卒 業 論 文

意味主導の日本語構文解析手法の確立を目指した基礎研究

東北大学工学部

B8TB2211 牧野雅紘

指導教員　大堀 淳 教授，菊池 健太郎 助教

提出年月: 令和 4 年 1 月 31 日

概要

本研究の目的は, 三上章により体系化された文法論を用いて意味主導で日本語構文を解析することである. 本論文ではまず, 三上章が提唱した文法論である三上文法と日本語の特徴について述べる. その後 SMLSharp を用いた解析手法を提案し, 具体的なシステム構築の戦略を述べる. 最後にシステムの評価と今後の課題について考察を行う.

目 次

第1章　序論

1.1　背景と目的

日本語には語順の曖昧さや, 主語が存在しない等の特徴がある. これらの特徴は日本語特有のものであり, 英語を中心とする西洋の言語にはない特徴である. そのため西洋言語の構文を高い精度で解析することに長けた文法主導解析では日本語の構文を十分に解析することは不可能であると考える.

そこで本研究では, 意味主導で日本語構文を解析することを目的とする. そのために, 日本語の言語学者である三上章が提唱した三上文法と関数型言語である SMLSharp を利用し, システム構築を行う.

1.2　関連研究

言語学者である三上章が体系化した文法規則を記した [1] と東北大学教授, 大堀淳の研究提案草稿 [2] を参考文献として使用している.

1.3　本論文の構成

本論文の構成は次のとおりである. 2 章では本研究で利用する三上文法について紹介する. 3 章では三上が紹介する日本語の特徴について述べる. 4 章では意味主導解析の方法について論じる. 5 章では具体的なシステム構築について述べる. 6 章ではシステムの評価を考察し 6 章で今後の課題について述べる.

第2章　三上文法について

三上章は日本語で最も重要な文法的手段を助詞の”は”であると述べている．”は”には 2 つの役割が存在する．1 つ目は主格ではなく提題, つまりこれから話す内容の題目を提示することであり 2 つ目は助詞である”がのにを”の役割を代行することである．この章では上記に述べた三上文法における助詞の”は”が果たす役割を紹介していく．

2.1　提題を表す助詞”は”

多くの日本語文法において, 助詞の”は”は助詞の”が”と並び主格であるとされている中, 三上章は, 主格の助詞を”が”のみであると述べている．”は”は提題の役割, つまりこれから話す内容の題目を提示しているのであって主格ではないと言う．例を用いて説明しよう．

- 象は鼻が長い
- わたしは富山の魚が好き

上の例では, ”は”により象という話題について話すことを明示している．その後, 鼻に”が”が付帯することで長い主体は鼻であることが説明されている．従来の日本語文法では”象は”を総主語, ”鼻が”を主語とするなどのように主格が 2 つあることで主語が 2 つ存在することを認めていた．三上章は, ”象は”が題目, ”鼻が”を動作の仕手とし, 主語の存在を否定している．下の例では, ”わたしは”が題目, ”富山の魚”を仕手としている．

2.2　他の助詞を代行する助詞”は”

助詞の”は”は, 他の助詞”がのにを”の役割を代行する. 以下に例を示す．

- 父はこの本を買ってくれました
- 父がこの本を買ってくれました

”は”が最も代行するのが”が”であり, これが”は”と”が”を同じ主格とする考えの根拠になっている．ただ”は”が文末まで係るにもかかわらず”が”は直後の語幹までしか係らない．また”は”は他の助詞を代行するが, ”が”は他の助詞を代行する役割はない．そもそも”は”が”が”以外の助詞を代行する場合は”は”を主格とは言えない．これらの違いは大きく, ”は”と”が”を同じ主格として扱ってはいけない根拠となっている．

- 去年は夏休みに沖縄に行った
- 去年の夏休みに沖縄に行った

”X は Y”において, Y が X の性質や消息を表している場合は”X の Y”と変換が可能である．

- 秋は色々な行事が続く
- 秋に色々な行事が続く

”X は”において X が時や人, 所の位置を表している場合は”X の”と変換が可能である.

