料理レシピテキストの構造化のための形式言語の提案

Xuezhe ZHANG† 王 元元† 河合由起子††,†††

† 山口大学工学部知能情報工学科 〒 755–8611 山口県宇部市常盤台 2-16-1
†† 京都産業大学情報理工学部 〒 603–8555 京都府京都市北区上賀茂本山
††† 大阪大学サイバーメディアセンター 〒 567–0047 大阪府茨木市美穂ヶ丘 5 番 1 号
E-mail: †{g039ff, y.wang}@yamaguchi-u.ac.jp ††kawai@cc.kyoto-su.ac.jp

あらまし 近年では，楽天レシピやクックパッドが代表となるユーザ投稿型レシピサイトの流行により，これらのサイトのデータを利用した研究が多くなっている．しかしながら，ユーザ投稿型レシピの研究では，前処理として複雑な自然言語処理が必要となる．また，IoT (Internet of Things) の普及により，ユーザが投稿したレシピを IoT に利用する研究も多くなっている．その代表例として，IoT に向けて機械が読めるレシピ MRR (Machine Readable Recipe) の研究があるが，これも自然言語をデータ構造に解析したものであり，同様に複雑な自然言語処理を行うことが必要である．そこで本研究では，料理レシピテキストの構造化のための形式言語を提案し，レシピ研究に必要なレシピの解析の簡略化を目的として，自然言語のレシピをその形式言語への変換ツールを開発する．本論文では，自然言語から形式言語への変換ツールの原理について述べ，開発した変換ツールの有効性を検証する．

キーワード レシピ分析，自然言語処理，形式言語，データ構造，手順テキスト構造化

1. はじめに

近年では，楽天レシピやクックパッドが代表となるユーザ投稿型レシピサイトの流行により，これらのサイトのデータを利用した研究が多くなっている．投稿型レシピサイトのレシピは基本的に，タイトル，材料リストなどのメタデータ，調理手順と画像，ユーザコメントで構成されている．

レシピ関係の研究の多く，特に自然言語レシピデータを用いた研究の典型的な処理過程として，前処理・研究の処理・結果を出力で構成されることが多い．自然言語処理技術の向上により，さまざまな自然言語処理ツールが開発されたが，レシピに向いている自然言語処理ツールが少なく，自然言語のレシピを研究するために，自然言語をレシピ構造への処理が手間がかかる問題がある．そのため，これらの研究の問題点として，まず，データ本来には，一意性の問題があるため，同じ文でも違う理解が出る可能性がある．レシピの構造化と無関係な研究で，自然言語レシピを構造化しないといけないため，不必要な手間がかかる．また，その解析方法の精度により，研究結果への影響を及ぼすことも考えられる．レシピを構造化にするため，研究ごと独自な方法で解析することが多いことがあげられる．これらの問題点により，研究に使用した元データと研究の処理で実際に処理したレシピ構造に偏りが生じ，研究結果の精度にも影響を及ぼす可能性は考えられる．

レシピの研究の中では，近年の IoT 家電の普及により，特にレシピデータと IoT を結合した研究は多くなっている．その中の代表例として，MRR (Machine Readable Recipe) の研究がある．MRR とは，機械が読めるレシピで，木やグラフなどで記述することが多い．しかし，その中も自然言語からデータ構造への解析研究が多く，そのため，自然言語と同様な問題がある．

これらの問題点の解決とレシピ研究の前処理の簡略化を目的として，レシピの手順文の構造化記述のための形式言語を提案した．この形式言語の位置づけとして，自然言語のレシピを機械が処理できるデータ構造に変換するためのものである．自然言語は人間に適し，データ構造は機械に適することに対し，形式言語の利点として，以下があげられる．

- 一意性がある
- 機械が処理しやすいため，レシピ解析の簡略化ができる
- 記述，修正，保存が容易
- 他の形式への変換が容易

本研究では，料理レシピの手順文を構造化記述のための形式言語の実現を目的として，レシピのための形式言語を提案する．また，「楽天公開データ」(注1) より，楽天レシピデータ [1] と自然言語処理技術を利用し，レシピ研究に必要なレシピの解析の簡略化を目的として，自然言語のレシピをその形式言語への変換ツールを開発する．

本論文の構成は以下のとおりである．次章では，自然言語処理，レシピ構造化のためのフローグラフ，MRR などの関連研究についてを紹介し，本研究の新規性について説明する．3 章では，具体的に提案した文法・仕様・ツールの原理を述べる．4 章では，ツールの出力結果，ツールの精度の評価実験およびその結果と考察について述べる．最後に，まとめと今後の課題について述べる．

2. 関連研究

2.1 レシピデータの前処理

近年，自然言語処理技術の向上により，多種多様な自然言語解析ツールも開発された．また，英語の解析より困難な日本語の

(注1)：https://www.nii.ac.jp/dsc/idr/rakuten/rakuten.html

自然言語解析ツールも多く開発された．その中の代表例として，MeCab [2]，CaboCha [3] と GiNZA [4] などがある．近年新たに開発された日本語解析ツールとして，GiNZA が有名である．GiNZA とは Megagon Labs が開発した OSS の SudachiPy と spaCy を活用した新たな日本語解析ツールである．2019 年に論文の発表 [5] とともに GitHub でリリースされた．

