自然言語処理の哲学

田中省作

Ⅰ　はじめに

自然言語処理の「自然言語」は，私たちのまわりに存在する日本語や英語などの自然発生した言語を指す。この自然言語を，計算機でさまざまに処理するのが自然言語処理である。日本国内の大学では，工学や情報学といった自然科学系の部局で研究や教育がされていることが多い。その自然言語処理の基盤には，広義の言語学の成果が随所にみられ，また自然言語処理の成果が人文・社会科学で，昨今のデータ主導の研究法と併せて活用されつつもある。

自然言語処理は，たしかに言語を扱う分野で，ときに計算言語学と標榜されることもある。そんな自然言語処理にかかわる研究者はもれなく，言語学に関心をもっていようし，言語学からの知見を享受している。そんな自然言語処理研究者らが，言語学研究者からとかくよく投げかけられる問いのひとつが「それは言語を研究していることになるのか」というものである。この問いは両者の言語観の差異を如実に表すものである。

自然言語処理の主辞は「処理」で，言語に向き合う主体は，人ではなく「計算機」である。自然言語処理研究者は，言語の観察者であると同時に，計算機に言語を向き合わせるための，人形使いならぬ計算機使いでもある。どのように計算機に言語と向き合わせるのか，本稿ではいくつかの断片的な研究設計を示しつつ，前段の質問に一部答える。

II では言語に向き合う計算機をまず確認する。そののち III で，言語をどのようにとらえていくことになるのか，言語の典型的で単純なモデル化の過程を示す。IV では，語の意味を数値列でとらえるような，一見すると人が解釈するにはほど遠い，言語の取り扱いをみる。

Ⅱ　言語に向き合う主体としての計算機

1　自然言語処理

自然言語処理とは，計算機で自然言語を処理する技術や研究分野である。自然言語処理の身近な成果には，私たちが計算機やスマートフォンなどで漢字を入力する際に，ひらがな列を入力し，漢字仮名混じり列を表出する仮名漢字変換，最近は相当に精度が高くなった英語を日本語に自動翻訳するような機械翻訳，電子メールの SPAM 判定などがあり，これらを日常的に使っている方も少なくないだろう。ほかにも，2022 年大きな話題となった ChatGPT も自然言語処理のわかりやすい成果である。この自然言語処理を，「方向」と「技術志向」という 2 つの軸で考えてみる[1)]。

まず方向には言語生成と言語解析がある。以降，それぞれを「生成」「解析」と記す。生成は

表現を作る，解析はその逆で，表現から語や構文，意味といった言語情報を得る。ChatGPT は人の指示をトリガに表現を自動的に綴っていくので，生成の自然言語処理である。コーパス研究やテキストマイニングなどでよく使われる，文中の語に品詞を自動的に振っていく形態素解析は，解析の自然言語処理である。解析と生成を行き来する自然言語処理もある。たとえば日本語から英語への機械翻訳は，入力された日本語表現を解析し，訳文生成のための処理を経たのち，英語表現を生成し，出力する。

技術志向には「基礎」と「応用」がある。基礎は，自然言語処理や他の情報処理技術の要素技術になるようなものである。形態素解析，構文構造を得る構文解析，意味構造を得る意味解析などが代表的なものである。それらは基礎解析とよばれる。応用は，具体的で特定の目的に構成される自然言語処理である。仮名漢字変換，機械翻訳，自動要約，対話・質問応答，誤り訂正など，枚挙に暇がない。

自然言語処理の研究姿勢にも触れておきたい。自然言語処理は文字通り，言語の「処理」が目的なので，議論の余地があることを承知の上で単純化すれば，言語そのものの探究には常に計算機での処理が念頭にある。自然言語処理では，処理の入出力に対する妥当性や精度が，重要な指標のひとつとなる。精度良く妥当な結果が得られるのであれば，処理の過程は人が言語を処理するような認知的手続きに沿わなくても良い，という考えは許容され得る。一方で，自然言語処理の具体的な課題を解いていく過程で，私たちの言語に対する直感や知見が，とりわけ研究初期においては重要な手がかりになっていることも多い。

2　計算機が言語に向き合うための要件

自然言語処理は，利用者が処理の途中，必然的に割り込むような設計となっていなければ，原則，計算機単独で言語を最初から最後まで処理することになる。近年の AI（Artificial Intelligence: 人工知能）の普及もあり，計算機は自律的に考え，処理するような機械を連想する人もいるかもしれない[2)]。計算機は「計算」機である。そんな計算機が言語と自律的に向き合うために必要な要件がいくつかある。ここでは，記述されていること，決定的であること，実時間であることの 3 点を確認しておこう。

計算機の初期状態は空っぽである。そして記述されていないもの，データ化されていないものは，計算機は扱えず，「ない」ということさえ「ない」と記述することになる。人であれば当たり前の「犬は動物としての性質をもっている」といった常識も記述しなくてはならず，「日本語の修飾の方向は，規範的には前から後ろである」といった文法知識は，なおさらである。

計算機は与えられた手続きに従って処理をする。その手続きは決定的でなければならない。処理が進んでいくうちに選択を迫られた場合には，優先順位など選択基準が必要である。「良い方を選ぶ」といったことは，計算機は「良い」がわからないので，なにも起こらない。II でも触れる構文解析など，さまざまな場面で曖昧さが生じる。たとえば，「新規開業した福岡の駅」は，品詞の並びだけでみると「新規開業した」は「福岡」も「駅」も修飾し得る。実際には「駅」を修飾するのが妥当で，それを計算機が決定できるよう，その仕組みや知識も記述しておかなければならない。