- メバルは煮つけにする
- メバルを煮つけにする

”は”が”を”の役割を代行する際に注意しなければいけないのが, ”は”は決して動作の仕手を表してはいないということである. 例の仕手は料理を実際に行っている人であり, 決してメバルではない. この例からも”は”を必ずしも主格としてはならないという主張の妥当性が分かるだろう.

第3章　日本語の特徴

日本語には主語がない, 連用修飾語の語順は自由である等の特徴があると三上章は述べている. この章では以上で述べた日本語の特徴について紹介する.

3.1　主語のない日本語

西洋語から輸入された主語は 2 つの特徴を持つ.

- 述語と呼応する
- 述語に対する動作主となる

例を用いて説明しよう.

- She plays baseball.
- I play baseball.

以上のように主語と述語は呼応するため, She から I に主語が変われば述語の形も変化する. また play(plays) の動作主は I(She) であることも明らかである. 日本語における主語と呼ばれているものには, 以上のような特徴を持っていないことを例を使って説明する.

- 私は紹介する
- 私が紹介する
- 私に紹介する
- 私を紹介する

いわゆる主語が変わったとしても述語は変化しない. は格や, が格を違う格で置き換えたとしても述語の変化はない. つまりは格やが格が述語と呼応していないことが分かる.

象は鼻が長い

またこの文の述語である長いの動作主は鼻であり, 象ではない. つまりは格は動作主にならない場合がある. 以上のことからは格, が格が”述語と呼応する, 述語の動作主である”の 2 つの特徴を備えていないことが分かる. このことから日本語には主語がないと主張が可能である.

3.2　連用修飾語の語順は自由

日本語では連用修飾語の語順が自由である. 例をあげて説明しよう.

私はあなたに数学を教える

(私は) と (あなたに) と (数学を) の 3 つの連用修飾語は入れ替え可能である. 例えば私は数学をあなたに教えるなどでも日本語文法では可能なのである. これは語順に厳格な英語にはない特徴である.

第4章　意味主導解析の方法

そこで意味主導解析のシステム構築について大堀に提案されている戦略 [2] を用いる.

4.1　意味表現の定義

文の意味を表現するために, ラベル付きレコードで意味表現を定義する. このレコードは, 以下の 3 つの属性を保持している.

- 文の種類を表す”種類”
- 文の主題を表す”提題”
- 文の述語を表す”述語”

本研究では, 簡略化のために文の種類を動詞文, 形容詞文の 2 つに絞っている. 例として”象は鼻が長い”の意味表現を挙げる.

```
{
  種類 = 動詞文,
  提題 = {名詞 = “ 象 ”},
  述語 = {動詞 = “ 速く ”, 主格 = {名詞 = “ 象 ”}}
}
```

4.2　文節ごとの部分意味表現

次に文節ごとの意味表現, 部分意味表現を定義する. 考えられ得る属性は 3 つ以下である. 文とは文節の集合であり, 文の意味表現は文節の部分意味表現の集合である. つまり部分意味表現は不完全なものであり, 3 つの属性を必ず持つとは限らない. また文全体の情報を用いて文節の部分意味表現を生成するのではなく文節のみの情報から部分意味表現を生成するため, 部分意味表現は文全体の意味表現の構成要素の候補を複数保持するリスト型となる. 例として”象は”の部分意味表現を挙げる.

```
{
  種類 = "形容詞文",
  述語 = {主格 = {名詞 = #名詞 s}},
  提題 = {名詞 = #名詞 s}
},
{
```

```
      種類 = "形容詞文",
      述語 = {主格 = {の格 = #名詞 s}},
      提題 = {名詞 = #名詞 s}
    },
    {
      種類 = "形容詞文",
      述語 = {に格 = #名詞 s},
      提題 = {名詞 = #名詞 s}
    },
    {
      種類 = "動詞文",
      述語 = {主格 = {名詞 = #名詞 s}},
      提題 = {名詞 = #名詞 s}
    },
    {
      種類 = "動詞文",
      述語 = {主格 = {の格 = #名詞 s}},
      提題 = {名詞 = #名詞 s}
    },
    {
      種類 = "動詞文",
      述語 = {に格 = #名詞 s},
      提題 = {名詞 = #名詞 s}
    },
    {
      種類 = "動詞文",
      述語 = {を格 = #名詞 s},
      提題 = {名詞 = #名詞 s}
    }
]
```