ユーザ投稿型レシピサイトの流行により，近年，レシピデータと自然言語処理を結合した研究も多くなっている．その中で，自然言語レシピからレシピに関する知識，属性の抽出の研究がある．山肩ら [6] が研究したレシピ用語の定義とその自動認識のためのタグ付与コーパスの構築では，レシピの調理手順に対しての用語抽出を目的として，レシピ中に出現する需要な用語を定義し，実際にコーパスに対してアノテーションし，実用的な精度の自動認識器を構築した．また，この論文で構築したレシピ用語の自動認識ツールは，レシピ言語処理マニュアル [7] としては公開され，このツールを使用することで自然言語レシピのレシピ用語の属性解析ができる．本研究では，この論文で構築したレシピ用語自動認識ツールを利用し，レシピ用語の属性を判定する．レシピ言語処理マニュアルでは，指定の書式へ変換するツールを提供したが，本研究では，形態素解析は GiNZA を利用したため，提供されたツールを使用せず独自で指定された書式を作成する．また，このツールが未対応な単語に対し，補足辞書を作成することで対応する．

レシピと自然言語処理を結合した研究の中では，自然言語のレシピテキストをレシピ構造に解析する研究もある．本研究でも，自然言語のレシピテキストから形式言語に変換するため，自然言語のレシピテキストの構造を解析する必要がある．レシピ構造の解析の代表例として，京都大学が開発したレシピの言語処理ツール (注2) がある．森ら [8] はレシピ構造を表示するために，手順文書をフローグラフとして表現する形式を，レシピテキストに応用してコーパスの作成を試みている．前田ら [9] が自然言語レシピからフローグラフへの解析も実現した手順文書からの意味構造抽出の研究を行った．また，苅米ら [10] が提案した料理レシピテキストの構造解析とその応用や浜田ら [11] が提案した料理教材における手順の構造化もあげられる．しかし，これらレシピ構造解析の研究の問題点として，自然言語は一意性の問題がある．そのため，自然言語から解析した構造は精度が高くても，自然言語本来の意味と偏りが生じることが可能である．本研究では，このような偏りを防ぐため，プログラミング言語のような一意性のある形式言語を提案する．また，自然言語から解析した構造の変更が困難である．入力となる自然言語を変更しても，解析された構造は思う通りになる限りではない．しかし形式言語を解析した構造は唯一のため，入力が形式言語だと，解析した構造の変更が容易になる．

2.2 レシピデータの応用

近年，IoT 家電の普及により，IoT へ投稿型レシピサイトのデータを応用する研究も多くなっている．その代表例として，クックパッド株式会社が新事業への投資として開発した MRR [12] がある．この MRR の資料も公開されている [13]．MRR とは，機械が読めるレシピという意味で，IoT 家電の問題点である，IoT 会社が書いた専用レシピでした使えないことの解決を目的とした研究である．クックパッドがレシピをさまざまな調理家電とつなぐ「Oicy」というサービスを提供し，このサービスに同社が開発した MRR を利用した．このサービスによって，クックパッドのレシピ内容に合わせて，機械を自動制御することが可能になる．

IoT や調理ロボットへのレシピの応用として，ロボット用のレシピの開発の研究もある．たとえば，滝ら [14] が研究した作業分析によるロボット料理レシピの作成手法がある．この研究では料理ロボットシステムに料理知識を作業として与えるために，調理工学の調理操作論にある作業分類と生産管理における作業分析を基に，人間の行う料理作業および料理レシピからロボットシステムのためのロボット料理レシピを作成する方法を提案した．この研究によって，ロボットが処理可能なレシピの作成ができるが，問題点として，この手法で作成された料理作業は厳密に手順などを表すため，複雑な表の形式で手順を表現している．そのため，一般人が作成するのが困難である．また，調理作業の内容の修正が難しいことと，人間の可読性も問題点としてあげられる．本研究で提案するレシピのための形式言語では，機械可読だけでなく，人間の読みやすさと修正しやすさも考慮したうえ，プログラミング言語と類似する形式言語を提案する．

1000000048	1	しじみを砂抜きしておきます。
1000000048	2	水からしじみをいれ火にかけます
1000000048	3	沸騰したらあくをとります
1000000048	4	お味噌を入れ味をみて調整します。

図 1　楽天レシピデータの例

表 1　入力例と期待する変換結果

原文	期待する変換結果
水からしじみをいれ火にかけます	火を掛ける (入れる (水, しじみ))
沸騰したらあくをとります	取る (あく,condition_until=沸騰 ())

3. 料理レシピテキストの形式言語

本研究では，既存の関連研究を参考し，レシピのための形式言語の文法を定義し，自然言語からその形式言語への変換ツールを開発する．

3.1 料理レシピテキスト

通常，料理レシピテキストのデータはタイトル，紹介文，材料リストなどメタデータ，手順文，他のユーザの試作レポートやコメントで構成されることが多い．今回使用した楽天レシピデータの手順文のデータは図 1 が示すように，レシピ ID，手順位置，手順作り方で構成される．

本研究において，文単位に自然言語レシピの手順文を変換するため，手順作り方を扱う対象となる．本研究の目的として，図 1 が示した自然言語のレシピ手順文を表すために，表 1 のよ

(注2)：http://www.lsta.media.kyoto-u.ac.jp/how-to/recipe-NLP/

表 2　定義で使用された EBNF 表記

記号	意味
::=	定義
(...)	グループ化
—	区切り
[...]	オプション
{...}	繰り返し
"..."	二重引用符
*	0 回以上

うに，一定な規則で定めた形式言語を提案し，その形式言語への変換ツールを作成する．

3.2　形式言語の概要

自然言語のレシピ手順文を厳密かつ一意性を持つようにするため，手順文を形式言語で記述する方法がある．形式言語とは，文法が場合によっては意味も形式的に与えられている言語である．意味が曖昧な自然言語に対し，一部の人工言語や，いわゆる機械可読なドキュメント類などは形式言語である [15].