最後に，計算機の処理に関する複雑性である[3)]。処理は，現実的な時間（実時間）で終えられ

なければならない。構文解析など多くの処理は一種の探索問題で，難しさのクラスや解き方によっては本質的に実時間では終えられないこともある。それでは実用的とはいえず，実時間に抑えられるよう工夫しなければならない。

計算機を言語の処理に向き合わせるには，このようにいくつかの要件を充足する必要がある。一方，計算機にはわれわれ人にはない強みもいくつかある。2点示しておく。ひとつは，人のような勘違いも体調による好不調もない，精確で高速な計算ができることである。もうひとつは膨大な情報を記憶，記録し，扱うことができることである。計算機は，言語に向き合う際に，これらの特性を活かした，人とはまったく異なるアプローチを採ることができる。

Ⅲ　言語のモデル

1　自然言語処理における言語のモデル化

自然言語処理が，言語をときにどのように捨象し，近似していくのかをみていくことで，人の言語を明らかにする立場との違いを強く感じることができる。そこで本節では，言語モデルを2つ示す。表現を直接生成する *n*-gram 言語モデルと，構文構造を介して表現を生成する確率文脈自由文法である。

これらの言語モデルは，いずれも表現を生成する言語モデルである。生成の言語モデルは大胆な仮定を置くことで，見通しよく設計することができることが多い[4)]。これらの言語モデルを素直に生成だけではなく，逆方向の解析にも使うことも多く，むしろ解析が主眼となっていることもある。生成の言語モデルによる解析は，与えられた表現が，どのような過程で生成されるのかを求めるわけである。自然言語処理が長らく，生成よりも解析の方に重点が置かれていたこともあり，解析の精度をもって生成の言語モデルの有効性が論じられることも，やや特異に映るだろう。

2　表現を直接生成する言語モデル

n-gram 言語モデルは，直接，語を確率的に発生させ，文を生成するものである。まず次のような文の断片が発生している場面を考えてみよう。

(1)　　温泉で有名な草津は $\boxed{x}$

この表現に続く語 x はどのようなものだろうか。単純化のために x の候補は，「群馬」「滋賀」「計算機」の3つとする。すると，次のようなことを直感するだろう。

(2)　　群馬は続きそう，滋賀は後の続き方によっては出るかもしれない，
　　計算機はまず出ない

ここまでを確率で記述し直してみよう。「温泉で有名な草津は」という表現が発生している下で，具体的な語 x が続く確率は，次のような条件付き確率である（付録A）。

(3)　　$p(x \mid$ 温泉で有名な草津は$)$

x は「群馬」「滋賀」「計算機」のいずれかで，(2)の傾向は次のように表すことができる。

(4)　p(群馬 | 温泉で有名な草津は) ＞ p(滋賀 | 温泉で有名な草津は) ≫ p(計算機 | 温泉で有名な草津は)

(4)が成り立っている下で，「温泉で有名な草津は」のあと，サイコロを振る要領で確率的に語が選ばれるならば，確かに「群馬」が続きやすくも稀には「滋賀」も出るし，「計算機」はまず続かない。これは，当時点までに出現した表現によって続く語が規定され，サイコロを振るように確率的に語列が生成されていく，という確率現象である。これが，*n*-gram 言語モデルにつながる基本的な考えである。人が物事を考えながら，言葉を選び，表現を紡ぎ出す，という言語活動とはずいぶんとかけ離れている。だが，生成される表現だけに注目するならば，似たような振る舞いを期待できそうでもある。

次に，“*n*-gram” を説明する。*n*-gram とは，適当な計数単位の *n* 連鎖である。「温泉で有名な草津は群馬にある」という文を例に，計数単位を語，*n*=4 の場合を考えてみる。

(5)　　＃＃＃温泉 で 有名な 草津 は 群馬 に ある ＃

便宜的に，語間に空白を入れた分かち書きをしている。‘#’ は文頭もしくは文末を表す特別な語である。この文が含む 4-gram をすべて列挙すると，次の 9 組となる。なお，*n*-gram 内の ‘-’ は語の区切りである。

(6)　　###-温泉, ##-温泉-で, #-温泉-で-有名な, 温泉-で-有名な-草津, で-有名な-草津-は, 有名な-草津-は-群馬, 草津-は-群馬-に, は-群馬-に-ある, 群馬-に-ある-#

n-gram は，語分割された言語データから非常に簡単に得られること，そして近い語間の共起は得られるものの，遠くの共起は失われることがわかる。

n-gram 言語モデルは語を基本単位とした場合の，語列の生成モデルである。語の確率的な生成は，直前の $n-1$ 語によって規定される。たとえば，4-gram 言語モデルでは，(1)の ***x*** は，前の 3 語「有名な草津は」の影響を受け，次のような条件付き確率に基づき決まる。

(7)　　$p(x \mid$ 有名な, 草津, は$)$

ここで，条件部には 3,2,1 語前の語がカンマ区切りで並んでいて，4 語以上前の「温泉で」など欠けていることに留意されたい。4-gram 言語モデルでは，4 語以上前の情報は失われるのである。

最初に戻って，文頭から実際に文が生成される過程を確認してみよう。文の始まりは “＃＃＃” である。

(8)　　$p(x \mid \#, \#, \#)$

すべての語に対して(8)の確率が与えられており，確率的に x が選ばれる。当然，(8)が高い x が発生しやすいが，ときには低い確率の語でも発生することはある。「温泉」が出たとしよう。

(9)　　### 温泉

ここで，下線と四角はわかりやすさのために付したもので，下線を付した語列が条件付き確率の条件部，四角に囲った語が確率的に発生した語である。(9)の次は $p(x \mid \#, \#, 温泉)$に基づいて語が発生する。x は「で」だったとしよう。すると，その次は $p(x \mid \#, 温泉, で)$に基づいて語が発生する。これを繰り返していくと，そのうち文末を表す # が発生し，文が終わる。

