動く文法!? －文をブロックで組み立てる－

佐藤理史（名古屋大学大学院工学研究科）

1. HaoriBricks3

どんな文でも簡単に合成できるソフトウェアシステムがほしい。どうすればそれが達成できるのか。小説の自動生成を目指すプロジェクト[1]の一環として、この課題に解決に取り組み、たどり着いたひとつの答えが、ブロック玩具にヒントを得た HaoriBricks3 (HB3) というシステムである[2]。

レゴに代表されるブロック玩具では、小さなブロック（レゴではブリック brick と呼ぶ）を組み合わせ、建物、乗物、動物など多様なモデルを作ることができる。これと同じように、HB3 では、日本語の文を組み立てるためのブリック（部品）を提供し、それらを組み合わせることで文を合成する。より具体的には、ブリックコードとよばれる記述形式で、文を生成するプログラムを記述する。そして、このコードを実行すると、いくつかの内部構造を経て、表層文字列が出力される。図 1 に、ブリックコードから表層文字列が得られるまでの過程の概略を示す。

ここで重要なことは、ブリックコードは、文を組み立てるコード（プログラム）であるという点にある。つまり、コードの任意の部分を変数化したり、条件分岐を追加したりすることができる。さらに、複数のコードを組み合わせることができるので、たとえば、単文を生成するコードを２つ組み合わせて複文を合成することなどが、簡単にできる。

2. 日本語を構成する部品は？

HB3 を実現するために行ったことを比喩的に言うならば、「工業製品と同じように日本語（という装置）を分解する」こととなろう。たとえば、自動車という製品は、2〜3 万個の部品から構成され、それらを巧みに組み合わせることによって「移動する」という機能を実現している。では、「情報を伝える」という機能を実現する装置として日本語を捉えた場合、どんな部品があり、それらはどのように組み合わされているのであろうか。そのような考え方に基づき、益岡・田窪文法[3]を出発点に、文生成に必要な文法事項を整理・再構成した体系が**羽織文法**である。羽織文法は、プログラム（動く文法）として HB3 の内部に定義されている。

日本語の文法体系は、おおよそ、次の要素から構成されるのが標準的であろう。

(1) 語の分類体系（いわゆる品詞体系）
(2) 活用する語の活用体系
(3) 構文構造の定義（文の背後にどんな構造を仮定するか）
(4) 文法的機能（テンス・アスペクト・モダリティなど）とその表現形式