形式言語を定義するために，まずその文法を定義しなければならない．そのため，ドメイン固有言語（DSL: domain-specific language）を実現するのが一般的である．しかし，DSL の主な問題点として，利用者に余計な学習コストが必要であること．それを防ぐため，できる限り既存なプログラミング言語と近似な文法を採用した．基本的に，Python における関数の呼び出しの文法 [16] を参考にして文法を作成した．プログラミングの経験者が簡単に理解できるという利点がある．

また，このような形式言語を定義するうえに，本研究では，自然言語のレシピを提案した形式言語への変換ツールを開発した．このツールは，3.1 節の表 1 が示したような変換ができる．

3.3　形式言語の定義と説明

形式言語を定義するために，BNF 記法と EBNF 記法がよく用いられている．本研究では，比較的に複雑な文法を簡単に表現できる EBNF 記法を採用した．EBNF (Extended Backus–Naur Form) とは，文脈自由文法を表現するメタ文法記法であり，コンピュータのプログラミング言語や形式言語の形式的表現として使われる [17].

今回の定義で使用された EBNF の表記は表 2 のように示す．本研究で提案した形式言語の文法は以下の EBNF で定義する．

```
program ::= (comment | expression_sentence) (NEWLINE (
    comment | expression_sentence))*
character ::= <any Unicode character except control
    characters>
id_character ::= <any Japanese or English character or
    number>
identifier ::= id_character*
comment ::= "#" character*
expression_sentence ::= expression [comment]
expression ::= identifier "(" [arguments] ")"
arguments ::= (object_arguments ["," keyword_arguments
    ])|[keyword_arguments]
object_argument ::= identifier | expression
object_arguments ::= object_arguments (","
    object_arguments)*
keyword_argument ::= identifier "=" (identifier | expression
    )
keyword_arguments ::= keyword_arguments (","
    keyword_arguments)*
```

1 つのレシピを 1 つのプログラムとして考える．1 つのプログラム（レシピ）は，複数の文で構成され，改行で文を区分する．文は「#」から始めるコメント文と，動作を表すステップ文にわかれている．

- コメント文は，実行しない文として，ステップの注釈などに利用できる．
- ステップ文は，ステップとオプションのコメント文で構成される．ステップは,「関数名 (関数の引数)」で構成され，基本的に,「切る (豚肉)」のような「動作 (動作の対象)」の形になる．
- 関数の引数は，関数の実行対象や，関数のプロパティーがある．複数の引数は「,」で区分し，関数の実行対象である対象引数と関数のプロパティーと，キーワードでプロパティー名を定めるキーワード引数にわかれている．たとえば,「トマトをくし形に切る」を「切る (トマト,shape=くし形)」に変換すると,「トマト」は対象引数になり,「くし形」はキーワード引数に当てはまる．全てのキーワード引数は，全ての対象引数の後ろの置く必要がある．
- 対象引数は，対象の名前，またはステップ文である．キーワード引数は,「プロパティー名=プロパティー値」の形で，プロパティー名はどんなプロパティーを指定するもので，上例だと,「shape」はプロパティー名になる．プロパティー値は対象引数と同様，対象の名前，またはステップ文である．
- プロパティー名は，ユーザの需要に応じで，自由に指定できるが，本研究で開発したツールにおいて，表 3 のプロパティーを利用した．

表 3　プロパティの基本例

プロパティー名	意味
tool	使用器具
time	時間
quantity	量
food_state	食品状態
tool_state	器具状態
extent	程度
condition_time	動作実行回数
condition_until	動作完成条件（XXX たら，XXX まで）
condition_before	動作は条件前実行（XXX のうち，XXX 前）

全てのトークン間では，ホワイトスペースは任意である．コメント文で使用可能な文字は，Unicode で改行などの制御文字を除いた任意文字で，他の名前で使用可能な文字は日本語文字，英文字，数字である．また，関数の引数は必須ではない，引数

