継続計算に基づく生成文字列の置き換え

関洋平　原田賢一
慶應義塾大学理工学部
e-mail: yohei@hara.cs.keio.ac.jp

1 はじめに

文の生成は, 大きく分けて, 文の生成手法についての研究と, 文を生成するための入力の与え方についての研究に分けられる。文の生成については, 一文レベルの生成とテキストの生成に関する研究に分けられる。一文レベルの生成の研究については, 統語情報に基づく文法を用意して, 語彙レベルから構文構造を組み上げて生成を行うと, 入力に応じた柔軟な文の生成が可能となるが, 繰り返し同じような文字列を生成する場合には, 無駄な計算が多くなる。一方, 決まり切った言い回ししか行わないような対話システムについては, 生成される文の枠組を用意しておき, その一部分を置き換えて生成を行うという方法も知られており, 古い例では Eliza[1] が有名である。

本研究は, 双方の手法の利点を考慮し, 生成中の無駄な計算を減らした上で, 必要に応じた語彙の選択の柔軟性も維持できるような文の生成を目標として, 継続 (continuation) の概念に基づく文の生成手法を提案する。

2 継続 (continuation)

2.1 継続 (continuation) とは

継続 (continuation) とは, プログラム上の位置から, その後の最終結果を与える実行過程を表現する関数のことを言い [6], 一般に計算の制御の流れを定義する。実際に継続を関数の機能として組み込んでいる言語としては, Scheme[7] が有名であるが, 継続の概念はどのようなプログラミング言語にも存在する。

継続関数の利用法としては, (1) 非局所的脱出と, (2) 大域変数に保存した上での複数回の実行再開 (3) 関数手続き呼び出しの実現 [6] などが考えられる。非局所的脱出については, ある条件が成立すると, その手続き内の残りの手続きを一切省略して, ある返り値に基づいてその手続き終了後の計算を続けるというかたちで実現する。これとは別に, 生成した継続関数を大域変数に保存しておき, 後から引数を与えて呼び出すと, 引数に応じた計算の再開を何度も実行できる。関数手続き呼び出しについては, 呼び出した時点で呼び出し元における継続を受け取り, 手続き内の命令を実行した上で, それを加えた継続を返すという形で実現可能であり, [5] では, 分岐命令に基づいた末尾再帰的な形での関数呼び出しの実現について述べている。

2.2 表層文生成における応用

表層文生成とは, おおざっぱにいって, 何らかのひとまとまりの意味の構成基本単位 (意味素) の意味的関係に基づく組合せを入力として, 統語的な情報に変換を行い, 表層的な文を生成して出力するということを行う。この意味素や意味的関係としてどのようなものを設定するかは応用に応じて異なる。本研究では, 以前に生成した意味素または意味素を組み合わせた要素についての生成を、質問応答システムと言うかたちで試みている。焦点 [3, 4] や旧情報・新情報などの情報構造 [2] に基づいて分類し, 以前生成された文についての情報を保存しておくことで, テキストの流れに応じた文の生成の計算を減らすということを試みる。

An Approach for permuting parts of phrases in sentence generation based on continuation function.

Yohei SEKI and Ken'ichi HARADA
Faculty of Science and Technology, Keio University
3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223, JAPAN

3 生成文字列の置き換え

本研究で提案する内容は，各構文単位ごとに生成する手続きを再帰的に呼び出しており，外側からトップダウンな生成手法を取っている。また，過去に生成された文の意味素の情報については，各構文要素ごとに文の内容に応じて環境に大域変数として保存しておく。以下では，幾つかの生成例に対して，継続計算を利用した文字列を置き換える例を示す。

1. 疑問詞に対する応答文の生成 (日本語)

   以下のような scheme の関数を準備する。(語順は調整される)

```
(list 述部生成部
      (call/cc
            (lambda (whphrase)
                  (set! c whphrase)
                  疑問詞生成部))
            助詞句生成部)
```

   助詞句生成部は，連想リストを参照することにより，既存の登録要素が見付かった場合には代名詞化を行い，そうでない場合にはリストに登録する。以上により，(c 京都)(c 花子) などとすることにより以下の応答文が生成される。

   (a) • 太郎は，花子とどこへ行ったの?
   • 彼は花子と京都へ行った

   (b) • 太郎は京都へ誰と行ったの?
   • 彼は京都へ花子と行った

   (c) • 太郎は，どの作家の推理小説が好きなんですか?
   • 彼は，松本清張の推理小説が好きなんです。

2. Wh 疑問文に対する応答における部分的文字列の省略 (英語)

   (a) Where did John and Bill meet Mary?

   (b) John met Mary in Harvard Square,
   and Bill [met Mary] in Porter Square.

   一般に質問で与えられている事象は先攻文脈で了解されているため，応答 (b) の第二文は動詞と目的語が省略されている。このような生成は，質問文の直後の応答の場合，同じ事象内容を生成しようとした際に，共有していない異なる句要素部分 (この場合は Bill) を除いた生成に際して，非局所的脱出の方法を応用することにより可能となる。

3. Yes/No 疑問文に対する応答文の生成と事象句の共有 (英語)

   (a) Did you get many medals in Atlanta Olympics?

   (b) Yes, I got many medals.

   Do で始まる疑問文は，代名詞の置き換え，時制の置き換えを除いて，事象句を共有する。このような生成計算は，置き換わる部分を除いて，継続関数として保存しておき，引数のみを置き換えて生成を行う。

4 おわりに

本研究で提案した手法は，既出要素に基づいた生成手法であり，照応や一部文字列の共有に応用可能である。

参考文献

[1] 石崎 俊: 自然言語処理, 昭晃堂 (1997).

[2] 神尾昭雄, 高見健一: 談話と情報構造, 日英語比較選書 2, 中右 実 編, 研究社出版 (1998).

[3] 関洋平, 飯島正, 原田賢一: "意味焦点駆動文生成," 情報処理学会研究報告 99-NL-131-11(1999), pp.79-85.

[4] Candace L.Sidner: "Focusing in the Comprehension of Definite Anaphora, " In Computational Models of Discourse, edited by M.Brady and R.C.Berwick, MIT Press, Chapter 5, pp.267-330(1983).

[5] Guy Lewis Steele Jr.: "Debunking the "Expresive Procedure Call" Myth or, Procedure Call Implementations Considered Harmful or, LAMBDA: The Ultimate GOTO," Proc. of 30th ACM Natl.Conf. , Seatle, pp.153-162(1977).

[6] 武市 正人 著: プログラミング言語, 岩波講座 ソフトウェア科学 4, 岩波書店 (1994).

[7] 湯浅太一: Scheme 入門, 岩波書店 (1991).