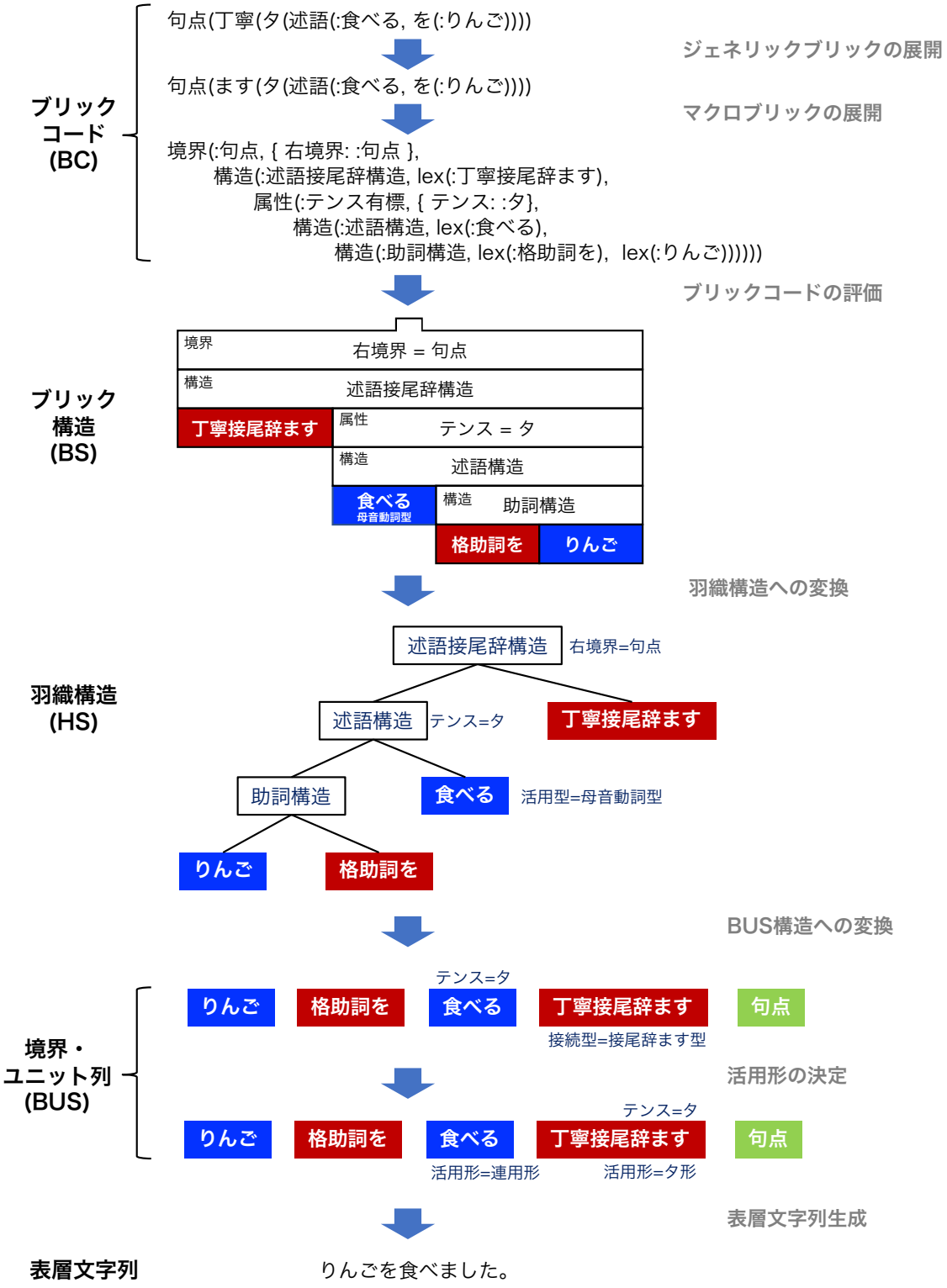

図 1 ブリックコードから表層文字列を生成する

羽織文法は、次のような特徴を持った文法である。

(1) 語の分類では、機能語と内容語の区別、活用する語と活用しない語の区別を重要視する。品詞は重要視しない。
(2) 詳細な活用体系（活用型と活用形）を提供する。これに加えて、**接続型**（後述）を提供する
(3) 表層文の背後に、羽織構造を仮定する。
(4) 文法的機能のほとんどを、機能語の働きによるものと考える**機能語中心主義**を採用する。
(5) 句読点や括弧などの記号を語とは明確に区別し、**境界記号**とみなす。

羽織文法の詳細は文献[2]を参照されたい。HB3 では、ブリックという考え方を用いて、文法をできるだけ隠蔽する方針をとっているため、この文法を熟知しなくても、ブリックコードを記述することができる。

図 1 の一番上のコードが、「りんごを食べました。」という文（表層文字列）を生成するために、実際に記述するコードである。このコードに含まれるコロンで始まる要素、すなわち、「**:食べる**」と「**:りんご**」が内容語に対応するブリックで、それぞれ「食べる」、「りんご」という内容語を生成する。

それ以外の要素は引数をとるブリックで、引数に対して、何らかの情報や操作を付加する。「**を**」は格助詞「を」を付加するためのブリックで、「**を(:りんご)**」というコードは、「りんごを」という文字列を生成する。「**述語**」は述語構造を作るブリックで、第 1 引数に述語、第 2 引数以降に述語が支配する要素を記述する。すなわち、「**述語(:食べる,を(:りんご))**」は、「りんごを食べる」を生成する。

「**タ**」はテンス有標を表す特別なブリックである。「**タ(述語(:食べる,を(:りんご)))**」を表層文字列化する仕組みは、結構複雑である。

(1) 「**述語**」の第 1 引数を、活用する語と解釈する
(2) 「食べる」の活用型を母音動詞型と推定する
(3) テンスを「食べる」で実体化できる（「食べる」にタ形が存在する）ことを確認するという一連の過程を経て、「りんごを食べた」を生成する。HB3 は活用する内容語の活用型を自動推定する機構を有しているので、特別な場合を除き、活用型を明示的に指定する必要はない。（つまり、内容語の辞書は不要である。）