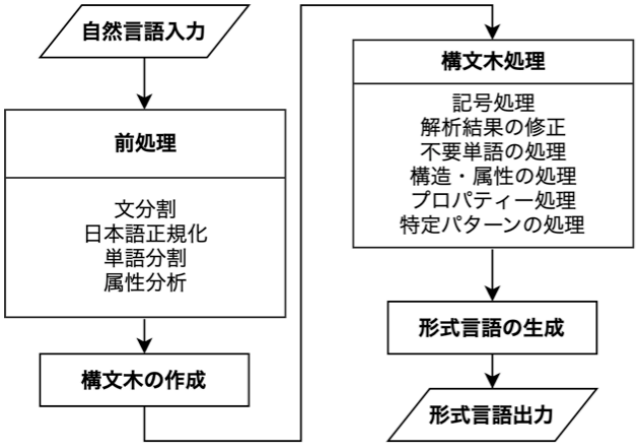

図 2　ツールの処理過程

のない関数や，対象引数またはキーワード引数だけの関数も使用可能である．

この形式言語は，HTML や XML のような言語で，レシピ手順の構造だけを表し，使用方法は使用するユーザ独自に決める．たとえば，レシピ関係の研究に使うことや，調理ロボットへ応用することなどができる．また，本文法は，Python と近似する文法を利用したため，Parser を開発せず，関数を定義すると直接 Python スクリプトとして使用が可能である．また，文法では引数は文字列に指定しなかったため，ラムダ式などで活用できる．Python スクリプトとして直接使用できるほか，もし形式言語からレシピ構造を解析する必要があれば，Parser の開発もできる．自然言語やレシピ構造の解析ライブラリーと比べ，以下の利点がある．

- 原文に一意性があるため，形式言語を解析した結果となる構造は，必ず原文の意味と一致する．
- 文法を定めたため，Parser の開発は自然言語解析より容易である．
- プログラミングができれば，レシピや自然言語に関する専門的知識がなくても独自で Parser の開発ができる．

3.4　自然言語から形式言語の生成ツール

本研究では，レシピのための形式言語を定義したうえ，レシピ研究に必要なレシピの解析の簡略化を目的として，自然言語のレシピをその形式言語への変換ツールを開発した．ツールの開発言語は Python である．このツールの処理過程は，図 2 にフローチャートとして示す．

各過程の処理について，本節で詳しく述べる．

3.4.1　入力，前処理と構文木の作成

入力として，ファイルからまたは標準入力からの入力ができる．前処理では，以降の処理の準備段階として，入力された自然言語のレシピの手順文を構文木の生成に適した形式に整形し，また自然言語文から以降の分析に必要な情報を抽出する．入力を文ごとにわける処理は 2 つのモードがあり，改行と句点だけで文を区分するモードと，読点や「そして」など，文をさらに短く分割するモードがある．

次の処理として，日本語の正規化である．最後の結果に原文を反映するために，この処理は分割された文を直接処理するわけでなく，別の変数に結果を保存した．この処理において，まず構文木の処理を簡単にするために，日本語の「ですます調」を「である調」に変換した．変換の方法は変換用の辞書を利用した．「である調」に変換した後，記号を統一し，手順に不要な単語を取り除いた．次に，手順文の番号を正規表現を利用して取り除いた．そして，2.1 節で述べた自然言語解析ツールの GiNZA を用いてレシピ用語解析ツールの属性解析ツールに適した書式に変換し，臨時ファイルに保存した．属性解析ツールの書式は，空白で文を分割したもので，以下の例のようなものである．

```
水 から しじみ を いれ 火 に かける
沸騰 し たら あく を とる
```

また，GiNZA が処理できない固有名詞があるため，GiNZA が使用した Sudachi ライブラリーのユーザ辞書を以下のように作成した．

```
黄金の味,4786,4786,5000,黄金の味,名詞,固有名詞,一般,*,*,*,
    オウゴンノアジ,黄金の味,*,*,*,*,*
万能ネギ,4786,4786,5000,万能ネギ,名詞,固有名詞,一般,*,*,*,
    バンノウネギ,万能ネギ,*,*,*,*,*
……
```

次に，属性解析ツールを用いて文の単語ごとの属性を取得する．出力は以下のようになる．

```
水/F−B から/O しじみ/F−B を/O いれ/Ac−B 火/T−B
    に/O かける/Ac−B
沸騰/Af−B し/O たら/O あく/F−B を/O とる/Ac−B
```

出力ファイルの書式として，分割した単語は「単語/属性+空白」の形になる．属性タグは表 4 に示したものがある．また，このツールでは識別不能や間違いが出る可能性があるため，補足辞書を作成した．属性辞書では表 5 に示したものがある．この手順で得た出力ファイルから，書式に沿って単語ごとの属性を取得する．この属性は，単語ノードの情報として，構文木の生成に使われる．

前例の「沸騰したらあくをとる」に対して，図 3 のような構文木を得られる．このように，GiNZA の解析より得た構文木は基本的に，動作が頂点になり，その動作を修飾する対象や条