温泉 で 有名な
温泉 で 有名な 草津
温泉 で 有名な 草津 は
温泉 で 有名な 草津 は 滋賀
温泉 で 有名な 草津 は 滋賀 に
温泉 で 有名な 草津 は 滋賀 に は
温泉 で 有名な 草津 は 滋賀 に は ない

(10)　　### 温泉 で 有名な 草津 は 滋賀 に は ない #

このようにして「温泉で有名な草津は滋賀にはない」という文が生成される。仕組み自体は単純で，計算機で容易に取り扱えそうなことがわかる[5)]。

結局，n-gram 言語モデルは，語彙と文頭･文末の数だけ語の目をもつ，条件（$n-1$ 語の語列）の数分のサイコロ（確率 p に相当）を作って，それを振って言語表現を生成する装置である。私たち，人の言語生成が，このようなサイコロを振るような過程でないことは自明であろう。しかし，人が産出する膨大な表現の集合を確率現象の結果と見立てる仮定は，とりわけ動くシステムを現実的な時間・空間的資源の下で設計，構築する工学的視点に立てば合理的でもある。自然な表現を生成し，一見すると知的な振る舞いで，2022 年から大いに注目を集めている対話システムに ChatGPT がある。この ChatGPT の技術的中枢のひとつは，この n-gram 言語モデルである。なにか構文や意味に対する認知的な処理など介しておらず，直接，表現を生成している。

前項で述べたように，自然言語処理では生成モデルをまず構築し，それを解析に用いるのも常套である。生成モデルによる解析では，ある表現が与えられたときに，その表現が言語モデルで生成される過程を求めることになる。n-gram 言語モデルを解析に使う場面を考えてみる。解析なので，文などの表現が与えられて，そこに含まれる情報を得る。この n-gram 言語モデルが表現から得られる情報は，生成に適用される確率 p の列と，語列の発生確率である。これまでと同様に，4-gram 言語モデルで考えてみよう。「草津 は 滋賀 に も ある」という文の発生確率は次のようになる。

$$
\begin{aligned}
&p(\text{草津} \mid \#, \#, \#) \times p(\text{は} \mid \#, \#, \text{草津}) \times p(\text{滋賀} \mid \#, \text{草津}, \text{は}) \\
&\quad \times p(\text{に} \mid \text{草津}, \text{は}, \text{滋賀}) \times p(\text{も} \mid \text{は}, \text{滋賀}, \text{に}) \\
&\quad \times p(\text{ある} \mid \text{滋賀}, \text{に}, \text{も}) \times p(\# \mid \text{に}, \text{も}, \text{ある})
\end{aligned} \tag{11}
$$

これだけだと，*n*-gram 言語モデルが算出する表現に対する発生確率が役立つ場面が想像しにくいかもしれないので，応用の一例に仮名漢字変換を考えてみよう[6)]。

(12)　みずうみをじてんしゃでいっしゅうした

単純化のため，このひらがな列の変換候補が，次の 2 つに限られているとする。

(13)　a. 湖を自転車で一周した
　　b. 湖を自転車で一蹴した

それぞれの表現の発生確率のうち，異なる部分は次の部分である。

(14)　a. $p(\text{一周した} \mid \text{を}, \text{自転車}, \text{で}) \times p(\# \mid \text{自転車}, \text{で}, \text{一周した})$
　　b. $p(\text{一蹴した} \mid \text{を}, \text{自転車}, \text{で}) \times p(\# \mid \text{自転車}, \text{で}, \text{一蹴した})$

p の推定結果次第ではあるものの，大方，(14b)は(14a)に比べ，きわめて小さな値になることが予想される。変換結果には，より発生確率が高い方を選択する方が合理的であろう。したがって，計算機は(12)の変換結果として(13a)を選択することになる。

3　構造を介して生成する言語モデル

(1) 文脈自由文法

言語の構造にはさまざまなものがあるが，ここでは構文構造に焦点をあて，文脈自由文法（Context Free Grammar: CFG）という言語モデルを取り上げる。CFG は情報学でも形式言語理論で論じられる重要で，身近な枠組みである。そんな CFG は，自然言語処理の基礎研究で注力されている構文解析において，長い間，有力な道具立てのひとつとして活用されてきた。理論言語学との相性の良さのほかに，CFG 上の構文構造を高速に求める手続きが数多くあることも理由である。現実的な時間で処理が完遂することは，自然言語処理において非常に重要なのである。

CFG の文法 G は次のような 4 つ組で与えられる。

(15)　$G = (N, \Sigma, R, S)$

N は非終端記号の集合，Σ は終端記号の集合，R は書き換え規則の集合，S は開始記号で N のひとつである。いったん，非終端記号は表層に現れない品詞や句の文法標識，終端記号は語，

書き換え規則は句構造規則，開始記号は文や特定の句の文法標識というように，言語に密着させて考えるとわかりやすい。ここでは日本語の名詞句を生成する，ごく小さな CFG を例にしてみよう。

(16)　$G = (N, \Sigma, R, S)$
N = {VP, NP, PP, N, P}
Σ = {秋めいた, 古びた, 京都, 旅館, の}
R = {NP → PP N, PP → NP P, NP → VP NP,
VP → 古びた, VP → 秋めいた, NP → 京都, N → 旅館, P → の}
S = NP