4.3 自然結合演算

SMLsharp における自然結合演算 join は, ラベル付きレコードのリスト同士を演算対象とする. まず共通のラベルをもっていたら, 値が同一であるかをテストする. 同一のラベルの値がひとつでも存在したら, 2 つのラベル付きレコードの和を取る. 例として”象が”と”走る”の自然演算結合の概要を以下に示す.

まず”象が”の部分意味表現は以下のようになる.”

```
    {
      種類 = "形容詞文",
      述語 = {主格 = {の格 = #名詞 s}},
      提題 = {名詞 = #名詞 s}
```

```
    },
    {
      種類 = "動詞文",
      述語 = {主格 = {名詞 = #名詞 s}},
      提題 = {名詞 = #名詞 s}
    }
]
```

次に”走る”の部分意味表現は以下のようになる.

```
[
  {
    種類 = 動詞文,
    述語 = {動詞 = “ 走る ”}
  }
]
```

次に”象が”と”走る”の部分意味表現を自然演算結合した結果である”象が走る” の意味表現は以下のようになる.

```
[
  {
    種類 = 動詞文,
    提題 = {名詞 = “ 象 ”},
    述語 = {動詞 = “ 走る， ”主格 = {名詞 = “ 象 ”}}
  }
]
```

”象が”の 1 つ目の部分意味表現と”走る”の部分意味表現は共通の属性の値が存在しない. ”象が”の 2 つ目の部分意味表現と”走る”の部分意味表現は種類属性の値, 動詞文が共通しているため”象が”の 2 つ目の部分意味表現と”走る”の部分意味表現の和が自然結合演算の結果となる.

以上のことから文節ごとに部分意味表現を生成し, 後に join することで整合性の取れた意味表現のみを結果として得られる. 文節ごとに独立して部分意味表現を生成するため, 語順による構文解析が不必要でありかつ, は格の取り得る部分意味表現を生成することも容易である.

第5章　システム構築

システムの行う流れは, まず解析したい語において形態素解析を行う. 次に三上文法を使って文節ごとに可能な部分意味表現を作り出す. その後, join を用いて自然演算結合を行う. この章では上記の流れを実行するシステム構築について述べる.

5.1　juman++による形態素解析

5.1.1　juman++とは

京都大学院の黒橋研究室により開発がされた形態素解析システムである. 言語モデルとして Recurrent Neural Network Language Model を利用しているのが特徴的である.

5.1.2　形態素解析

形態素解析により, 単語間の区切りと品詞が分かる. 形態素ごとに以下のような情報を保持したラベル付きレコードを作り, そのレコードを要素としたリストを作成する.

```
type jumanOutputTy =
  {
    表層形 : string,
    読み : string,
    見出し語 : string,
    品詞大分類 : string,
    品詞大分類_ID : int option,
    品詞細分類 : string,
    品詞細分類_ID : int option,
    活用型 : string option,
    活用型_ID : int option,
    活用形 : string option,
    活用形_ID : int option,
    意味情報 : jumanOutputSemTy option
  }
```