表 4　属性タグと意味

タグ	意味
F	食材
T	道具
Ac	調理者の動作
Af	食材の変化
Sf	食材の様態
St	道具の様態
D	継続時間
Q	分量

表 5　補足辞書の単語例

単語	英語タグ	日本語タグ
たら	condition_until	条件-まで
少し	extent	程度
ポテトマッシャー	T	道具
……	……	……

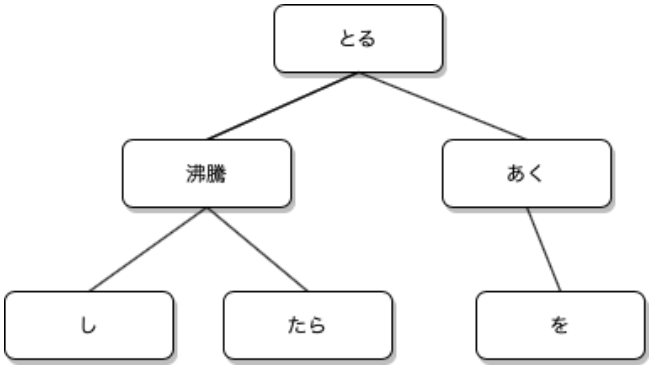

図 3　GiNZA の構文木例

件などがその動作の子ノードになる．特にレシピの手順文は，基本的に動作で構成されているため，ほどんとの文の構文木はこのような形になる．そして，次の処理のため．まず構文木のノードを格納するためのノードクラスを作成した．このクラスは，単語の原文，単語が動詞や形容詞の場合ならその原型，固有表現抽出の結果，原文での単語順番，属性分析で得た属性，子ノードや親ノード，また子を追加するなどのメソッドがある．GiNZA で解析した結果では，固有表現であっても，1 つの単語が複数に分割され，複雑な階層構造になることがあるため，このようなノードをマークした．特に，同じ階層にあるべき食材が複数の階層になることや,「30 分」など固有表現が,「30」「分」などわかれた形に分割されたなどの場合の処理に必要である．

3.4.2　構文木の処理

自然言語から得た構文木を，形式言語への変換に適した構文木に変換するため，木の全てのノードに対し，名詞や動詞など必要な単語だけを残す．

構文木の処理では，投稿型レシピサイトでは複数の調味料を，特殊記号や「A，B，C」「ア，イ，ウ」などでまとめて表記することがある．このような場合の処理として，構文木分析の手順では，変数として入れた調味料リストからこのような記号を探し，もしあったらその記号を調味料として，seasoning というノードの子ノードに作り直す（以下，A を B の子ノードに作り直すことを A を B としてマークすると称する）．それ以外は,「unknown-symbol」としてマークし，このような属性が判定できない単語を削除する．

1 つの文で，1 つの動詞がルートにならない場合があるため，このような構文木をチェックし，最初の動詞がルートになるように作り直す．そして，全てのノードをそのレシピ用語の属性としてマークする．また，動詞に動作の属性を付加し，形容詞に食材状態の属性を付加した．マークするとき，属性は表 6 に示したルールで適用した．次に,「A をしたら，B する」や,「A のうちに B する」などのパターンを適した形に処理する．表 7 の単語をパターンとして表記した．最後に，文法を生成後，文法間違いが起こらないように，重複した属性をまとめる．

表 6　属性タグのノード名

属性タグ	ノード名
F	food
T	tool
D	time
Q	quantity
Ac	user_action
Af	food_action
Sf	food_state
St	tool_state
他	そのまま

表 7　単語のパターン例

単語	パターン
まで	condition_until
たら	condition_until
なら	condition_until
一度	condition_time
さらに	flow_then
うちに	condition_before

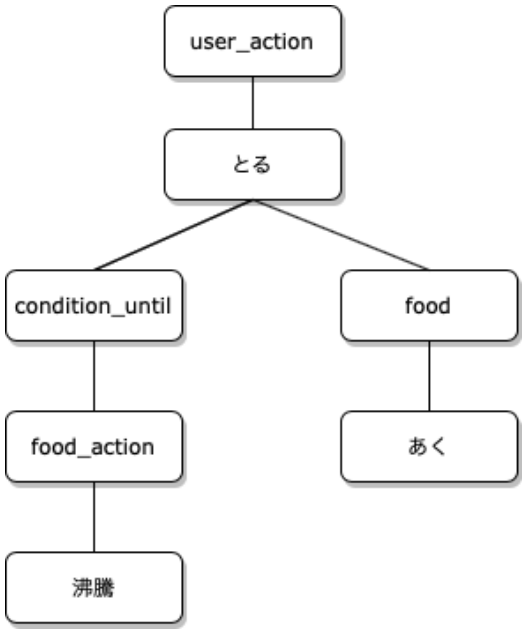

図 4　処理した構文木例

以上の処理を経て，この構文木を形式言語に変換する．もし変換が失敗して，結果に動作がない場合，結果を「unknown_function」の対象として，解析不能としてこの結果を返す．以上の処理を経て，3.4.1 節で例としてあげられた「沸騰したらあくをとる」の構文木は図 4 に示す．

3.4.3　形式言語の生成

以上の処理を経て，構文木の不要な部分が取り除かれ，単語がその属性としてマークされたため，この処理後の構文木を用いて形式言語を生成することが可能である．ルートノードを

変換することで，構文木の形式言語化ができる．前例の「沸騰したらあくをとります」は「取る (あく,condition_until=沸騰())」に変換され,「沸騰したら取るという動作をして，取る対象はあくである」という意味の形式言語になる．

4. 評価実験

3.4 節では，自然言語料理レシピの手順文を 3.3 節で提案した形式言語に変換するツールの原理について述べた．本章では，このツールの精度を評価する．また，このツールは自然言語処理ライブラリーと比べ，レシピ手順文に対する解析の改善効果を評価する．

