行列だけで
          可能 then
        ＢをＣに、ＣをＢにして以上と同じ
        end if
      else if 対角線で両方の行列でも
          不可能 then
        if 垂直線で両方の行列でも可能 then
          両方の行列で進む
        else if 垂直線でＡとＢの行列
            だけで可能 then
          D ← D.c
        else if 垂直線でＡとＣの行列
            だけで可能 then
          D ← D.b
        else if 垂直線で両方の行列
            でも不可能 then
          D ← D.x x は最小のコストを選ぶ
        end if
      end if
    end while
  end func 類推解決
```

4 特徴

4.1 早期終了

重ね合わせの制約によってＡにある文字がＢやＣでない場合は類推のない場合と分かる。その場合、Ａを全て読まないで早期に終了可能である。

もう一つは、Ａの長さが二つの pdist の和より小さくなると重ね合わせの制約が満たされない。この場合にも類推がなく、早期終了可能。

4.2 文字列で操作

4.2.1 書字方向

$\mathrm{A} = a_1.a_2...a_k$ の反転を $\overline{\mathrm{A}} = a_k...a_2.a_1$ で表すと、本アルゴリズムが、次の問題を解決できる場合には、

$\mathrm{A} : \mathrm{B} = \mathrm{C} : \mathrm{x} \;\Rightarrow\; \mathrm{x} = \mathrm{D}$

以下の関係も解決できる。

$\overline{\mathrm{A}} : \overline{\mathrm{B}} = \overline{\mathrm{C}} : \mathrm{x} \;\Rightarrow\; \mathrm{x} = \overline{\mathrm{D}}$

これはアラビア語に対してもそのまま使えるアルゴリズムであることを意味する。

4.2.2 接辞

付録で、様々な接頭辞、接尾辞、接中辞が解ける例を示して、本アルゴリズムの適用可能性の具体例を見る。

4.2.3 繰り返しや倒置

インドネシア語ではある単語の単数形からその単語を繰り返して複数形が作られる。例えば、「*orang*」は人のことで、「*orang-orang*」は何人かのこと。類推で言えば「 $orang : orang\text{-}orang = meja : meja\text{-}meja$ 」。ここで「meja」は机のこと。

倒置とは、ある二つの文字を倒置すること。例えば原セム語では「 $yaqtilu : qatil = yuqtilu : qutil$ 」で q と a や u と倒置した [Kuryłowicz 61]。

両方の現象は本アルゴリズムで解決することができない。

4.3 言語非依存とコード依存

本アルゴリズムは文字のレベルでしか操作しないからどんな言語でも扱える。類推は普遍的な現象であり [Itkonen 94] 本アルゴリズムはこの普遍性を満たすアルゴリズムである。

逆に本アルゴリズムはコードに依存する。例えば日本語の三種類の書き方を使って、違う結果が出ることを次に示す。

漢字かな: 待ちます : 待つ = 働きます : x
x = 無結果

ヘボン式: $machimasu : matsu = hatarakimasu : \mathrm{x}$
x = 無結果

以上の二つの類推が解けない理由は、本アルゴリズムでは文字レベルの下にある情報が見えないからだ。このため以下の類推が解決できない。

ち : つ = き : く　　$chi : tsu = ki : ku$

ところが以下の類推は解くことができる。

訓令式: $matimasu : matu = hatarakimasu : \mathrm{x}$
$\mathrm{x} = hataraku$

すなわち、ＡＢＣの類推が直接解決できない場合、記号化関数 f があれば次の様に類推が解ける。

$$f(A) : f(B) = f(C) : \mathrm{x} \quad \Leftrightarrow \quad A : B \equiv C : f^{-1}(\mathrm{x})$$

この式では左にある類推しか本研究のアルゴリズムで解けないと考えられる。一般的には、類推関係を見出すことは記号化の関数を見つける事だと言う。

おわりに

文字列のレベルでの類推を解くアルゴリズムを記述した。これは以下の特徴を持つアルゴリズムである。

- 現実の問題を解決（接尾辞や接頭辞や接中辞）
- 言語非依存でコード依存
- 類推のない場合に早期終了可能

今後は類推を使って、音声認識装置から出力される誤りを含む文を書き直すことを試してみる。また、類推の対象として文字列を考えるだけでなく、単語列の文についても考える。例えば、文では、それぞれの単語を品詞や意味コードに置換して、それにより作られる新しいパータンを本アルゴリズムに適用する。逆

に、あるコーパスから、その中にある文パターンを類推で取り出すことも可能であると考えている。

A　付録:　アルゴリズムの適用結果