”象は鼻が長い”の形態素解析の結果を以下に示す.

```
  [
      {
```

```
  1_表層形 = "象",
  2_読み = "ぞう",
  3_見出し語 = "象",
  4_品詞大分類 = "名詞",
  5_品詞大分類_ID = SOME 6,
  6_品詞細分類 = "普通名詞",
  7_品詞細分類_ID = SOME 1,
  8_活用型 = NONE,
  9_活用型_ID = NONE,
  10_活用形 = NONE,
  11_活用形_ID = NONE,
  12_意味情報 =
    SOME
      {
        カテゴリ = NONE,
        代表表記 = SOME "象/ぞうカテゴリ:動物漢字読み:音",
        反義 = NONE,
        漢字読み = NONE
      }
},
{
  1_表層形 = "は",
  2_読み = "は",
  3_見出し語 = "は",
  4_品詞大分類 = "助詞",
  5_品詞大分類_ID = SOME 9,
  6_品詞細分類 = "副助詞",
  7_品詞細分類_ID = SOME 2,
  8_活用型 = NONE,
  9_活用型_ID = NONE,
  10_活用形 = NONE,
  11_活用形_ID = NONE,
  12_意味情報 = NONE
},
{
  1_表層形 = "鼻",
  2_読み = "はな",
  3_見出し語 = "鼻",
  4_品詞大分類 = "名詞",
  5_品詞大分類_ID = SOME 6,
  6_品詞細分類 = "普通名詞",
  7_品詞細分類_ID = SOME 1,
```

```
  8_活用型 = NONE,
  9_活用型_ID = NONE,
  10_活用形 = NONE,
  11_活用形_ID = NONE,
  12_意味情報 =
    SOME
      {
        カテゴリ = NONE,
        代表表記 =
          SOME "鼻/はなカテゴリ:動物-部位漢字読み:訓",
        反義 = NONE,
        漢字読み = NONE
      }
},
{
  1_表層形 = "が",
  2_読み = "が",
  3_見出し語 = "が",
  4_品詞大分類 = "助詞",
  5_品詞大分類_ID = SOME 9,
  6_品詞細分類 = "格助詞",
  7_品詞細分類_ID = SOME 1,
  8_活用型 = NONE,
  9_活用型_ID = NONE,
  10_活用形 = NONE,
  11_活用形_ID = NONE,
  12_意味情報 = NONE
},
{
  1_表層形 = "長い",
  2_読み = "ながい",
  3_見出し語 = "長い",
  4_品詞大分類 = "形容詞",
  5_品詞大分類_ID = SOME 3,
  6_品詞細分類 = "*",
  7_品詞細分類_ID = NONE,
  8_活用型 = SOME "イ形容詞アウオ段",
  9_活用型_ID = SOME 18,
  10_活用形 = SOME "基本形",
  11_活用形_ID = SOME 2,
  12_意味情報 =
    SOME
```

```
        {
          カテゴリ = NONE,
          代表表記 =
            SOME "長い/ながい反義:形容詞:短い/みじかい",
          反義 = NONE,
          漢字読み = NONE
        }
    }
  ]
```

Juman++の形態素解析結果が未定義の場合は NONE としている.

5.2 文節 (名詞+助詞) に分ける

文節に分けることで, 名詞に対して助詞の果たす格の候補を特定できる. ”が”であればが格つまり主格の役割をはたしている. ”は”であれば”は”がのにを”を代行する事実から, が格, の格, に格またはを格が候補となる. このように考えられる格の種類によって表現されうる文の意味表現を列挙することが可能である.

