3M3-3

小説生成器とはどんなシステムか

What is Automatic Story Generator?

佐藤 理史
Satoshi Sato

名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University

This paper shows an answer to the question "what is automatic story generator." Because of the vagueness of the generator input, the function and configuration of the generator are unclear. Starting from three basic requirements, this paper draws an outline of a system that we can call automatic story generator.

1. はじめに

我々は,「きまぐれ人工知能プロジェクト『作家ですのよ』」[1] に参加し，コンピュータによるショートショートの自動生成の実現に取り組んでいる [2, 3, 4]．自動生成の実現とは，結局のところ,「小説生成器」と呼びうるシステムの実現に帰着されるが，それがどのような要件を満たすシステムであるかは，明確ではない．

「小説生成器」の出力は「小説」である．そこまでは揺るぎない.「小説」が何を意味するかを定義することは容易ではないが，自動生成の文脈においては,「破綻していないお話（テキスト)」ぐらいのガイドラインでよいだろう．一方,「小説生成器」の入力は，よくわからない．入力が定まらなので，システムの機能も定まらない．現状は，このような状況であるから,「小説生成器」を作る研究のかなりの部分は,「小説生成」という問題をどのように定義するかという，設計問題となる．

このような背景により，本稿では,「どんなシステムを作ったら『小説生成器を作った』と言えるか」について検討する．これは,「小説生成」という問題自体を定義することを意図したものではなく,「小説生成器」の一つの具体例を示すことを試みるものである．

2. 出発点

小説を出力するシステムを作ることは簡単である．出力すべき小説をそのまま内部に持っておけばよい．しかし，これを誰も小説生成器とは呼ばないだろう．一方，言語的な情報として語のリストだけを保持しているシステムが，それらの語を組み合わせて小説を作り上げれば，ほとんどの人は小説生成器と認めるだろう．我々が作ることができる小説生成器は，これら 2 つの両極端の間のどこかに位置付けられることになろう．

議論の出発点として，小説生成器 (以下では，単にシステムと呼ぶ) が満たすべき要件として，次の 3 つの要件を設定する．

要件 1 システムが出力するテキストは,「お話」として破綻していないこと．

要件 2 システムは，実際に，多くの異なるテキストを生成できること．

要件 3 システムは，あらかじめ内部に，出力するテキストをそのままの形で保持していないこと．あるいは，システムの入力として，出力するテキストをそのままの形で与えないこと．(つまり，システムは，何からの方法で，より小さな「部品」からテキストを組み上げること．)

要件 2 の「多くの」が意味する数は，たとえば「数千以上の」である．この要件が満たされれば，事実上，要件 3 は自動的に満たされることになると思われる．

連絡先: 佐藤理史，名古屋大学大学院工学研究科電子情報システム専攻，〒 464-8603 名古屋市千種区不老町 C3-1 (631), ssato@nuee.nagoya-u.ac.jp

3. システムの図式

次に，システムの抽象的な図式を描いてみよう．

1. システムを $\mathcal{M}$ と書くことにする．
2. システム $\mathcal{M}$ の出力は，小説（テキスト）である．これを t と書くことにする．
3. システム $\mathcal{M}$ の入力はどんなものかわからないが，これを P と書くこととする．これは，空でもよいこととする．

これらにより，以下の図式が描ける．

$$P \longrightarrow \boxed{\mathcal{M}} \longrightarrow t \tag{1}$$

次に，入力 P と出力 t の関係について考えよう．P によって t が一意に定まるのであれば，$\mathcal{M}$ は関数となり，$\mathcal{M}$ の機能は (広い意味での) 形式変換となる．つまり，入力 P は，出力テキスト t の，いわば「完全な設計図」となる．このとき，$\mathcal{M}$ は「P をテキスト化した」と言うことはできたとしても，小説を作ったとは言いがたいだろう．それゆえ，次のような関係を設定する．

$$t \in T = \mathcal{M}(P) \tag{2}$$

この式が意味するところは，次のことである．

- 入力 P が定まると，システム $\mathcal{M}$ が生成できるテキストの集合 T が定まる．
- システム $\mathcal{M}$ は，その集合 T の一つの要素 t を (たとえばランダムに) 出力する．

このように考えると，入力 P は,「出力すべきテキストが満たすべき条件」をシステム $\mathcal{M}$ に伝えていることになる．以降，この P を (小説の) 設計図と呼ぶことにする．