「**丁寧**」は述語を敬体（丁寧体）に変更するブリックである。このブリックは、敬体に変換する語の活用型が動詞型の場合は「**ます**」を、イ形容詞型・判定詞型の場合は「**です**」を付加する。この場合は動詞型なので「**ます**」が選ばれる。「食べた」を生成するコードに「**ます**」を追加すると、「食べたます」ではなく、「食べました」を生成する。これを実現するのが接続型とテンス順送り機構である。

接続型は、機能語が、直前の語の活用型・活用形をどのように制約するかを記述するために導入した制約類型である。たとえば、「ます（丁寧接尾辞ます）」の接続型は、「接尾辞ます型接続」で、この接続型は、

大分類	中分類	数
カーネル	基本ブリック	6
	その他	2
文法	活用型	53
	活用形	55
	接続型	89
	構造	14
	品詞	15
	その他	19
小計		253

大分類	中分類	数
機能語	助詞	115
	判定詞	1
	助動詞	93
	述語接尾辞	74
	複合辞	326
機能語以外	接頭辞	6
	連体詞	12
	慣用句	121
境界記号		30
小計		778

大分類	中分類	数
従属節	従属節	190
	従属節読点	170
	句	108
	節末機能語	39
	複合辞	22
ジェネリック		88
マクロ		54
テキスト		12
その他		1
小計		684

表 1 Basic パッケージに含まれるブリック数

(1) 活用型が子音動詞型ラ行イ（「いらっしゃる」）の場合は、連用イ形（いらっしゃい-ます）と連用形（いらっしゃり-ます）に接続可能
(2) それ以外の動詞型の場合は、連用形（食べ-ます、書き-ます）に接続可能
と定義されている。

この接続型の定義に従い、「ます」は「食べる」に連用形を要求する。そうすると、タ形を採用できなくなり、テンスを実体化できなくなる。実体化できない場合、テンスは後続要素に順送りされる。これが**テンス順送り機構**である。後続要素「ます」はタ形を取ることができるので、ここでテンスが実体化され、「食べました」が生成される。

最後に残った「**句点**」は句点を付加するブリックである。

以上のように、HB3 では、ブリックを次々と組み合わせていくことにより、文を組み立てるコードを記述する。

(1) 内容語は、内容語ブリック（コロン付きの文字列）として記述する。
(2) すべての機能語に対して、その働きを組み込んだ機能語ブリックが用意されている。
(3) 構造を組み立てるブリックが用意されている。主要な構造は、述語構造、修飾構造、並列構造である。機能語にもその種別毎に機能語構造（たとえば、助詞構造）があるが、これは、(2)の機能語ブリックに組み込まれている。
(4) テンス有標を表す特別なブリックが用意されている。
(5) 句読点、括弧の類を表すブリック（境界記号ブリック）が用意されている。

これらのブリックのうち、HB3 の中核は、(2)の機能語ブリックである。羽織文法では、文法が扱う機能語をすべて列挙し、そのそれぞれに、ID、機能語の細分類、活用型、接続型を定義している。この機能語の定義が羽織文法の中核であり、その定義に基づいて、HB3 の機能語ブリックが提供されている。

表 1 に、Basic パッケージとよばれる HB3 の標準仕様で提供されているブリック数を示す。総計は 1,715 種類で、別名を含めると 2,917 種類となる。