形態素解析の結果, 各形態素の品詞は特定できているため, 形態素の集合を文節の集合に分けることは容易である. 以下のような関数を定義して文節分けを実現した.

```
fun separate record lis =
if lis = nil then
  case #4_品詞大分類 record of
    "形容詞" => [Type.形容詞 {形容詞 = #1_表層形 record, 形容詞情報 = record}]
  | "動詞" => [Type.動詞 {動詞 = #1_表層形 record, 動詞情報 = record}]
  | "名詞" => [Type.名詞 {名詞 = #1_表層形 record, 名詞情報 = record}]
  | _ => raise Fail "unknown hinshi"
else
  if #4_品詞大分類 (List.hd lis) = "助詞" then
    case #1_表層形 (List.hd lis) of
      "は" => List.@ ([Type.名詞は {名詞 = #1_表層形 record, 名詞情報 = record, 助
詞情報 = List.hd lis}], separate (List.hd (List.tl lis)) (List.tl (List.tl lis)))
    | "が" => List.@ ([Type.名詞が {名詞 = #1_表層形 record, 名詞情報 = record, 助
詞情報 = List.hd lis}], separate (List.hd (List.tl lis)) (List.tl (List.tl lis)))
    | "の" => List.@ ([Type.名詞の {名詞 = #1_表層形 record, 名詞情報 = record, 助
詞情報 = List.hd lis}], separate (List.hd (List.tl lis)) (List.tl (List.tl lis)))
    | "に" => List.@ ([Type.名詞に {名詞 = #1_表層形 record, 名詞情報 = record, 助
詞情報 = List.hd lis}], separate (List.hd (List.tl lis)) (List.tl (List.tl lis)))
    | "を" => List.@ ([Type.名詞を {名詞 = #1_表層形 record, 名詞情報 = record, 助
詞情報 = List.hd lis}], separate (List.hd (List.tl lis)) (List.tl (List.tl lis)))
    | "へ" => List.@ ([Type.名詞へ {名詞 = #1_表層形 record, 名詞情報 = record, 助
詞情報 = List.hd lis}], separate (List.hd (List.tl lis)) (List.tl (List.tl lis)))
```

```
    | _ => raise Fail "unknown hinshi"
  else
    case #4_品詞大分類 record of
      "形容詞" => List.@ ([Type.形容詞 {形容詞 = #1_表層形 record, 形容詞情報 = record}], separate (
    | "動詞" => List.@ ([Type.動詞 {動詞 = #1_表層形 record, 動詞情報 = record}], separate (List.hd
    | "名詞" => List.@ ([Type.名詞 {名詞 = #1_表層形 record, 名詞情報 = record}], separate (List.hd
    | _ => raise Fail "unknown hinshi"
```

形態素の品詞をレコードから取り出し, 助詞であれば前に存在する名詞と助詞の情報を統合し, 文節に分けることを実現している.

以下に"象は鼻が長い"の文節分け結果を示す.

```
[
    名詞は
    {
      助詞情報 =
        {
          1_表層形 = "は",
          2_読み = "は",
          3_見出し語 = "は",
          4_品詞大分類 = "助詞",
          5_品詞大分類_ID = SOME 9,
          6_品詞細分類 = "副助詞",
          7_品詞細分類_ID = SOME 2,
          8_活用型 = NONE,
          9_活用型_ID = NONE,
          10_活用形 = NONE,
          11_活用形_ID = NONE,
          12_意味情報 = NONE
        },
      名詞 = "象",
      名詞情報 =
        {
          1_表層形 = "象",
          2_読み = "ぞう",
          3_見出し語 = "象",
          4_品詞大分類 = "名詞",
          5_品詞大分類_ID = SOME 6,
          6_品詞細分類 = "普通名詞",
          7_品詞細分類_ID = SOME 1,
          8_活用型 = NONE,
          9_活用型_ID = NONE,
          10_活用形 = NONE,
```

```
      11_活用形_ID = NONE,
      12_意味情報 =
        SOME
          {
            カテゴリ = NONE,
            代表表記 =
              SOME "象/ぞうカテゴリ:動物漢字読み:音",
            反義 = NONE,
            漢字読み = NONE
          }
    }
},
名詞が
{
  助詞情報 =
    {
      1_表層形 = "が",
      2_読み = "が",
      3_見出し語 = "が",
      4_品詞大分類 = "助詞",
      5_品詞大分類_ID = SOME 9,
      6_品詞細分類 = "格助詞",
      7_品詞細分類_ID = SOME 1,
      8_活用型 = NONE,
      9_活用型_ID = NONE,
      10_活用形 = NONE,
      11_活用形_ID = NONE,
      12_意味情報 = NONE
    },
  名詞 = "鼻",
  名詞情報 =
    {
      1_表層形 = "鼻",
      2_読み = "はな",
      3_見出し語 = "鼻",
      4_品詞大分類 = "名詞",
      5_品詞大分類_ID = SOME 6,
      6_品詞細分類 = "普通名詞",
      7_品詞細分類_ID = SOME 1,
      8_活用型 = NONE,
      9_活用型_ID = NONE,
      10_活用形 = NONE,
```

```
        11_活用形_ID = NONE,
        12_意味情報 =
          SOME
            {
              カテゴリ = NONE,
              代表表記 =
                SOME "鼻/はなカテゴリ:動物-部位漢字読み:訓",
              反義 = NONE,
              漢字読み = NONE
            }
      }
  },
  形容詞
  {
    形容詞 = "長い",
    形容詞情報 =
      {
        1_表層形 = "長い",
        2_読み = "ながい",
        3_見出し語 = "長い",
        4_品詞大分類 = "形容詞",
        5_品詞大分類_ID = SOME 3,
        6_品詞細分類 = "*",
        7_品詞細分類_ID = NONE,
        8_活用型 = SOME "イ形容詞アウオ段",
        9_活用型_ID = SOME 18,
        10_活用形 = SOME "基本形",
        11_活用形_ID = SOME 2,
        12_意味情報 =
          SOME
            {
              カテゴリ = NONE,
              代表表記 =
                SOME "長い/ながい反義:形容詞:短い/みじかい",
              反義 = NONE,
              漢字読み = NONE
            }
      }
  }
]
```