R の書き換え規則は $X \rightarrow \alpha$ というかたちをしている。X は非終端記号で，α は非終端記号もしくは終端記号の列である。$X \rightarrow \alpha$ は，X を α に書き換え可能であることを意味する[7]。「文脈自由」というのは，この書き換え規則の適用条件に相応する→の左辺部分が非終端記号 1 つ，つまり規則の適用条件が参照できる情報は X のみであることを意味する。非終端記号の VP は動詞句，NP は名詞句，PP は後置詞句，N は名詞，P は後置詞で，R 内は典型的な句構造規則である[8]。非終端記号は書き換え規則による書き換え対象で，終端記号はこれ以上書き換えられることがない記号である。

CFG による生成は，S に対する書き換え規則を適用することから始まる。書き換わった記号列に非終端記号が含まれる場合は，適用できる書き換え規則のいずれかを適用する。これを繰り返し，書き換えた記号がすべて終端記号になった時点で終わる。その終端記号列が生成された表現となる。G からの生成例をいくつかみてみよう。

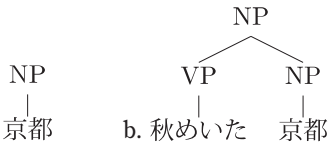
(17)　NP ⇒ 京都

ここで，$\beta \Rightarrow \gamma$ は，β が書き換え規則によって γ に書き換わった（β から γ が導出された）ことを表している。書き換え規則の→と，導出の⇒はよく似た記号だが，意味は異なるので留意されたい。β, γ は，非終端記号もしくは終端記号の列である。(17)内の下線と四角はわかりやすさのために付したもので，β の下線部分は書き換え対象となった非終端記号，γ の四角で囲った部分は書き換えで導出された記号列である。(17)は，開始記号であり非終端記号である NP に対して書き換え規則 “NP → 京都 ” が適用され，終端記号の「京都」が導出され，書き換えが終わる。その結果，「京都」という表現が生成されたことになる。もう少し長い例をみてみよう。

(18)　NP ⇒ VP NP ⇒ 秋めいた NP ⇒ 秋めいた 京都

非終端記号が非終端記号を含む列に書き換えられ，より長い表現が生成されている。これらの導出は理論言語学でよく描かれる樹形図や構文木にも対応する。本稿では構文木とよぶこととする。(17),(18)に対応する構文木は次のとおりである。

(19)

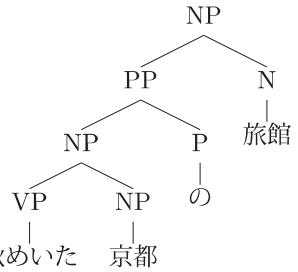


さらに長い例をみてみよう。

(20) NP ⇒ PP N ⇒ NP P N ⇒ VP NP P N ⇒ 秋めいた NP P N ⇒秋めいた 京都 P N
⇒秋めいた 京都 の N ⇒秋めいた 京都 の 旅館

構文木で表してみると理解しやすい。

(21)

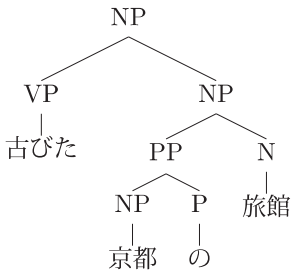


この表現は「京都が秋めいており，その京都にある旅館」という意味で，構文木もそれを反映したものとなっている。このように CFG は，*n*-gram 言語モデルにはなかった，表層的には現れない構造を作りながら，表現を生成する。

さて，「秋めいた」が「古びた」に変わった「古びた京都の旅館」もみてみよう。*G* による導出と構文木を併せて示す。

(22) a. NP ⇒ VP NP ⇒ 古びた NP ⇒古びた PP N ⇒古びた NP P N ⇒古びた 京都 P N
⇒古びた 京都 の N ⇒古びた 京都 の 旅館

b.



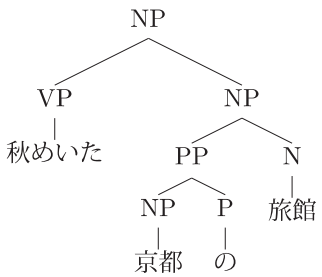
この表現は，「京都にある，古びた旅館」という意味である。「秋めいた京都の旅館」と「古び

た京都の旅館」の品詞の並びは，同じ “VP NP P N” である。「秋めいた」は直後の「京都」を，「古びた」は「京都」ではなく，その先の「旅館」を修飾している。両表現の意味の違いが，(21),(22) の構文木の違いにも反映されていることがわかる。

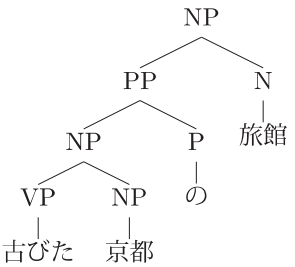
次は解析をみてみよう。CFG による解析とは，与えられた表現の導出，つまり構文木を求めることである。任意の表現に対して，所与の CFG が生成し得るものなのか，もし生成できるならばどのような導出が可能なのか，これらをすべて列挙する。そのための効率的な手続きはすでに提案されている。したがって，計算機が(16)の文法 G における「秋めいた京都の旅館」や「古びた京都の旅館」の(21)や(22)などの構文木は高速にみつけることができる。

ところで，両表現は G の下で，次のような構文木も構成し得る。

(23)



(24)



それぞれの意味をひねり出すならば，(23)は「京都にある，秋めいている旅館」で，(24)は「京都が古びていて，その京都にある旅館」となる。しかし，人なら「秋めいた」が「旅館」を，「古びた」が「京都」を形容するのは，表現のすわりなどから多少の不自然さを感じ，(23),(24)を (21),(22) に優先することは稀だろう。他方，計算機には CFG 以上の情報はないので，(21)と(23)，(22)と(24) の間に差はなく，解析結果は平等に示されることになる。

