

2 自然言語処理

2.1 自然言語処理

言語は、大きく分けて、人工言語と自然言語に分類することができます。人工言語とは、人工的に作られた言語です。例えば、コンピュータに関連する人工言語としてはプログラミング言語があります。プログラミング言語には、目的などの違いにより、Fortran, Basic, C, Javaといった様々な言語があります。Fortranは、数値計算を行うための言語として開発されました。Basicは、初心者向けのコンピュータ言語として開発されました。Cは、UNIXと呼ばれるOS（オペレーティングシステム）で標準的に利用される言語です。Javaは、OSに依存しないプログラミングの実現を目指して開発された言語です。このような人工言語は、文法が明確であり、コンピュータにとって扱いやすい言語と言えます。一方、自然言語は、日本語や英語、フランス語、ドイツ語、中国語など、我々人間がコミュニケーションをするのに使う、日常生活で用いる言語のことです。自然言語は、文法が必ずしも厳密でない部分や例外等も多く、このため、コンピュータによる処理は人工言語に比べて難しくなります。

このような自然言語の処理を行うのが自然言語処理です。自然言語処理では、一般に、1) 形態素解析 2) 構文解析 3) 意味解析 4) 文脈解析といった処理が行われます。

2.2 形態素解析

形態素解析の形態素とは、言葉として意味を持つ最小の単位です。1つの形態素、あるいは複数の形態素が組み合わさることで1つの単語を形づくります。例えば、3角形を表す英語のtriangleという単語では、triおよびangleが形態素となります。Triは3、angleは角という意味を表します。しかし、コンピュータで処理する場合、形態素解析は、しばしば、単語レベルの処理といった意味合いで使われます。このような単語レベルの処理は、英語や日本語といった言語の違いによって、必要な処理が異なってきます。

英語と日本語の処理を行ううえで、最も大きな違いの1つに、それぞれの単語が繋がっているか否かの違いがあります。例えば、「This is a pen.」のような英文を見ると、Thisやisといった単語が空白（スペース）によって区切られています。この場合、文字列の中からスペースを表す文字コードを探し、それらの間に挟まれる文字列を選び出すことで、1つ1つの単語を切り出すことができます。

一方、日本語の「これは、ペンです。」といった文を考えると、1つ1つの単語が空白で分けられていないため、このままでは、コンピュータは、どこからどこまでが1つの単語であるか認識できません。このため、日本語を処理する場合、文を1つ1つの単語に分割することが、形態素解析の大きな目的の1つとなります。

形態素解析では、形態素を同定するために、辞書が用いられます。この辞書には、形態素の読み、品詞、活用型などが記述されています。また、専門用語に対応するため、医療などの特定の専門分野に特化した辞書が用意される場合もあります。このような辞書から、入力された文と一致するように、単語を順次探し出していくことで、もとの文を構成する単語の候補を求めています。しかし、単語に分割する際に、複数の候補が考えられるなど、曖昧性が生じる場合も多くあります。このような単語分割の曖昧性を解消する方法として、こうすればうまくいく場合が多いという経験則が用いられます。このような経験則としては、最長一致法や分割数最小法などがあります。

最長一致法は、頭からできるだけ長く辞書中の単語と一致させる方法です。例えば、「中国語学校」を単語分割する場合を考えてみましょう。辞書に、「中、中国、中国語、学、学校」といった単語が登録されていることを仮定します。最長一致法の場合、まず、「中国語学校」の頭から、一致する単語を探していきます。まず、「中」は可能な候補となります。また、「中国」も可能な候補となります。また、「中国語」も可能な候補となります。この場合、最も長く一致する「中国語」を1単語とします。次に、残った部分である「学校」の頭から、一致する単語を探します。まず、「学」は可能な候補となります。また、「学校」も可能な候補となります。この場合、最も長く一致する「学校」までを1単語とします。このように、最長一致法で分割した場合、「中国語学校」は、「中国語」と「学校」の2つの単語に分割されます。

一方、分割数最小法は、分割数が最も少なくなるように単語の分割を行う方法です。例えば、「中国語学校」は、「中」と「国語」と「学校」のように3分割することができます。また、「中国語」と「学校」のように2分割することも可能です。このように複数の可能性がある場合、より分割数の少ない分割を正しい答えであると考えるのが分割数最小法です。

単語に分割する際の経験則として、品詞の接続関係を利用することもできます。これは、2つの品詞が接続するか否かを設定し、文法的に不自然な単語分割を制限するものです。例えば、「形容詞の後に名詞は続く。しかし、助詞は接続しない」といった制限を設定します。また、例えば、「文頭に名詞が来ることはできる。しかし、助詞は文頭になれない」といった制限を設定します。このような品詞の接続関係による制限は、前述の最長一致法や分割数最小法と併用して用いられる場合もあります。