4. 完全な設計図と不完全な設計図

2 節の出発点で示した要件 3 は，テキストをより小さな要素から組み上げることを要求する．ここで,「テキスト以外の記号を言語化すること，および，語や句や文などから (小説全体の) テキストを組み上げる操作」を総称して，**テキスト化**と呼ぶことにする．システム $\mathcal{M}$ がテキスト化の機能を持たなければならないことは，要件 3 より自明である．

いま，次の条件を満たすような設計図 $\dot{P}$ を「完全な設計図」と呼ぶこととしよう．

$$|\mathcal{M}(\dot{P})| = 1 \tag{3}$$

完全な設計図から生成できるテキストは 1 種類のみである．このときの $\mathcal{M}$ の機能が，テキスト化である．

次に，$\mathcal{M}$ が生成しうるすべてのテキスト t に対して，t を生成する完全な設計図が (少なくとも 1 つは) 存在することを仮定しよう．この仮定は,「テキストを生成する前に，まず，完全な設計図を作る」か，あるいは,「テキストが生成された暁には，完全な設計図が (副産物として) できあがっている」ことを意味する．

一方,

$$|\mathcal{M}(P)| > 1 \tag{4}$$

を満たす設計図を「不完全な設計図」と呼ぶ．不完全な設計図には，完全には詳細化されていない部分が存在する．一般に，入力 P は不完全な設計図であるから，システムには，この部分を詳細化しなければ，生成するテキスト t を決定できない．この詳細化，すなわち，設計図を完成させる作業が，システムの二つ目の機能となる．入力として与えられる不完全な設計図を「骨格」とみなせば，この作業は「肉付け」作業である．そして，無から有を生み出せないコンピュータにとっては，この「肉付け」作業の実体は，(あらかじめ用意された，あるいは，なんらかの計算によって作り出される) 選択肢からの選択にならざるをえない．

以上の議論をまとめると，システムは次の 2 つの機能を持つことになる．

1. **設計図の詳細化**：不完全な設計図 P を肉付けして，式 (3) を満たす完全な設計図 $\dot{P}$ を作ること．
2. **テキスト化**：完全な設計図 $\dot{P}$ をテキスト t に変換すること．

5. 設計図の中身

残された課題は，設計図 P の中身をはっきりさせることである．いささか抽象的な議論が続いたので，ここでは，具体例に即して検討を進めよう．具体例として，我々が現在作ろうとしている「スマホ物語」の冒頭の導入部分を用いる．

> スマホが震えた．深夜一時ごろ．ここは，薄暗い研究室のなか．鈴木邦男は，先月ここに配属されたばかりであるが，平均帰宅時間はすでに深夜零時を超えている．邦男は大きなあくびをしながら，ポケットからスマホを取り出した．

このようなテキストを生成するためには,「テキストの設計図」のようなものが必要である．それは，おおよそ,「何をどのような順番で書くか」を示すものである．これを**テキストプラン**と呼ぼう．上記のテキストは，たとえば，次のようなテキストプランを実体化したものとみなすことができる．

導入部のテキストプラン
1. イベント＜スマホに電話が着信した＞の描写
2. そのイベントが起きた時空の描写
a. 時間の描写
b. 空間の描写
3. ＜主人公＞の導入と描写
4. イベントに対するアクション＜電話に出ようとする＞の描写

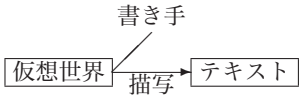
テキストプランは，生成すべき文のプランを終端ノードとする木構造である．それぞれの文プランに従って文を生成すれば，全体のテキストが完成する．文プランから文を生成する方法には色々な実現法が考えられるが，一つの簡便な方法は，文プランに対して文テンプレートを用意しておき，それを実体化する方法である．たとえば，＜スマホに電話が着信した＞に対して,

s_1　*phone が *ring

というテンプレートを用意し，*phone や*ring の値を，それぞれ用意した選択肢の中から選べば，先ほどのテキストの 1 文目に相当する文が生成できる．

このような仕組みは色々な方法でより複雑にすることができる．それに応じて，生成できるテキストの種類も増えよう．しかしながら，どんなに複雑化したとしても，結局のところ，処理の実体は「テキストを，より小さなテキストの切り貼りによって作る」ことに過ぎない．テキスト (小説) と，それが描写する世界を区別しないのであれば,「小説生成＝テキストの切り貼り」である [2, 3].