4.1 自然言語から形式言語の生成例

自然言語ライブラリーを用いてレシピ手順文を解析し，生成された構文木を直接形式言語に生成する場合（以下,「改善前」と記す）,「トマトをくし形に切る」の変換結果は以下のようになる．

```
# トマトをくし形に切る
切る (トマト,food_state=くし形)
```

文が短いかつ条件などを含まないような入力であれば正確な出力になる．しかし,「沸騰したら塩を入れる」と「ポテトマッシャーでポテトをつぶす」の出力は以下のようになる．

```
# 沸騰したら塩を入れる
入れる (沸騰 (),塩)

# ポテトマッシャーでポテトをつぶす
潰す (ポテト)
```

このような条件が含まれる文だと，その条件をうまく解析できず，また，複雑な固有名詞がある場合，解析もできない．3.4 節で述べたツールを用いてレシピ手順文を解析し，形式言語を生成する場合（以下,「改善後」と記す），以下のような結果になる．

```
# トマトをくし形に切る
切る (トマト,food_state=くし形)

# 沸騰したら塩を入れる
入れる (塩,condition_until=沸騰 ())

# ポテトマッシャーでポテトをつぶす
潰す (ポテト,tool=ポテトマッシャー)
```

改善前と同じく，単純な入力の場合は正確な出力になる．そのうえに，辞書の作成によって複雑な固有名詞が処理可能になり，また条件もパターンごとの対応により処理可能になった．

4.2 実験概要

4.2.1 データセット作成

まず，楽天レシピデータ [1] から無作為で手順データを選出した．楽天レシピデータでは，レシピ手順が属するレシピ番号，レシピ手順のテキスト，レシピ手順の番号などがあるが，今回はレシピ手順のテキストのみを分析した．

選出されたデータから，手順文以外のヒントなどの文を取り除く，そのデータを句読点と「そして」で分割し，分割前の文から 50 件のデータ（以下,「長文」と記す）を選出し，また分割後の文から長文と重複のない 50 件のデータ（以下,「短文」と記す）を選出した．長文と短文のデータをそれぞれ改善前のツールと改善後のツールを用いて形式言語に変換した．

4.2.2 評価方法

ツールの適合率，再現率と F 値を計算するために，生成結果の正確単語数を測る必要がある．そのため，生成結果の単語ごとの正確性を判断する必要がある．

4.2.1 節で生成されたデータを Google フォームを利用し，結果の単語ごとの正確性を判断するためのアンケートを改善前と改善後それぞれ作成した．アンケートでは改善前と改善後共に合計 100 問あり，前 50 問は長文の判断で，その後の 50 問は短文の判断である．

10 名の 20 代の情報系大学生にアンケート調査を依頼した．これらの被験者は，男性 8 名，女性 2 名であり，また全員プログラミングの経験がある．プログラミングレベルに対して,「得意」は 1 名,「普通」は 7 名,「苦手」は 2 名である．また，料理レベルに対して,「普通」は 7 名,「苦手」は 3 名である．被験者全員に，3.3 節で提案した形式言語について文法を説明し，また，4.2.3 節で述べる評価基準について説明した．被験者を半数ずつの両グループ A と B にわけ，グループ A に対し，改善前のアンケートの調査を行い，グループ B に対し，改善後のアンケートの調査を行った．

4.2.3 評価基準と結果

生成結果の単語ごとの正確性を判断するために，いくつかの判断基準を定めた．

正確性の判断の基本的な考え方として，この生成結果を実行すれば，原文と同じ動作をするかどうかである．何に基づいて単語の正確性を判断するかについて，まず文全体に文法間違いなどがあるかどうかを判断する．そして，単語は，原文にあるかどうか，間違えて動作やプロパティーを識別したかで判断する．そしてプロパティーは，値は原文にあるかどうか，またプロパティー名とっているかどうかで判断する．

判断の順番について，プログラミング言語の実行順は一般的に括弧の内側から外側への実行のため，判断時も同様に一番内側から正確性を判断する．また，括弧の外側の間違いは，括弧の内側に影響がないが，括弧の内側に間違いがあると括弧の外側も間違いとする．また，原文にプロパティーなど必要な単語を失った場合，その必要な単語の出現すべき位置の外側も間違いとする．

判断が難しい場合，被験者は独自基準で判断してもらうが，全ての問題に対して同じ基準で判断する必要がある．表記が違うが，意味は同じな単語は正確とする．たとえば，仮名と漢字の違い，文法が違うが同じ意味を表す文，動詞の変形，類語などがあげられる．動作を表さない単語やヒントを表す単語は，不要な単語としてすべて間違いとする．動作の判断を行うとき，

表 8　ツール精度の評価実験の結果

データ	適合率	再現率	F 値
改善前（全体）	53.81%	50.31%	52.00%
改善前（長文）	50.18%	46.84%	48.45%
改善前（短文）	60.00%	56.25%	58.06%
改善後（全体）	**74.44%**	**69.60%**	**71.94%**
改善後（長文）	70.25%	65.12%	67.59%
改善後（短文）	**81.44%**	**77.27%**	**79.30%**