コストを設定し、そのコストを最小化することで、最適な単語の分割を求める方法もあります。例えば、つながりやすい品詞には低いコストを設定し、つながりにくい品詞には高いコストを設定します。これにより、つながりやすい品詞が優先されて単語分割が行われます。また単語の出現しやすさによってコストを設定することができます。出現しやすい単語のコストは低くし、出現しにくい単語のコストは高く設定します。これにより、出現しやすい単語を優先して、単語分割が行われます。それぞれのコストの設定については、人手で行う方法もありますが、多量の文書データを用いて統計的にコストを求めることもできます。

実際に、「太郎と林檎を食べた。」という文を形態素解析してみましょう。図2-1に示したのは、ChaSenという解析ソフトを用いて形態素解析を行った結果です。この例では、1列目は分割された単語を表しています。2列目は分割された単語の終止形を表しています。分割された「食べ」という単語の終止形が「食べる」であることが分かります。3列目は読み、4列目は発音を表しています。「太郎」の読みは、「たろう」ですが、発音は、「たろー」のように伸ばして発音することが分かります。5列目はそれぞれの単語の品詞の詳細になっています。EOSは、文の区切りを表します。この結果から、「太郎と林檎を食べた。」という文は、「太郎」という固有名詞と、「と」という助詞、「林檎」という名詞、「を」という助詞、「食ベル」という動詞、「た」という助動詞から成ることが分かります。

太郎	太郎	タロウ	タロー	名詞-固有名詞-人名-名
と	と	ト	ト	助詞-並立助詞
林檎	林檎	リンゴ	リンゴ	名詞-一般
を	を	ヲ	ヲ	助詞-格助詞-一般
食べ	食べる	タベ	タベ	動詞-自立 一段 連用形
た	た	タ	タ	助動詞 特殊・タ 基本形
。	。	。	。	記号-句点
EOS				

図2-1 形態素解析の例

コンピュータ上で形態素解析を行うには、ChaSen以外にもいくつかのフリーソフトを利用することができます。JUMANは、京都大学で開発されたもので、最も古い形態素解析ソフトの1つです。MeCabは、ChaSenなどに比べて処理が高速で、辞書に載っていない単語の処理を定義することができます。KAKASIは、日本語の全文検索システムであるNamazuの分かち書きなどに利用されています。

2.3 構文解析

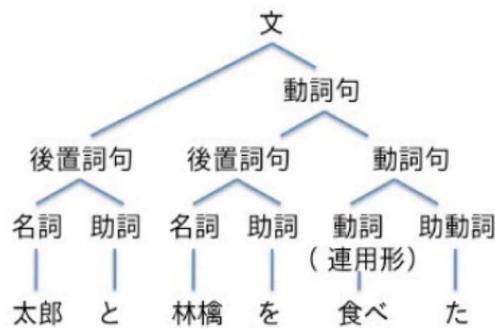
単語からどのように句や節が構成され、また句や節からどのように文が構成されているかといった文の構造を解析するのが構文解析になります。構文解析の方法としては、文法規則を形態素解析結果に適用していくことで構文解析を行う方法があります。英語の場合は、「in Japan」のように、名詞の前に前置詞がついて、前置詞句を形成します。日本語の場合は、「日本に」のように名詞の後に助詞がつきますが、これを後置詞句と呼びます。構文解析では、図2-2に示すような、「後置詞句は名詞と助詞で構成される」、「文は、後置詞句と動詞句から構成される」といった文法規則が利用されます。また、この際、「林檎」は名詞で、「と」は助詞であるというような、それぞれの単語が名詞なのか、助詞なのかといった情報も必要になります。

文	→	後置詞句	動詞句	名詞	→	太郎
後置詞句	→	名詞	助詞	名詞	→	林檎
後置詞句	→	名詞句	助詞	助詞	→	と
動詞句	→	後置詞句	動詞句	助詞	→	を
動詞句	→	動詞	助動詞	動詞	→	食べる
名詞句	→	名詞	助詞	名詞	助動詞	→ た