小説に対する典型的な捉え方の一つは,「ある仮想世界 (のイベント列) を描写したもの」であろう．つまり，次のような図式である．



(5)

たとえば，先ほどのテキストは，次のような仮想世界を記述しているものとみなすことができる．(一部省略した．)

1. 1 か月前	
オブジェクト	＜鈴木邦男＞ (性別=男，職業=学生)
	＜研究室＞ (詳細情報不明)
イベント列	1. ＜鈴木邦男＞が＜研究室＞に配属される
2. 現在	
時刻	真夜中
場所	＜研究室＞
オブジェクト	＜鈴木邦男＞ (連日遅くまで研究している)
	＜スマホ＞ (モード=マナーモード)
イベント列	2a. ＜鈴木邦男＞は研究している
	2b. ＜鈴木邦男＞の＜スマホ＞に電話が着信する
	2c. ＜鈴木邦男＞が電話に出ようとする

この仮想世界のイベント 2b を記述する文テンプレート s_1 の*ring の値は，仮想世界の＜スマホ＞のモードの設定に依存すると考えるのが妥当である．マナーモードであれば「震えた」となるし，ノーマルモードであれば「鳴った」や (着メロ設定であれば)「歌い出した」などの選択肢が考えられよう．逆に，テキストが「震えた」となっていれば，それは暗に，仮想世界の＜スマホ＞がマナーモードになっていることを示唆するのである．

「鳴った」と「震えた」という語の選択肢から，そのどちらを選ぶかは，単なる表現の選択である．一方，システムの内部に仮想世界を導入すれば，その仮想世界の＜スマホ＞の着

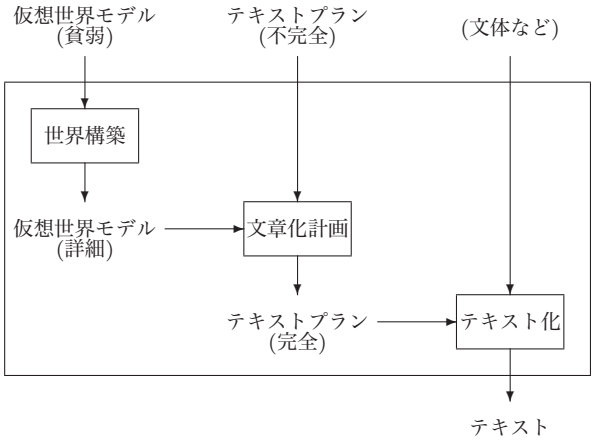


図 1: 小説生成器の構成 (概念図)

信モードをどう設定するかという，仮想世界の詳細化の選択となる．

それでは，仮想世界を十分に詳細に設定すれば，テキストは一意に定まるのであろうか．この答は，否である．一つの仮想世界を描写する方法は無数にある．たとえば,「マナーモードのスマホに電話の着信があった」ことを描写する場合でも，何をどこまで描写するかによって，次のような選択肢が考えられる．

- スマホが震えた．
- スマホがブルブルと震えた．
- ポケットの中のスマホが震えた．
- ポケットの中のスマホがブルブルと震えた．
- ポケットが震えた．
- ブルブルブル．ポケットのスマホが震えた．

つまり，仮想世界の設計図 (これを**仮想世界モデル**とよぶ) があったとしても，依然として，仮想世界をどのように描写するかというテキストの設計図 (テキストプラン) は必要である．

以上をまとめると，前節で設計図 P と呼んでいたものは，(少なくとも) 次の 2 つの部分から構成されることになる．

1. **仮想世界モデル** (仮想世界の設計図)：小説の描写対象とする世界をどのように設定するか
2. **テキストプラン** (テキストの設計図)：その仮想世界を，描写するか

6. システム構成図

これまでの議論を踏まえた小説生成器の構成図を図 1 に示す．入力として与えられる 2 つの設計図，すなわち，仮想世界モデルとテキストプランは，いずれも不完全である．テキストを生成するためには，テキストプランを完全なものにする必要があり，そのためには，仮想世界モデルをより詳細にする必要がある．

この図は概念図であり，プロセス (動作) モデルではない．仮想世界モデルは，いくらでも詳細化が可能である．つまり，完全な仮想世界モデルというものはありえない．必要な詳細レベルは，テキストプランによって規定される．そのため,「まず，十分に詳細な仮想世界モデルを作り，それに基づいてテキストプランを完全なものにする」という手順は採用できない．そのゆえ,「テキストプランを詳細化する過程で，必要に応じて仮想世界モデルを詳細化する」[5] という動作モデルを採用する．