解析では，このような曖昧な結果に優先順位を付すことが重要である。そのためには適切な選定基準が求められる。どのような基準が考えられるだろうか。たとえば，左枝分かれ構造を優先する，つまり，修飾語はより近くの修飾可能な語と関係を結ぶ，というヒューリスティクス（経験則）もあるだろう。しかし，このような一律的な基準は生成と解析の対応が不明瞭になってしまったり，ここでの例では(22)のような構文木が決して選ばれないように，柔軟性に欠けたりする。この解決策のひとつが，次項で示す，構造を生成していく過程を確率現象とみなす CFG である(日高, 1995)。

(2) 確率文脈自由文法と語彙化

CFG を確率化した確率文脈自由文法（Probabilistic CFG: PCFG）は，CFG の書き換え規則にそれが適用される確率が加わったもので，次のような 5 つ組で定義される。

(25) $G' = (N, \Sigma, R, S, p)$

N, Σ, R, S は CFG のそれらと同様で，p は R に含まれる書き換え規則が適用される確率への関数である。具体的な CFG (16)の G に，p が加わった PCFG G' を考えよう。書き換え規則 “NP → PP N” が適用される確率は，p(NP → PP N)のように表すのが一般的である。これは，NP という非終端記号が発生しているときに，“PP N” という記号列に書き換える確率なので，次のような条件付き確率の方が明解であろう。

(26) p(PP N | NP)

本稿では前項の n-gram 言語モデルと合わせて，(26)の条件付き確率で記すこととする。さて，NP に対する G' の書き換え規則は，これ以外にも 2 つある。あらためてすべて挙げてみる。

(27) NP → VP NP
NP → 京都

確率は全事象の総和は 1 であるから，次式が成り立つ。

(28) $p(\text{PP N} \mid \text{NP}) + p(\text{VP NP} \mid \text{NP}) + p(\text{京都} \mid \text{NP}) = 1$

もし，p(PP N | NP)の割合が大きければ，後置詞句を伴った名詞句を生成しやすい。また，p(VP NP | NP)の割合が大きければ，動詞句を伴った長い名詞句に，p(京都 | NP) の割合が大きければ，語のみの名詞句になりやすい。

PCFG の生成をみてみよう。CFG 同様, 開始記号の非終端記号から始まる。G' であれば NP で，NP に対する書き換え規則が確率的に適用される。(18)同様に「秋めいた京都」が確率的に導出されたとしよう。

(29) NP ⇒ VP NP ⇒ 秋めいた NP ⇒秋めいた 京都

3 つの書き換え規則が適用された結果なので，(29)が G' から生成される確率は次のようになる。

(30) $p(\text{VP NP} \mid \text{NP}) \times p(\text{秋めいた} \mid \text{VP}) \times p(\text{京都} \mid \text{NP})$

結局，PCFG で構文木の確率は，導出に適用されたすべての書き換え規則の確率の積で与えら

れる。この確率が解析時の構文木の選択基準になる。

それでは，「古びた京都の旅館」を例に，PCFG の解析をみてみよう。PCFG でも CFG と同様，与えられた表現の構文木がすべて列挙され，それぞれの構文木に確率が付与される。「古びた京都の旅館」の(22)と(24)の構文木の確率は次のとおりである。

(31)
a. $p(\mathrm{PP\ N} \mid \mathrm{NP}) \times p(\mathrm{NP\ P} \mid \mathrm{PP}) \times p(\mathrm{VP\ NP} \mid \mathrm{NP}) \times p(\text{古びた} \mid \mathrm{VP})$
$\times p(\text{京都} \mid \mathrm{NP}) \times p(\text{の} \mid \mathrm{P}) \times p(\text{旅館} \mid \mathrm{N})$

b. $p(\mathrm{VP\ NP} \mid \mathrm{NP}) \times p(\text{古びた} \mid \mathrm{VP}) \times p(\mathrm{PP\ N} \mid \mathrm{NP}) \times p(\mathrm{NP\ P} \mid \mathrm{PP})$
$\times p(\text{京都} \mid \mathrm{NP}) \times p(\text{の} \mid \mathrm{P}) \times p(\text{旅館} \mid \mathrm{N})$

各項の順序は異なるものの積は交換可能な演算なので，結局，(31a)と(31b)は同値となり，(22)と(24)の選択はできない。こうなってしまうのは文法が文脈自由であることが一因である。「古びた京都の旅館」の(22),(24)を区分することになる書き換え規則は，“NP → VP NP” である。しかし，文法範疇間の関係を規定するのみで，確率の上でも依然として区別がつかない。人が(22)や(24)の構文木をみると，語間の関係を “NP → VP NP” で結びつけた上で(22)を排除することが内省される。そこで，非終端記号の，終端記号による細分化で，言語の上では語彙化に相応する精密化を施す(田辺他, 2000)。

非終端記号 X が，最終的に導出する終端記号列の中心的な記号を h とする。非終端記号を句などの文法標識，終端記号を語と考えれば，h は X の主辞である。X を $X(h)$ と表し直し，G' を再構成する[9)]。