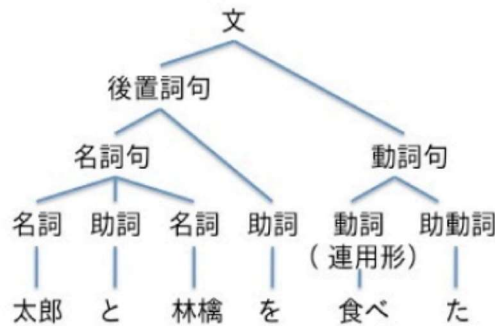
図2-2 文法規則の例

このような文法規則を用いた構文解析の方法としては、トップダウン法やボトムアップ法があります。トップダウン法では、文から文法規則を繰り返し適用することで、構文構造の同定が行われます。すなわちトップが文で、ボトムがそれぞれの単語に当たります。そして、文という大きな単位から処理を行い、単語の並びが実際の文の構造と合うものを探していきます。ボトムアップ法では、逆に、単語列から文へと、文法規則を繰り返し適用することで、構文構造を同定していきます。

構文解析を行うことで、例えば、図2-3のような構文解析結果を得ることができます。この図に示すように、構文構造は、上から下に枝分かれしていく木のような構造で表現することができます。このような木構造は、構文木と呼ばれます。構文解析の結果得られる構文木は、必ずしも1つになるとは限りません。図2-3(a)に示したものは意味的に正しい構文木です。一方、図2-3(b)のような構文木も可能です。この場合、「太郎」と「林檎」が並列の関係となり、「太郎」と「林檎」の両方を食べてしまうこととなります。これは、文法的には正しい構文木ですが、意味的には誤った解釈になります。このような誤りを避けるためには、文法規則だけではなく、「食べる」という動詞の対象が食べ物であるといった意味的な知識なども必要になります。



(a) 解釈 1



(b) 解釈 2

図2-3 構文解析結果

一方、日本語は、語順の自由度が高く、助詞によって意味が決まる場合も多くあります。このため、日本語の構文解析では、文法的な構造よりも、それぞれの文節の係り受け関係を明らかにすることを目的として構文解析が行われる場合もあります。

日本語の構文解析において、係り受け関係を求める単位となる文節としては、例えば、「太郎が」のような「名詞＋ガ」，「本を」のような「名詞＋ヲ」，「読み」のような「動詞連用形」，「読む」のような「動詞連体形」などが考えられます。この時，「名詞＋ガ」は，「動詞連用形」や「動詞連体形」には係るが，「名詞＋ガ」は，「名詞＋ヲ」には係らないといった制約を設定することができます。また，日本語の場合，例えば，以下に示すような制約を利用できます。

- 1) 係り受け関係は前の文節から後ろの文節に係る。
- 2) 係り要素は，通常，より近い受け要素に係る。
- 3) 係り受け関係は交差しない。

しかしながら，こういった制約には，例外があります。例えば，係り受けが交差している例としては，「うどんを香川に食べに行く」といった文があります。「うどんを」は「食べに」に係っています。また，「香川に」は「いく」に係っています。この場合，係り受け関係は交差してしまいます。しかし，このように係り受けが交差する場合は，通常は，稀なケースになります。

また，単語が同時に出現しやすいかどうかは，係り受け関係を決める場合の1つの手がかりとなります。例えば，「電話で話している太郎を見た。」と，「双眼鏡で話している太郎を見た。」という2つの文を考えてみましょう。この2つの文で違っているのは，「電話」と「双眼鏡」の部分だけです。この場合，「電話」と「双眼鏡」が，「話して」に係るのか「見た」に係るのかは文法だけでは決定できません。しかし，「電話」と「話す」は同時に出現しやすい単語です。また，「双眼鏡」と「見る」は同時に出現しやすい単語です。このような，単語が同時に出現しやすいかどうかという知識があれば，「電話」が「話して」に係り，「双眼鏡」が「見た」に係ると正しく係り受け関係を決定することができます。

一方，このような係り受け関係を決めるための制約や優先度は，すべてを人手で設定することは必ずしも容易ではありません。これを解決する方法の1つとして，多量の例文を収集し，そこから制約や優先度を抽出する方法が考えられます。このような例文の集まりはコーパスと呼ばれます。コーパスには，単純に例文を集めたものと，人手により，係り受け関係などの情報を別途付与したものとがあります。このようなコーパスから，統計的な性質を利用したり，類似する用例を探したりすることで，構文の曖昧性を解消することができます。

構文解析でも，いくつかのフリーソフトを利用することができます。KNPは，JUMANの形態素解析結果から構文解析を行います。また，奈良先端科学技術大学院大学で開発されたCaboChaなども利用することができます。

例えば，KNPを用いて「太郎と林檎を食べた。」という文を構文解析すると，図2-4のような結果が得られます。この結果を見ると，「太郎」と「林檎」が文節に分けられています。そして，それらの関係が並列の関係であると解釈しています。これは，「太郎」と「林檎」の両方を「食べた」という解釈を行っているということで，正しく構文解析を行えていないことが分かります。一般には，形態素解析よりも構文解析の方が，処理は難しくなります。