この図では，仮想世界モデルを詳細化することを**世界構築**，テキストプランを詳細化することを**文章化計画**と名付けた．これらは，それぞれ,「お話の世界を作ること」,「その世界をどう描写するか (語るか) 決めること」に対応する．**これらは概念的には区別されるべきである**．つまり，同一の仮想世界モデルであっても,「どう語るか (テキストプラン)」が異なれば，異なるテキストが生成される．そのような自由度を担保すべきである．

最後の**テキスト化**は「テキストを書くこと」に対応する．実際の人間の小説執筆ではこれら 3 つの処理は密接に絡み合っているであろうし，作成するシステムにおいても厳格にモジュール化できないことも予想される．しかしながら，小説生成を,「話を作る」「どう語るか決める」「書く」という 3 つの要素の組み合わせとして捉えるという見方は必要であり，小説生成器が何を自動化しているかを整理する切り口となる．

7. 小説生成器の機能

ここでは，図 1 の 3 つのモジュールについて，その機能を掘り下げよう．

「テキスト化」モジュールは，いわゆるテキスト生成器 [6] に相当し，文字列としてのテキストを生成する機能を担う．モジュールの入力となるテキストプランの設計にも依存するが，おおよそ単文に対応する深層構造の列を，一続きの文の列に変換する処理と考えるのが妥当であろう．そこで行なう処理の中核は，aggregation (必要に応じて複数の単文を複文にまとめる処理) と，表層文生成である．我々は，現在，表層文生成を行なう文生成器を作成中である [7].

「文章化計画」モジュールは，仮想世界のどの部分を切り出し，それをどのような順番で「語るか」を決めることを担う．もし，仮想世界モデルが十分に詳細化されていれば,「文章化計画」モジュールは単独でテキストプランを生成できるが，不十分であれば，仮想世界モデルの詳細化を「世界構築」モジュールに依頼することになる.「文章化計画」単体の実現法として，典型的には，次のような方法が考えられる．

- ストーリー文法により，テキストプラン (作るべきテキストの構造と内容) を実体化する [8]．これは，概念的には，起承転結のような，テキストの典型的なパターンを準備することに等しい．
- イベントの描写，オブジェクトの描写に対して，それぞれ，多数の文テンプレート (あるいは，それに類するもの) を用意する．これは，文レベルのパターンを準備するということである．
- 同様に，(登場人物間の) 対話に関しても，対話の典型的なパターンと，典型的な発話を文法やテンプレートという形で用意する．
- 典型的なレトリックを生み出す仕組みを用意する [8]．たとえば，比喩や擬人化など．これも，実際には，テンプレートや実例，あるいは，類推のような機構を用意することになる．

このモジュールは，物書きの「スキル」に相当する．システム作成においては，このモジュールをどこまで作り込むかによって，生成できるテキストの多様性と品質が決まると考えられる．

「世界構築」モジュールは，語るべき対象となる仮想世界を作ることを担う．これは，人間においては，想像力あるいは創造力と呼ばれる能力に相当するのかもしれない．しかしながら，システムを作成する立場に立てば，そこで必要とされるこ

とのほとんどは常識の範疇であり，そこに，ほんの少しだけ非常識 (あるいは，ランダムな要素) が含まれるように思われる．システムでは，仮想世界モデルのイベント列やオブジェクトの生成や詳細化を行なうことになるが，この方法として，たとえば，次のようなものが考えられる．

- あるイベントを出発点として，因果関係の連鎖の知識を用いて，そこから起こりうるイベント列を生成する．(常識的推論)
- 既存の小説等からイベント列を抽出し，それを仮想世界のイベント列として採用する [9, 10]．(過去の小説からの流用)
- 外部知識を用いて，仮想世界中のオブジェクトを肉付けする．たとえば，ウェブで公開されている個人のプロファイル情報を用いて，主人公の各種情報 (名前，性別，趣味，性格等) を設定する．(経験的知識の利用)
- 既存の小説に出現するオブジェクトの一部を変更して，仮想世界のオブジェクトを作る．(過去の小説からの流用)
- 歴史的事実の一部を変更して，仮想世界を構築する．たとえば,「将軍が女性だったら」など．(少しの非常識)

私の見るところ，小説の自動創作の研究者の主な興味の対象は,「世界構築」であるように思われる [11]．確かに，語るべき対象が構築されない限り小説は存在し得ないのだから，その選択は頷ける.「話を作るために，その骨格を作る．その前提となる (仮想世界の) イベント列を作る．それこそが，創作の根幹だ」．それは一つの立場である．