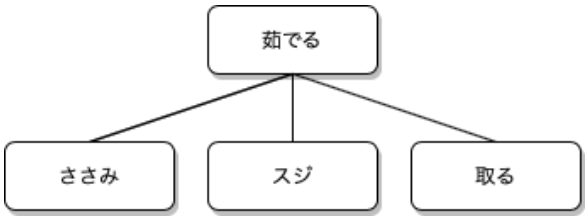

図 5　間違った構文木例

その引数の順番，関係も判断する必要がある．たとえば，動作の実行順や状態，条件を表す文を対象として識別した間違いなども判断すべき.「unknown_function」が出現するなど，解析失敗を表す単語がある結果，また文法が間違った結果では，その結果ごと全部間違いとする．文法の間違いとして，属性の重複，対象とプロパティーの順番間違い，文法ルールに沿わない結果などがある．また，意味不明な文字も間違いとする．

評価基準は適合率と再現率とその調和平均である F 値を用いて，原文と変換結果の単語の数で算出した．原文で出現した単語の数を N_{ref}，変換結果で出現したトークンの数を N_{sys}，過半数の人が正確だと判断した単語の数を N_{int} とし，以下の式で適合率，再現率と F 値を算出する．

$$\text{適合率} = \frac{N_{int}}{N_{sys}}$$

$$\text{再現率} = \frac{N_{int}}{N_{ref}}$$

$$F\text{ 値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}}$$

ツール精度の評価実験の結果は表 8 に示す．その結果は，以下のとおりである．

- 改善後全体の適合率，再現率，F 値とともに改善前よりあがった．
  - 適合率は改善前により 20.63%あがった
  - 再現率は改善前により 19.29%あがった
  - F 値は改善前により 19.93%あがった
- 改善前も改善後も，長文より，短文の適合率，再現率，F 値が高い結果となった．
  - 改善前では，長文は短文より適合率は 9.82%高い，再現率は 9.41%高い，F 値は 9.61%高い
  - 改善後では，長文は短文より適合率は 11.19%高い，再現率は 12.16%高い，F 値は 11.71%高い

4.3　考　　察

4.2.3 節に示したツール精度の評価結果では，全体的に長文の精度は短文より 10%ほど低いことがわかった．その理由として，長文だと，複雑な状況や条件が含まれ，また複数な動作が

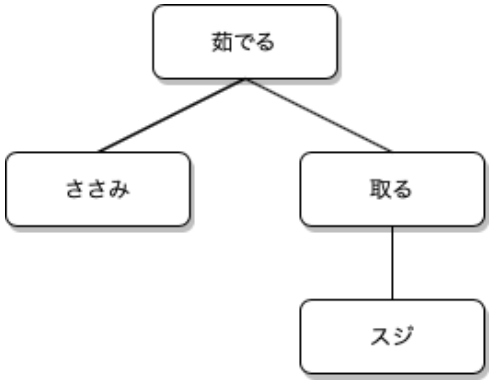

図 6　正しい構文木例

含まれることが考えられる．自然言語解析で得られた構文木も複雑になり，対応していないパターンが出る可能性があるため，長文を短く分割することで精度の改善ができると考えられる．また，改善前の精度は 52.00%であることがわかった．改善前では，構文木を直接形式言語に変換したため，構文木に対する処理が不十分である．不必要な単語の削除や特定パターンの識別と処理が足りないことが理由として考えられる．さらに，改善後の精度は 71.94%で，改善前より 19.93%精度があがった．その理由として，構文木に対する処理を追加し，単語の必要性の判断，パターンとユーザ辞書の追加や文法の正確性のためのチェックなどを追加したことが理由として考えられる．しかし，この改善後の精度も高くないため，間違った解析結果に対する考察により，以下のような原因が考えられる．

まず，利用している自然言語解析ツールと属性分析ツールの精度が 100%ではないため，変換ツールの精度もこの 2 つのツールの低い方の精度が上限になる.「材料を準備」という入力を考える場合，出力は以下になる．

```
# 材料を準備
unknown_function(材料)
```

しかし，正しくは「準備 (材料)」になるべきである．この間違いが出た原因として，利用したツールの両方は「準備」を動詞または動作として識別できなかったため，準備は処理の過程中に不要だと判断され，変換結果が「材料」だけになり，動作がないため「unknown_function」になった．

また，入力の問題も考えられる．入力が自然言語のため，表現が曖昧で一意性がないや文法が厳密でない問題がある．たとえば，入力が「ささみはスジを取り茹でる.」として変換した場合，出力は以下になる．

```
# ささみはスジを取り茹でる.
茹でる (ささみ,スジ,取る ())
```

しかし,「取る」という動作には，対象である「スジ」がなくなり，間違いになっている．正しくは「茹でる (ささみ, 取る (スジ))」である．この間違いが出た原因として，原文の「取り茹でる」は曖昧で，2 つの動作を 1 つの動作のように記述したため，自然言語解析ツールはこの文の「取り茹でる」を連語とし

て解釈し,「取り」を「茹でる」を修飾するものに識別した．そのため，図 5 のような間違った構文木になった．正しくは，図 6 のような構文木になるべきである．この間違いに対して，入力の「取り茹でる」を「取り，茹でる」や「取って茹でる」に変更することで修正ができるが，このような単語が連語か 2 つの動作かはプログラムによる判断が困難である．また，自然言語の形は無限になるため，このツールが対応していないパターンがあるため，より多くのテストをし，パターン処理の改善も必要である．