太郎と<P>┐
 林檎を<P>─PARA─┐
 食べた。
 EOS

図2-4 構文解析結果

2.4 意味解析と文脈解析

意味解析では、「格」と呼ばれる概念が用いられます。これは、文中でのそれぞれの単語の持つ役割を表すものです。代表的なものとしては、動作主格、対象格、時間格、場所格といったものがあります。動作主格は、動作を引き起こす主体となります。対象格は、動作が作用する対象となります。場所格、時間格は、動作が起こる場所や時間を表します。それぞれの単語が、どういう役割を果たしているか、つまり「文の格構造」を明らかにすることは、意味解析の大きな目的の1つとなります。

単語の格は、大きく「深層格」と「表層格」という2つに分けることができます。深層格とは、実際の意味的な役割を表す格で、さきほど挙げた、動作主格、対象格、時間格、場所格などがこの深層格に当たります。表層格とは、表層的な手がかりから決まる格です。日本語の場合、助詞によって表層格が決まります。例えば、助詞の「が」を伴う「ガ格」や、「を」を伴う「ヲ格」等が表層格に当たります。

文の格構造を決める場合、深層格は、表層格からある程度は推定できます。例えば、「ガ格」は、「太郎が食べる」のように動作主格になることが多くなります。しかし、「音楽が好き」のように対象格にもなり得ます。このため、深層格を決めるためには、例えば、「食べる」という動詞は、「(人/動物)ガ(食べ物)ヲ食べる」といった使われ方がされ、(人/動物)の部分が動作主格、(食べ物)の部分が対象格になるといった知識が必要になります。

格構造、すなわち、動詞と他の単語の持つ役割との関係には、いくつかのパターンを考えることができます。この格構造のパターンは、格フレームと呼ばれます。格フレームには、動詞について、それがどのような格を持つかや、その動詞に係る表層格と深層格の対応のしかたなどが記述されています。このような格フレームを、数多くの動詞について集めたものが格フレーム辞書になります。格フレーム辞書の例としては、例えば、IPALの動詞辞書や京都大学格フレーム等があります。

このような意味解析により、「太郎と林檎を食べた。」という文は、例えば、図2-5(a)のような格構造を持つと判断されます。この文の述語は「食べる」であり、時制は過去になります。また、食べる対象は林檎になります。また、その行動に太郎が随伴すると判断されます。これにより、「太郎と林檎を食べた。」という文の意味がかなり明確になってきました。しかし、この文だけ見ていたのでは、動作の主体となる動作主格は、明らかになりません。

一方、文脈解析は、前後の文脈を用いることで行われる処理です。文脈を用いた処理としては、いくつか考えられます。例えば、文と文との関係を明らかにすることが文脈解析の1つの目的になります。ある文とある文との関係が理由を述べたものなのか、例示しているのか、詳細化しているのか、といったことを明らかにします。この場合、「そして」や「しかし」といった接続詞などの情報を利用できます。また、「それはおいしい。」といった文で、「それ」が何かを推定するのも文脈解析の目的です。通常は、直前の文などから「それ」に対応するものを探します。また、省略された主語などを推定するといった処理も文脈解析の目的となります。この場合、前後の文から状況を把握することが必要になります。

例えば、「2人で食事をした。花子は林檎をむいた。太郎と林檎を食べた。」という文章があった場合、前後の文脈で、登場人物が2人で、太郎と花子が食事をしている場面であることが分かります。太郎と林檎を食べるわけですから、この場合、図2-5(b)のように、林檎を食べた動作を行ったのが花子であることが分かります。これにより、「太郎と林檎を食べた。」という文の意味を理解できたわけです。しかしコンピュータによる意味解析や文脈解析では、こういった「推定する」という処理を高い精度で実現することは必ずしも容易ではないのが現状です。

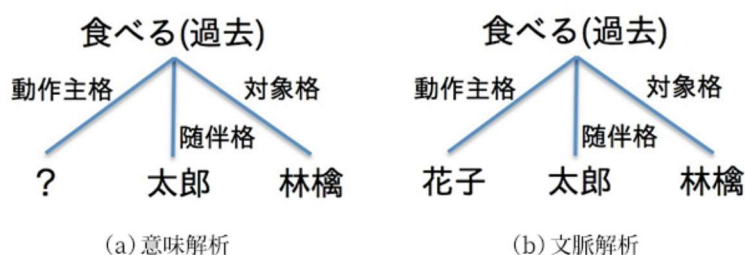


図2-5 意味解析と文脈解析