3. 動く文法

HB3 で実現したことは、文の合成に関連する文法現象をブリック（部品）という形で

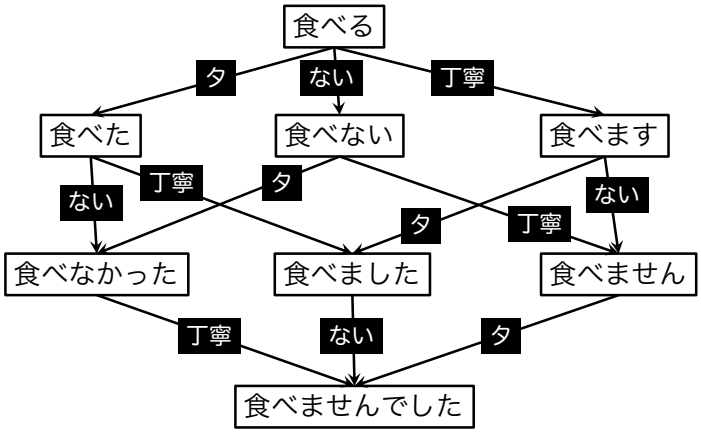

図 2 テンス・ない・丁寧の接続順序の自由化

抽象化したことと捉えることもできる。ここでは、いくつかの言語現象をどのように扱っているかを示す。

最初の具体例は、ボイス接尾辞（「れる、られる、せる、させる」）の付加である。母音動詞型（「食べる」）には「られる・させる」が接続するが、子音動詞型（「書く」）には「れる・せる」が接続する。動詞（の活用型）が決まれば付加すべき接尾辞は定まるが、動詞が変数化された場合、コード中には具体的な接尾辞を書くことができない。このような現象に対処するために、「**受身**」や「**使役**」ブリックを用意した。これらのブリックは、直前の動詞が定まった時点で、その活用型を調べ、適切な接尾辞を選択する。このように、特定の機能語には直接対応せず、抽象化された機能を提供するブリックを**ジェネリックブリック**と呼んでいる。

前節で説明した「**丁寧**」や、否定を表す「**ない**」もジェネリックブリックである。これにテンス有標を表す「**タ**」を加えた３つのブリックは、どの順番で述語に接続しても正しい表層形を出力するように実装されている（図３）。この図の左端の経路、すなわち、「**丁寧(ない(タ(述語(:食べる))))**」でも、右端の経路「**タ(ない(丁寧(述語(:食べる))))**」でも、「食べませんでした」が生成できる。

HB3 の実装で苦労したのは、述語の取り立てと敬語表現である。文法書には、取り立ての形式は記述されているが、HB3 では、元の形（取り立て前の形式）から取り立て後の形式を作り出す方法を実装しなければならない。

たとえば、「読んでいる」という述語には、次の２つの取り立て形式が存在する。

(1) 読んでもいる
(2) 読んでいもする

羽織文法では、直前の動詞にテ形を要求する機能語、たとえば、「（〜て）いる」や「（〜て）ほしい」を、テ助動詞とみなす。ほとんどのテ助動詞は、直前に取立助詞の挿入を許す。取立助詞が挿入された場合でも、その前の要素は依然としてテ形である。これを、「取立助詞は（後続要素の）接続型による要求を透過する」ことによって実現する。(1) の形式は、その具体例である。

これに対して(2)の形式は、「読む」を「読みもする」のように、一般の動詞に対して可能な取り立て形式である。この形式では、取立助詞の直前は連用形であるが、これを、「『する』が（取立助詞を透過して）動詞に連用形を要求する」とみなすのが、上記のテ助動詞の場合と整合する。それゆえに、この「する」を「取立助動詞する」とみなし、他の「する」と区別する。ちなみに、上記の(1)(2)を生成するためのブリックコードは、以下のとおりである。

(1) 取立も(テいる(述語(:読む)))

(2) 取立する(も，テいる(述語(:読む)))

取立助動詞には、この他に、イ形容詞型や判定詞型の述語を取り立てる「ある」、「なる」、「ない」(「美しくもある、確実でもある、美しくもなる、確実にもなる、美しくもない、確実でもない」）などがある。

敬語表現はよくわからない現象が多い。たとえば、「お食べに(も)なる」は、「食べる」に接頭辞「お」が付加されて「お食べだ（ナ形容詞)」となり、それに「取立助動詞なる」が付加された形とみなすことにした。そうすると「お食べください」は「お食べ（ナ形容詞語幹）-ください」と考えるのが自然だが、この「ください」は「お・ご〜」という形のナ形容詞語幹にしか接続しない。そのため、「食べる」から直接「お食べください」を作り出す「**お〜ください**」というブリックを定義することにした。なお、この「ください」は、動詞の「ください」や助動詞の「(〜て）ください」と区別し、「敬語接尾辞ください」とみなす。「いただく」、「なさる」、「いたす」なども同様である。