5.3 文節ごとに部分意味表現を作成する

文節ごとの意味を保持した表現 (部分意味表現と呼ぶ) を作る. 部分意味表現を作成するにあたって, ”は”が”がのにを”を代行すること, ”は”は文の提題を行うことを利用する. ”名詞+は”の文節を検出した際にはレコード属性の”提題”に”は”の前に存在する名詞の情報を格納する. また”は”が代行する助詞の候補”がのにを”を上げ, それぞれの助詞に対応する部分意味表現をレコードとして作成する. 文節を受け取り, その文節の部分意味表現を生成する関数を以下に示す.

```
fun undone t =
  case t of
    Type.名詞は s =>
      [
        Dynamic.dynamic
          {
            種類 = "形容詞文",
            述語 = {主格 = {名詞 = #名詞 s}},
            提題 = {名詞 = #名詞 s}
          },
        Dynamic.dynamic
          {
            種類 = "形容詞文",
            述語 = {主格 = {の格 = #名詞 s}},
            提題 = {名詞 = #名詞 s}
          },
        Dynamic.dynamic
          {
            種類 = "形容詞文",
            述語 = {に格 = #名詞 s},
            提題 = {名詞 = #名詞 s}
          },
        Dynamic.dynamic
          {
            種類 = "動詞文",
            述語 = {主格 = {名詞 = #名詞 s}},
            提題 = {名詞 = #名詞 s}
          },
        Dynamic.dynamic
          {
            種類 = "動詞文",
            述語 = {主格 = {の格 = #名詞 s}},
            提題 = {名詞 = #名詞 s}
          },
        Dynamic.dynamic
```

```
        {
          種類 = "動詞文",
          述語 = {に格 = #名詞 s},
          提題 = {名詞 = #名詞 s}
        },
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {を格 = #名詞 s},
          提題 = {名詞 = #名詞 s}
        }
    ]
| Type.名詞が s =>
    [
      Dynamic.dynamic
        {
          種類 = "形容詞文",
          述語 = {主格 = {名詞 = #名詞 s}}
        },
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {主格 = {名詞 = #名詞 s}}
        }
    ]
| Type.名詞の s =>
    [
      Dynamic.dynamic
        {
          種類 = "形容詞文",
          述語 = {主格 = {の格 = #名詞 s}}
        },
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {主格 = {の格 = #名詞 s}}
        }
    ]
| Type.名詞に s =>
    [
      Dynamic.dynamic
        {
```

```
          種類 = "形容詞文",
          述語 = {に格 = #名詞 s}
        },
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {に格 = #名詞 s}
        }
    ]
| Type.名詞を s =>
    [
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {を格 = #名詞 s}
        }
    ]
| Type.名詞へ s =>
    [
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {主格 = {へ格 = #名詞 s}}
        }
    ]
| Type.形容詞 s =>
    [
      Dynamic.dynamic
        {
          種類 = "形容詞文",
          述語 = {形容詞 = #形容詞 s}
        },
      Dynamic.dynamic
        {
          種類 = "動詞文",
          述語 = {主格 = {連用修飾 = #形容詞 s}}
        }
    ]
| Type.動詞 s =>
    [
      Dynamic.dynamic
        {
```

```
            種類 = "動詞文",
            述語 = {動詞 = #動詞 s}
          }
      ]
  | Type.名詞 s =>
      [
        Dynamic.dynamic
          {
            種類 = "準詞文",
            述語 = {名詞 = #名詞 s}
          }
      ]
```