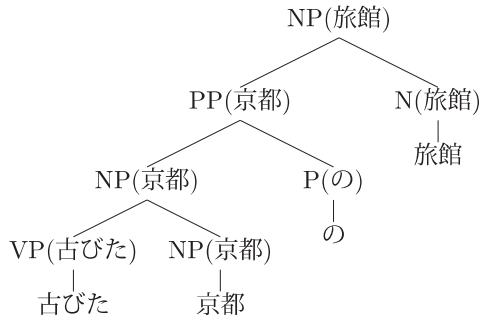
(32)
$G'' = (N, \Sigma, R', S', p)$
$R' = \{\mathrm{NP}(h) \rightarrow \mathrm{PP}(h')\ \mathrm{N}(h), \mathrm{PP}(h) \rightarrow \mathrm{NP}(h)\ \mathrm{P}(h'), \mathrm{NP}(h) \rightarrow \mathrm{VP}(h')\ \mathrm{NP}(h),$
$\mathrm{VP}(\text{古びた}) \rightarrow \text{古びた}, \mathrm{VP}(\text{秋めいた}) \rightarrow \text{秋めいた},$
$\mathrm{NP}(\text{京都}) \rightarrow \text{京都}, \mathrm{N}(\text{旅館}) \rightarrow \text{旅館}, \mathrm{P}(\text{の}) \rightarrow \text{の}\}$
$S' = \mathrm{NP}(\text{京都})$

R' 内書き換え規則の h, h' はいろいろなものに変わる変項で，特殊な書き換え規則が加わったようにみえる。実際には，すべての語を組み合わせ的に割り当てられた具体的な書き換え規則の略記である。したがって，CFG の構文木を高速に列挙する手続きは，この PCFG でもそのまま適用できる。再構成した文法 G'' の書き換え規則を確認しておこう。“$\mathrm{NP}(h) \rightarrow \mathrm{PP}(h')\ \mathrm{N}(h)$” は，後置詞句と名詞で構成される名詞句の主辞は名詞の主辞 h が継承される，という意図である。このように，非終端記号を細分化することで，終端記号である語の情報が構造内に伝搬する。(17)の「古びた京都」を導出する書き換え規則 “NP → VP NP” は，次のようになる。

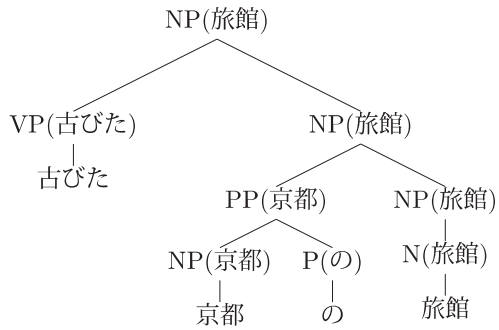
(33) $\mathrm{NP}(\text{京都}) \rightarrow \mathrm{VP}(\text{古びた})\ \mathrm{NP}(\text{京都})$

G'' で「古びた京都の旅館」の(24),(22)に対応する構文木は，それぞれ次のようになる。

(34)



(35)



(34),(35)の構文木間で異なる書き換え規則は，次のとおりである。

(36) a. NP(京都) → VP(古びた) NP(京都)
b. NP(旅館) → VP(古びた) NP(旅館)

「古びた」と「京都」,「古びた」と「旅館」の共起に関する違いが，書き換え規則で区別されることがわかる。(34),(35)の構文木の確率の差は，次のような大小に帰着される。

(37) $p(\mathrm{VP(古びた)\ NP(京都)} \mid \mathrm{NP(京都)}) \lesseqgtr p(\mathrm{VP(古びた)\ NP(旅館)} \mid \mathrm{NP(旅館)})$

私たちが感じる「『古びた京都』が『古びた旅館』に優先されない」という直感は，(37)の右辺が左辺より大きい，という確率上の関係に現れるはずである。その結果，計算機も(35)の構文木が選択できるようになる。PCFG には，h のより細かな導入の仕方や，書き換え規則の爆発的増大による p の推定（パラメタ推定）など, 興味深い課題がつづく。本稿の趣旨からは逸れるので，ここで留めることにする。

構造を介して表現を生成する CFG ベースの言語モデルは，より細かな言語に関する情報を扱えるようになる。ただ，人が抱く表現の妥当性に関する直感めいたものも，明示的かつ形式的に組み込まなければ，その効用を得ることはできない。自然言語処理における言語モデルは，言語に対する観察に基づいた直感であっても，枠組みへの一貫した記述が求められ，改変されていくのである。

Ⅳ　言語のベクトル表現

1　語の意味記述とベクトル

本節では語の意味記述と，ベクトルという単なる数値列の関わりをとおして，あらためて自然言語処理の言語への視座を考える。語の意味記述は，自然言語処理において，欠かすことができない基礎的で重要な事項である。さて，語の意味記述で私たちがまず想起するのは，辞書に記述されている語釈であろう。辞書の語釈は人が適切に解釈することを前提に，言語で人が記述したものなので，計算機が直接利用することは難しい[10]。自然言語処理は当初，語に対する性質，属性などの観点で，語の意味を分解的に表したり，語を意味的観点で分類したシソーラスを使ったりした方法が採られていた。ただ，膨大な数のすべての語で，それぞれ複雑な使用実態に即して，無矛盾な意味記述を人手で完遂することは難しいことが明らかとなってくる。そこで，近年，実用性という点でも注目されているのが，並行して取り組まれてきたベクトルによる表現である。

まず，ベクトルに関して確認しておこう。ベクトルは数値の列で，方向と大きさを含意する[11]。たとえば，縦と横のような2次元平面上のベクトル(2, –3)は，第1軸で2単位，第2軸で–3単位進む方向と，大きさは $2^2+(-3)^2=13$ の平方根になる。これに高さを加えた3次元ベクトル(2, –3, 10)ならば第1軸で2単位，第2軸で–3単位，第3軸で10単位進む方向と，大きさは $2^2+(-3)^2+10^2=113$ の平方根となる。4次元以上となると物理的なイメージが湧きにくくなるかもしれないが，次元は自然に増やしていくことができる。ベクトルは計算機にとって非常に取り扱いやすい形式である。また，2つのベクトルを考えた場合，それらの方向を比較することで，類似度を計算できることも利点である[12]。