文法とは、一般に、「文の構成・整理、語が文を成す時の配列、語形変化に関する法則」と説明される。言語学では、どんな形式・規則性があり、それらがどんな意味を表すかに興味の主軸がある。

これに対して、**動く文法**である羽織文法(HB3)は、どんな形式・規則性があるかを追求する点は同じであるが、その形式を作り出す仕組みの実現に興味の主軸がある。「食べる」の連用形は「食べ」と説明するのが（動かない）文法だとすれば、「食べる」から「食べ」の形式を作り出すのが動く文法である。「ます」は動詞の連用形に接続すると説明するのが（動かない）文法だとすると、「食べる」に「ます」を接続すると、実際に「食べます」を作り出すのが動く文法である。

4. 文表示とブリックコード

日本語処理ツールを利用するプログラムを書いていると、それらの出力に対していささか懐疑的になる。たとえば、「書いていたにちがいありません」を Juman で形態素解析すると、図３のような結果が得られるが、素朴な疑問として、「こんなにたくさん情報は必要なのであろうか？」

「書いていたにちがいありません」を生成するためのブリックコード（の一例）は、以下のとおりである。

敬体ます(にちがいない(テンス有標(テ助動詞いる(述語(:書く)))))

```
書いて かいて 書く 動詞 * 子音動詞カ行 タ系連用テ形
いた いた いる 接尾辞 動詞性接尾辞 母音動詞 タ形
に に に 助詞 格助詞 * *
ちがい ちがい ちがう 動詞 * 子音動詞ワ行 基本連用形
あり あり ある 動詞 * 子音動詞ラ行 基本連用形
ませ ませ ます 接尾辞 動詞性接尾辞 動詞性接尾辞ます型 未然形
ん ん ぬ 助動詞 * 助動詞ぬ型 音便基本形
```

図 3　形態素解析結果の例

これを解析結果的に出現順に書き直すと、次のようになる。(述語は省略した)

書く，テ助動詞いる，テンス有標，にちがいない，敬体ます

この記述から表層文字列を生成（復元）できるということは、解析結果も、これでいいのではないかという考えが浮かんでくる。

文末述語解析器 Panzer[4]は、文から文末の述語を切り出し、HB3 のブリック列に変換するツールである。このツールは、いくつかのアプリケーション[5][6]で、文末を変形することが必要になり、それを実現するために実装した。

たとえば、上記の例で、「敬体ます」を削除すると、常体の「書いていたにちがいない」が生成できる。「にちがいない」を削除すると、モダリティを落とした「書いていました」が生成できる。あるいは、「にちがいない」を「だろう」に置換すると、「書いていたでしょう」が生成できる（図４）。このように、述語を、核となる述語とそれに続く機能語・複合辞の列として表現すると、文末変換が容易になる。