A.1　接尾辞、接頭辞の挿入や削除

日本語:　自由 : 不自由な = 用意 : x
x = 不用意な

中国語:　科学 : 科学家 = 政治 : x
x = 政治家

ラテン語:　*oratorem* : *orator* = *honorem* : x
x = *honor*

フランス語:　*répression* : *répressionnaire* = *réaction* : x
x = *réactionnaire*

インドネシア語:　*tinggal* : *ketinggalan* = *duduk* : x
x = *kedudukan*

A.2　接尾辞や接頭辞の置換

日本語:　食べる : 食べます = 認める : x
x = 認めます

インドネシア語:　*kawan* : *mengawani* = *keliling* : x
x = *mengelilingi*

インドネシア語:　*keras* : *mengeraskan* = *kena* : x
x = *mengenakan*

ポーランド語:　*wyszedłeś* : *wyszłaś* = *poszedłeś* : x
x = *poszłaś*

英語:　*wolf* : *wolves* = *leaf* : x
x = *leaves*

A.3　接中辞

日本語:　乗る : 乗せる = 寄る : x
x = 寄せる

ドイツ語:　*lang* : *längste* = *scharf* : x
x = *schärfste*

ポーランド語:　*zgubiony* : *zgubieni* = *zmartwiony* : x
x = *zmartwieni*

ドイツ語:　*fliehen* : *er floh* = *schließen* : x
x = *er schloß*

A.4　複数接中辞

アラビア語:　*huzila* : *huzāl* = *ṣudi‘a* : x
x = *ṣudā‘*

アラビア語:　*arsala* : *mursilun* = *aslama* : x
x = *muslimun*

ヘブライ語:　*i̱aḥmoḏ* : *maḥmåḏ* = *i̱a‘ᵃḇọr* : x
x = *ma‘ᵃḇår*

参考文献

[Gentner 83] Dedre Gentner
Structure Mapping: A Theoretical Model for Analogy
Cognitive Science, 1983, vol. 7, no 2, pp. 155–170.

[Hall 89] Rogers P. Hall
Computational Approaches to Analogical Reasoning: A Comparative Analysis
Artificial Intelligence, Vol. 39, No. 1, May 1989, pp. 39-120.

[Hoffman 95] Robert R. Hoffman
Monster Analogies
AI Magazine, Fall 1995, vol. 11, pp 11-35.

[Itkonen 94] Esa Itkonen
Iconicity, analogy, and universal grammar
Journal of Pragmatics, 1994, vol. 22, pp. 37-53.

[Kuryłowicz 61] Jerzy Kuryłowicz
L'apophonie en sémitique
Ossolineum, Wrocław–Warszawa–Kraków, 1961.

[Nagao 84] Nagao Makoto
A Framework of a Mechanical Translation between Japanese and English by Analogy Principle
in *Artificial & Human Intelligence*, Alick Elithorn and Ranan Banerji eds., Elsevier Science Publishers, NATO 1984.

[Saussure 16] フェルディナン・ド・ソシュール
一般言語学講義
小林英夫訳 岩波書店

[Steinhart 94] Eric Steinhart
Analogical Truth Conditions for Metaphors
Metaphor and Symbolic Activity, 1994, 9(3), pp 161-178.

[Wagner & Fischer 74] Robert A. Wagner and Michael J. Fischer
The String-to-String Correction Problem
Journal for the Association of Computing Machinery, Vol. 21, No. 1, January 1974, pp. 168-173.

[Yvon 94] François Yvon
Paradigmatic Cascades: a Linguistically Sound Model of Pronunciation by Analogy
Proceedings of ACL-EACL-97, Madrid, 1994, pp 428-435.