ベクトルは物理的な場面を連想しやすいかもしれないが，言語にも相性が良い。例に単純化した文書検索を考えてみよう。まず，ベクトルの作り方を定義する。文書をその文書のなかで出現した内容語の頻度で特徴づけることにすると，辞書項目となっているようなすべての内容語の数の次元のベクトルで対応付けられる。つまり，内容語が1万あれば，1万次元のベクトルを文書に対応付けるわけである。文書 d のベクトル $\boldsymbol{v}_d$ は次のようなものである。

$$\boldsymbol{v}_d = (\underbrace{f_1, f_2, \cdots, f_{10000}}_{d \text{ 内の内容語の頻度分布}}) \tag{38}$$

ここで f_i は，便宜的にナンバリングされた i 番目の内容語の文書 d 内での頻度である。要するに，このベクトルは文書内の内容語の頻度分布である。文書検索時のクエリ q も小さな文書だと考えてしまえば，クエリも文書と同様にベクトル $\boldsymbol{v}_q$ に直る。クエリのベクトル $\boldsymbol{v}_q$ と，各文書のベクトル $\boldsymbol{v}_d$ をそれぞれ比較し，$\boldsymbol{v}_q$ に方向が似ている $\boldsymbol{v}_d$ 順に文書 d を検索結果として提示すれば良い。文書の数，ベクトルの次元数がたとえ膨大でも，計算機は高速に計算し，処理することができる。このようなベクトルで，語を表現する試みがある。

2　分布仮説と語のベクトル表現

語の意味をベクトルで表現する際に，最も基本的な考え方になっているのが分布仮説である。分布仮説は，語の意味が周辺にある語（共起する語；共起語；文脈語）によって決まる，というものである(Firth, 1957)。「勉強する」を含む 2 文を考えてみる。

(39)　a. 大学受験のため，苦手科目を集中的に勉強している
　　b. 2 つ買おうと思うので，もう少し勉強してもらえませんか

私たちは，(39a)と(39b)の「勉強する」の意味が，それぞれ「学ぶ」と「安くする」とすぐわかるだろう。その手がかりとなっているのが，同文内の共起語で，分布仮説にも通じる。

この分布仮説が効くような，非常に素朴な語のベクトル表現を考えてみよう。前項の文書検索のときのようにベクトルを定義するための語彙を定め，次元を対応させる。次元数は n，共起語を同文内に出現した語とする。語 w のベクトル $\boldsymbol{v}_w$ は，w と共起語との頻度を，共起語が対応する次元に割り当てたものとすれば，一般形は次のようになる。

(40)　$$\boldsymbol{v}_w = (f_1, f_2, \cdots, f_n)$$

ここで f_i は次元 i に対応する語と w の共起頻度である。このようなベクトルは，語分割された言語データがあれば容易に構成できる。たとえば，「勉強する」のベクトル $\boldsymbol{v}_{\text{勉強する}}$ は，「受験」や「科目」，「買う」や「もらう」などに対応する次元では 0 ではない値，共起しない語や共起が観測されなかった語に対応する次元では 0 が入った，次のようなベクトルとなる。

(41)　$$\boldsymbol{v}_{\text{勉強する}} = (0, \ldots, \overset{\text{受験}}{100}, \cdots, \overset{\text{買う}}{50}, \cdots, \overset{\text{科目}}{20}, \cdots, \overset{\text{もらう}}{15}, \cdots, 0)$$

これも結局，言語データから得られた共起語の頻度分布そのものである。分布仮説を認めれば，語の意味は共起語で規定されるので，間接的にその語の意味を表したことになる。文書検索のとき同様，ベクトルは他のベクトルとの類似度が容易に計算できる。あらゆる語間で意味の類似性が与えられるので，汎用性も高い。

共起語の頻度分布のベクトル表現は，分布仮説に基づき，人がベクトルを作成するための語彙や次元を適当に決め，値は素直な頻度で素朴に構成したものである。ここで，当然，そんな素直な頻度でよいのか，適切な次元数はいくつか，表記形か原形か，同義語・類義語の扱い，言語データから観測されない共起の可能性はどう考えるべきか，ほかにも考慮すべきことがあるのでないか，と疑問は尽きない。こういった事柄にもさまざまな取り組みがある。それが近年は，ほとんどを計算機が吸収してしまうような，各語のベクトル表現を自動的に構成する方法が提案され始めている。そして，そのようにして得られた語のベクトル表現が実際に成果を上げ始めている。word2vec とよばれる成果のごく一部を示す(Mikolov, T. *et al.*, 2013)。図 1 は英語を対象としたもので，1,000 次元のベクトルを 2 次元に圧縮したものである。左に国，右に首都が布置されている。図 1 の各国からそれぞれの首都に引かれたベクトルは，「国」を「首都」

に変換するものになっていることもわかる。図には現れていない，たとえば“U.K.”に対するベクトルに，次のような演算を加えると，“London”のベクトル表現が得られることも期待できる。

(42) $$\boldsymbol{v}_{\text{U.K.}} + \underbrace{\boldsymbol{v}_{\text{Tokyo}} - \boldsymbol{v}_{\text{Japan}}}_{\text{国から首都への右に引かれたベクトル}} \sim \boldsymbol{v}_{\text{London}}$$