一方，システムが出力するテキストの出来栄えを気にするのであれば，プロセスの下流，すなわち，出力に近いところに注力する必要がある．読むに耐えうるテキストを生成したいのであれば，まずは「テキスト化」が重要であり，次に「文章化計画」がポイントとなる．この 2 つのモジュールが実現できれば，十分に詳細なレベルで与えられた仮想世界に対して，その小説化 (ノベライゼーション) が可能となる．これが実現できないようであれば，いかにうまく仮想世界が自動的に構築できるようになろうとも，人間が読むに耐えうる小説は生成できないであろう．

8. 誰が小説を作ったのか

いま，図 1 のようなシステムを私が作成し，その出力テキストが「小説」と認められたとしよう．最後の問いは「そのテキストを作ったのは誰か」ということである．それは，私か，あるいは，システムか.「それはシステムである」と認められて，はじめて「小説生成器が作れた」ことになる．

コンピュータプログラムであるシステム $\mathcal{M}$ は，抽象的には，入出力関係を規定するものに他ならない．そのシステムの作成者は，その規定を作った張本人である．その意味において,「テキストの間接的な作者がシステム作成者である」ということからは逃れようがない．

しかしながら，システムが十分複雑になれば，システム作成者はその挙動を完全には予測できなくなる．加えて，たとえば，システムがある入力 P に対して，潜在的に出力できる多数のテキストのうちの 1 つを (ランダムに) 出力するのであれば，何が出力されるのかは，ほどんど予測できなくなる．このような状況になれば,「システム作成者がテキストを作成した」というよりは,「システムがテキストを作成した」という方が，現実をより良く言い表していることになるだろう．

結局のところ，私は，システムがどれだけ多くの種類のテキストを生成できるかがポイントだと考えている．それに基づいて，2 節の出発点で示した要件 2 は，次のようにブレークダウンする．

2a. システムは，ある入力 P に対して色々なバリエーションのテキストを生成できること．
2b. システムは，異なる入力 P に対して，異なるテキスト集合を生成できること．

これらの要件が達成できれば,「私が作ったシステムは小説生成器であり，小説生成器が小説 (テキスト) を作った」と言ってよいと考える．

謝辞 本研究は，JSPS 科研費 24300052，および，中山隼雄科学技術文化財団の研究助成に受けて実施した．

参考文献

[1] 松原仁, 佐藤理史, 赤石美奈, 角薫, 迎山和司, 中島秀之, 瀬名秀明, 村井源, 大塚裕子. コンピュータに星新一のようなショートショートを創作させる試み. 2013 年度人工知能学会全国大会論文集, pp. 2D1–1, 2013.

[2] 緒方健人, 佐藤理史, 駒谷和範. 模倣と置換に基づく超短編小説の自動生成. 2014 年度人工知能学会全国大会論文集, pp. 1C3–OS–14b–2, 2014.

[3] 高木大生, 佐藤理史, 駒谷和範. 会話を中心とした超短編小説の自動生成. 2014 年度人工知能学会全国大会論文集, pp. 1C3–OS–14b–3, 2014.

[4] 高木大生, 佐藤理史, 松崎拓也. プロットと背景知識を用いた短編小説の自動生成. 情報処理学会第 77 回全国大会講演論文集, 2015.

[5] Ivo Swartjes and Mariet Theune. The virtual storyteller: Story generation by simulation. In *Proceedings of the Twentieth Belgian-Netherlands Conference on Artificial Intelligence*, pp. 257–265, 2008.

[6] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[7] 緒方健人, 佐藤理史, 松崎拓也. 文節木の段階的実体化による日本語文生成器の作成. 2015 年度人工知能学会全国大会論文集, 2015.

[8] Selmer Bringsjord and David A. Ferrucci. *Artificial Intelligence and Literary Creativity*. Lawrence Erlbaum Associates, Inc. Publishers, 2000.

[9] Neil McIntyre and Mirella Lapata. Learning to tell tales: A data-driven approach to story generation. In *Proc. of ACL/IJCNLP-2009*, pp. 217–225, 2009.

[10] Neil McIntyre and Mirella Lapata. Plot induction and evolutionary search for story generation. In *Proc of ACL-2010*, pp. 562–1572, 2010.

[11] Pablo Gervas. Computational approaches to storytelling and creativity. *AI Magazine*, pp. 49–62, 2009.