以下に”象は”の文節の部分意味表現を例として挙げる.

```
名詞は
[
    {
      種類 = "形容詞文",
      述語 = {主格 = {名詞 = "象"}},
      提題 = {名詞 = "象"}
    },
  Dynamic.dynamic
    {
      種類 = "形容詞文",
      述語 = {主格 = {の格 = "象"}},
      提題 = {名詞 = "象"}
    },
  Dynamic.dynamic
    {
      種類 = "形容詞文",
      述語 = {に格 = "象"},
      提題 = {名詞 = "象"}
    },
  Dynamic.dynamic
    {
      種類 = "動詞文",
      述語 = {主格 = {名詞 = "象"}},
      提題 = {名詞 = "象"}
    },
  Dynamic.dynamic
    {
      種類 = "動詞文",
```

```
      述語 = {主格 = {の格 = "象"}},
      提題 = {名詞 = "象"}
    },
  Dynamic.dynamic
    {
      種類 = "動詞文",
      述語 = {に格 = "象"},
      提題 = {名詞 = "象"}
    },
  Dynamic.dynamic
    {
      種類 = "動詞文",
      述語 = {を格 = "象"},
      提題 = {名詞 = "象"}
    }
]
```

5.4 部分意味表現を自然結合演算

文節ごとの部分意味表現を自然結合演算する. そうすることで文全体の意味表現の候補を出力結果として得られる. 以下に”象は鼻が長い”の意味表現の候補を挙げる.

```
    {
      "提題" : {"名詞" : "象"},
      "種類" : "形容詞文",
      "述語" :
        {"主格" : {"の格" : "象", "名詞" : "鼻"}, "形容詞" : "長い"}
    }
    {
      "提題" : {"名詞" : "象"},
      "種類" : "形容詞文",
      "述語" :
        {"に格" : "象", "主格" : {"名詞" : "鼻"}, "形容詞" : "長い"}
    }
    {
      "提題" : {"名詞" : "象"},
      "種類" : "動詞文",
      "述語" :
        {"主格" : {"の格" : "象", "名詞" : "鼻"}, "連用修飾" : "長い"}
    }
    {
      "提題" : {"名詞" : "象"},
```

```
  "種類" : "動詞文",
  "述語" :
    {"に格" : "象", "主格" : {"名詞" : "鼻"}, "連用修飾" : "長い"}
}
{
  "提題" : {"名詞" : "象"},
  "種類" : "動詞文",
  "述語" :
    {"を格" : "象", "主格" : {"名詞" : "鼻"}, "連用修飾" : "長い"}
}
```

第6章　評価

評価を行うために複数の文の解析結果を以下に示す.