以上の考察より，今後の課題として，以下のことがあげられる．まずは精度の向上である．現在では，改善後の精度でも 71.94%で，特に長文に対して精度は 67.59%しかなく，実用化にはまだ精度が不足である．精度の向上の方法として，辞書の単語数を増やす，構文木の処理で処理できるパターンの数を増やすことがあげられる．そして，現在のツールは動詞や形容詞を原型に直せるが，食材や分量の正規化ができない．IoT や調理ロボットなどへの応用のために，このような表記ゆれの統一も今後の課題として考えられる．今回では，自然言語からの変換ツールを作成したが，今後はこの形式言語を利用して，他の基本ツールも作成ができる．たとえば，形式言語から解析した構造を用いてレシピのフローチャートの作成や，レシピをほかの表現への変換，要約や翻訳などが考えられる．

5. おわりに

本研究では，料理レシピ手順文を構造化記述のための形式言語の実現を目的とした形式言語の文法と仕様について提案し，そして自然言語からその形式言語へ変換するためのツールを開発した．評価実験では，楽天レシピデータを用いて自然言語から形式言語への変換ツールで自然言語を形式言語に変換し，その変換精度の検証を行い，長文より短文の精度が高い結果であることが確認できた．また，改善後の変換精度は 71.94%であり，提案した変換ツールの有効性を確認した．

今後の課題として，辞書やパターンを増やすことによりツール精度の向上，機械学習などの手法により精度の改善，表記揺れの解消，ほかの基本ツールの作成，IoT や調理ロボットなどへの応用のためのデータセットの作成などをあげられる．

謝　辞

本研究では，国立情報学研究所の IDR データセット提供サービスにより楽天株式会社から提供を受けた「楽天データセット」を利用した．また，本研究の一部は，2020 年度国立情報学研究所公募型共同研究（20FC01）の助成を受けたものである．ここに謝意を表す．

文　献

[1] 楽天株式会社. 楽天レシピデータ. 国立情報学研究所情報学研究データリポジトリ. (データセット), 2016. https://doi.org/10.32130/idr.2.4.

[2] 京都大学情報学研究科－日本電信電話株式会社コミュニケーション科学基礎研究所. Mecab: Yet another part-of-speech and morphological analyzer, 2020.5.15 閲覧. https://taku910.github.io/mecab/.

[3] 工藤拓, 松本裕治. チャンキングの段階適用による日本語係り受け解析. 情報処理学会論文誌, Vol. 43, No. 6, pp. 1834–1842, June 2002.

[4] megagonlabs. Ginza - japanese nlp library, 2020.5.15 閲覧. https://megagonlabs.github.io/ginza.

[5] 松田寛, 大村舞, 浅原正幸. 短単位品詞の用法曖昧性解決と依存関係ラベリングの同時学習. 言語処理学会第 25 回年次大会発表論文集, 2019.

[6] 笹田鉄郎, 森信介, 山肩洋子, 前田浩邦, 河原達也. レシピ用語の定義とその自動認識のためのタグ付与コーパスの構築. 自然言語処理, Vol. 22, No. 2, pp. 107–131, 2015.

[7] 森信介, 山肩洋子, 門脇拓也. レシピ言語処理マニュアル, 2020.5.15 閲覧. http://www.ar.media.kyoto-u.ac.jp/how-to/recipe-NLP/home.html.

[8] 森信介, 山肩洋子, 笹田鉄郎, 前田浩邦. レシピテキストのためのフローグラフの定義. 情報処理学会研究報告. 自然言語処理研究会報告, Vol. 2013, No. 13, pp. 1–7, November 2013.

[9] 前田浩邦, 山肩洋子, 森信介. 手順文書からの意味構造抽出. 人工知能学会論文誌, Vol. 32, No. 1, pp. E–G24 1–8, 2017.

[10] 苅米志帆乃, 藤井敦. 料理レシピテキストの構造解析とその応用. 言語処理学会 第 18 回年次大会 発表論文集, 2012.

[11] 浜田玲子, 井手一郎, 坂井修一, 田中英彦. 料理教材における手順の構造化. 第 60 回全国大会講演論文集, Vol. 2000, No. 1, pp. 9–10, March 2000.

[12] 杉山沙希, 妹尾堅一郎, 伊澤久美, 関本奈菜子. クックパッド株式会社の料理レシピサイト事業: プラットフォームビジネスの進展に関す一考察. 年次学術大会講演要旨集, Vol. 33, pp. 322–326, 2018.

[13] 伊尾木将之. Cookpad に投稿されたレシピを，機械可読なデータに変換する mrr 開発の舞台裏, 2020.5.15 閲覧. https://logmi.jp/tech/articles/321152.

[14] 滝聖子, 大崎紘一, 宗澤良臣, 梶原康博. 作業分析によるロボット料理レシピの作成手法. 日本経営工学会論文誌, Vol. 56, No. 4, pp. 302–311, 2005.

[15] Wikipedia. 形式言語, 2020.11.18 閲覧. https://ja.wikipedia.org/wiki/形式言語.

[16] Python Software Foundation. Python 3.8.3 documentation, 2020.5.16 閲覧. https://docs.python.org/3/reference/expressions.html.

[17] Wikipedia. Ebnf, 2020.5.16 閲覧. https://ja.wikipedia.org/wiki/EBNF.