この容易さは、変換の対象としているブリック列が、本質的な情報のみから構成されていることに起因する。たとえば、「書く」の活用形がテ形であることは、後続要素が

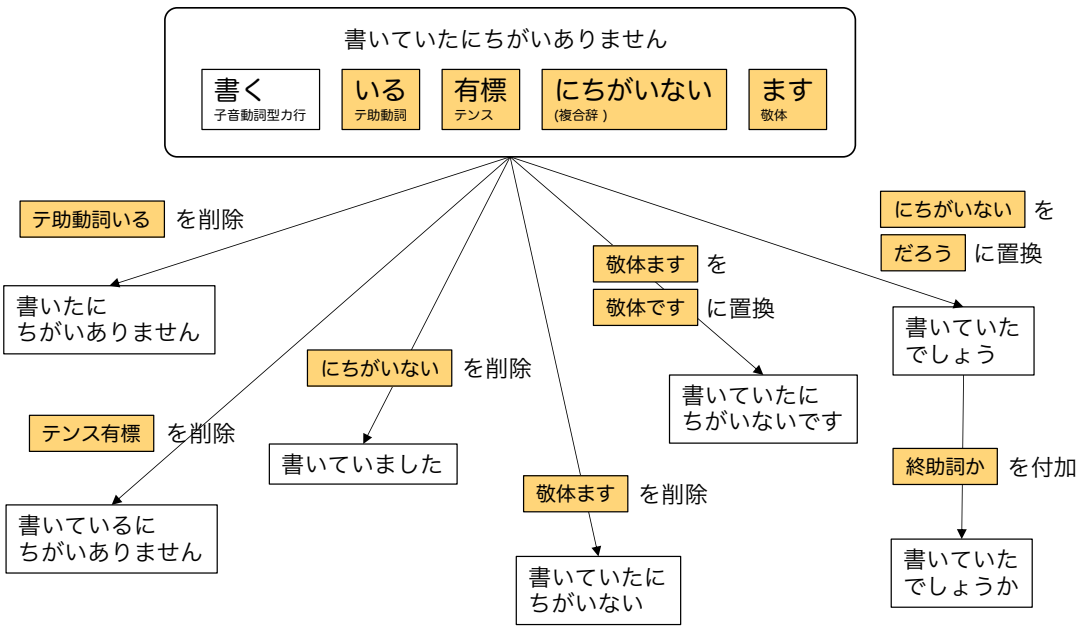

図 4 文末変換の具体例

「テ助動詞いる」であることから自動的に決まる。「テ助動詞いる」の活用形が夕形であることも、直後に「テンス有標」があることから自動的に決まる。つまり、これらの要素の活用形情報は冗長である。Juman の解析結果では「にちがいありません」は4形態素であるが、本質的な情報は、複合辞「にちがいない」の敬体（マス体）であることである。文を生成するためのコードは手で書かなければならないので簡潔に書けることが必須条件となるが、解析結果も、必要最小限のコンパクトな記述形式の方が、その後の処理が容易となる。

言語処理では、解析にしても生成にしても、何からの（文の）内部表現（＝文表示）を採用する。「生成できる文は解析でき、解析できる文は生成できる」。このあたり前のことを実現するためには、文の内部表現は、解析と生成で同一の形式であることが望ましい。そのためには、機能語・複合辞の規格化（標準リストの作成）が不可欠と考える。機能語を完全に列挙するという羽織文法の方針は、このような考え方の現れでもある。

5. おわりに

私は、すでに HB3 で 1 万字以上の文章を書いており、任意の文を合成できるソフトウェアシステムの実現という当初の目標は、書き言葉に対しては、ほぼ達成できたと考えている。HB3 が現在の形となるまでに 6 年を要したが、最終的には「動く文法（羽織文法）」として収束した。

動く文法の最大のメリットは、実際に文を合成することによって、文法の完全性と整合性をチェックすることができる点にある。頭の中で考えるだけでなく、目に見える形で規則に瑕疵がないかどうかを確かめることができることは、文法をより完全なものに近づけていくための強力な武器となる。加えて、規則性の捉え方に複数の選択肢がある場合に、実装（プログラムによる実現）がシンプルになる方が望ましいという選択基準を持ち込める。これは文法体系に、ひとつの評価軸を与えるものである。

文献

[1] 佐藤理史. コンピュータが小説を書く日 －AI 作家に「賞」は取れるか. 日本経済新聞出版社, 2016.

[2] 佐藤理史. HaoriBricks3: 日本語文を合成するためのドメイン特化言語. 自然言語処理, Vol. 27, No.2, pp411-444, 2020.

[3] 益岡隆志, 田窪行則. 基礎日本語文法－改訂版－. くろしお出版, 1992.

[4] 佐野正裕, 佐藤理史, 宮田玲. 文末述語における機能表現検出と文間接続関係推定への応用. 言語処理学会第 26 回年次大会, B6-3, pp1483-1486, 2020.

[5] 平良裕汰朗, 佐藤理史, 宮田玲, 今頭伸嘉. ダイレクト広告コピー文の分析と自動生成. 言語処理学会 第 25 回年次大会 発表論文集, P1-26, pp406-409, 2019.

[6] 柳将吾, 佐藤理史. ウィキペディアから抽出した人物エピソードの話し言葉への変換. 言語処理学会第 26 回年次大会, C2-3, pp437-440, 2020.