- 象は鼻が長い

```
{
  "提題" : {"名詞" : "象"},
  "種類" : "形容詞文",
  "述語" :
    {"主格" : {"の格" : "象", "名詞" : "鼻"}, "形容詞" : "長い"}
}
{
  "提題" : {"名詞" : "象"},
  "種類" : "形容詞文",
  "述語" :
    {"に格" : "象", "主格" : {"名詞" : "鼻"}, "形容詞" : "長い"}
}
{
  "提題" : {"名詞" : "象"},
  "種類" : "動詞文",
  "述語" :
    {"主格" : {"の格" : "象", "名詞" : "鼻"}, "連用修飾" : "長い"}
}
{
  "提題" : {"名詞" : "象"},
  "種類" : "動詞文",
  "述語" :
    {"に格" : "象", "主格" : {"名詞" : "鼻"}, "連用修飾" : "長い"}
}
{
  "提題" : {"名詞" : "象"},
  "種類" : "動詞文",
  "述語" :
    {"を格" : "象", "主格" : {"名詞" : "鼻"}, "連用修飾" : "長い"}
}
```

1つ目の意味表現は象の鼻が長いという解析結果が得られており, 正しい. 2つ目の意

味表現は象に鼻が長いという解析結果が得られている. 象は鼻が長いの文の意味としては不適切ではあるが構文的には問題がない. 富士山は私が登るなどの文章などの例からもわかる. 3 つ目 5 つ目の意味表現は述語動詞が存在していないにもかかわらず, 動詞文として扱われており, 不適切である.

- 私は彼に仕事を勧める


```
{
  "提題" : {"名詞" : "私"},
  "種類" : "動詞文",
  "述語" :
    {
      "に格" : "彼",
      "を格" : "仕事",
      "主格" : {"名詞" : "私"},
      "動詞" : "勧める"
    }
}
{
  "提題" : {"名詞" : "私"},
  "種類" : "動詞文",
  "述語" :
    {
      "に格" : "彼",
      "を格" : "仕事",
      "主格" : {"の格" : "私"},
      "動詞" : "勧める"
    }
}
```


1 つ目の意味表現では私が彼に仕事を勧めるという解析結果が得られており, 正しい. 2 つ目の意味表現は, の格のに対する名詞が存在しておらず正しくない. ただ, に格が”私の彼”と解析ができれば, 私の彼に仕事を勧めるという文として解析したことになり画期的である.

- メバルは煮る


```
{
  "提題" : {"名詞" : "メバル"},
  "種類" : "動詞文",
  "述語" : {"主格" : {"名詞" : "メバル"}, "動詞" : "煮る"}
}
{
  "提題" : {"名詞" : "メバル"},
  "種類" : "動詞文",
```

```
  "述語" : {"主格" : {"の格" : "メバル"}, "動詞" : "煮る"}
}
{
  "提題" : {"名詞" : "メバル"},
  "種類" : "動詞文",
  "述語" : {"に格" : "メバル", "動詞" : "煮る"}
}
{
  "提題" : {"名詞" : "メバル"},
  "種類" : "動詞文",
  "述語" : {"を格" : "メバル", "動詞" : "煮る"}
}
```


4 つ目のメバルを煮るが正しい解析結果である. 2 つ目の解析結果は, の格に対する名詞が存在しておらず正しくない.

第7章　今後の課題

今後の課題として以下をあげる.

- 述語としての形容詞が存在していない形容詞文の意味表現を排除
- 述語としての動詞が存在していない動詞文の意味表現を排除
- の格の対象となる名詞が存在していない文の意味表現を排除

以上のような課題を改善し改良を重ねていけば, より美しい日本語に訂正するシステムを開発し実用化も見えてくるだろう.

謝辞

本論文の作成にあたり, 指導してくださった大堀淳教授, 菊池健太郎助教に感謝申し上げます. 並びに, 研究生活を支えてくださった大堀研究室の皆様に感謝いたします.

参考文献

[1] 三上章. 象は鼻が長い. くろしお出版, 2021

[2] 大堀淳. 意味論主導の日本語構文解析について. unpublished manuscript, 2022.