国や都市がベクトルで表現できているだけではなく，国から首都という意味的な関係もベクトルで表すことができ，意味が演算できることが含意される。

このような意味のベクトル表現が得られる過程を，日本語の作例で簡単に説明する。次のような空所を埋める語を予測する問題を考える。

(43)　大学受験のため，苦手科目を集中的に $\boxed{x}$

このような空所予測問題を膨大に作成し，計算機に解かせるのである[13)]。現在のAIの基盤となっている深層のニューラルネットワーク（Neural Network: NN）で，空所予測問題を解くNNを構成する。このNNのなかで，語は予測に寄与するベクトル表現へ自動的に直されていく。計算機は基本的には共起語をヒントに解こうとするので，まさに分布仮説の実態がベクトル表現に現れてくる。

このように構成されたベクトルは，あくまでも空所予測問題の精度向上のための副産物である。それにもかかわらず，たしかに語の意味などを反映した，今までの人による意味記述にはない形式と機能を有している。ChatGPTでも，語をこのようなベクトルで扱っており，各処理の性能向上に寄与している。一方で，大きな次元の下，個別の次元，次元ごとの値，語それぞれのベクトルは，人が解釈することはきわめて難しい。こういったものに対する解釈支援の研究も活発になりつつあるものの，言語そのものに注目し，観察と内省に基づいて，語の意味を真正面から考えていく視座とは大きく異なる。いったん人の手をずいぶんと離れたところから意味にアプローチする研究でもある。

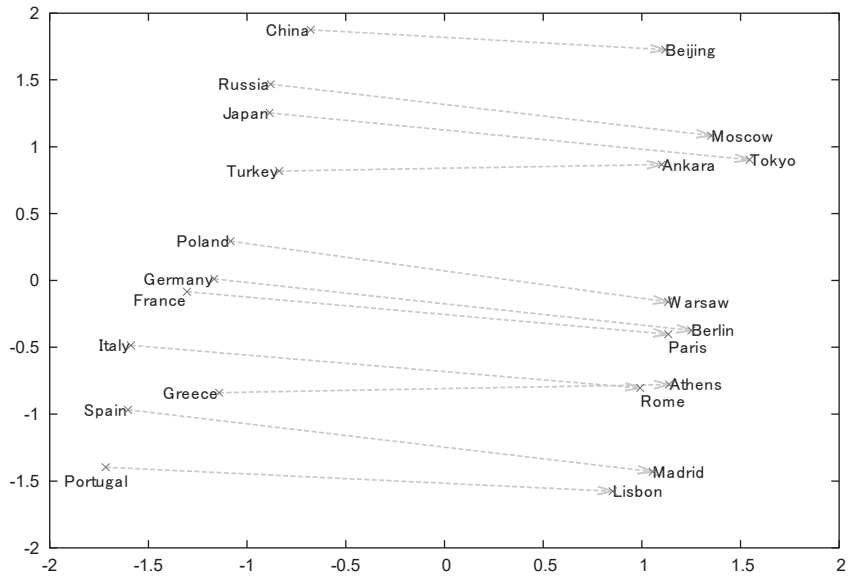


図 1　国と首都のベクトル表現(Mikolov, T.(2013) の Figure 2)

Ⅶ　おわりに

本稿は言語に向き合う主体が計算機となった際，どのような条件が求められるのか，そして自然言語処理の言語に対する視座，研究姿勢を，簡単な事例とともに示した。工学や情報学におけるシステム実装を念頭にした諸制約と精度指向など，人文学などの言語に対する価値観からすると著しく異質にみえるはずである。ただ，おおよそすべての自然言語処理研究者は，言語そのものにも強い関心をもっていることは間違いなく，このようなバイパス的な言語への接近で，言語の新しい知識を得ることも期待している。

注

1）重要な自然言語処理のテーマである，辞書やコーパスの言語資源の整備などは，必ずしもこの軸に素直にはのらない。

2）AI は物事を基本的に分類問題に帰着し，判断している。なにか生理的な実態がある知能を構成しているわけではない。

3）複雑性には，本文で記した時間以外にも，処理の際に必要となる記憶空間などの空間的複雑さもある。

4）自然言語処理で言語モデルに対する大胆な前提を許容できる点も，人が人の言語に向き合う言語学との違いを感取できる。

5）膨大な数になるパラメタ（確率を与える p）をどのように推定するのか，類義語・同義語などをどう扱うべきか，といった課題がある。実際，自然言語処理は n-gram 言語モデルの枠組みよりも，これらの課題にほとんどの時間を要したといっても過言ではない。

6）実装され，実用されている仮名漢字変換は，変換結果の選択に，文法的な制約など，より多くの情報を活用している。この例はあくまでも n-gram 言語モデルを理解する上での作例である。

7）$X \rightarrow \alpha$ は右辺から左辺からみると，α が X を構成する，とも解釈できる。たとえば，“NP → PP N” は「後置詞句と名詞が合成して名詞句になる」とみることができる。

8）本稿では G を小さくするために，説明に必要なものに限っている。対象を名詞句だけに限ったとしても書き換え規則（句構造規則）や文法標識も，本来ならば，より細かく設定することになる。
9）とくに語間の共起制約の観点から h を導入した文法は，語彙共起制約文法とよばれる。
10）語釈から語の意味に関する知識獲得を試みる研究もある。
11）日常会話で「（なにか物事と物事の）ベクトルが似ている」といった表現がみられることがある。この場合は，方向という性質が転じているようである。
12）2 ベクトルのなす角が 0 度のときもっとも似ていて，90 度は直交していて独立，180 度は真逆である，ととらえられる。ベクトル間のなす角は次元に関係なく 0 ～ 360 度で，内積を通して容易に計算できる。イメージが湧きにくい場合は，2 次元で考えてみると良い。
13）インターネットの普及と，テキスト蓄積技術の進展に伴い，想像を絶する言語テキストが利活用できるようになっている。空所予測の問題と解答は，そんなテキストから機械的に大量に作成される。

参考文献

Firth, J. (1957). A Synopsis of Linguistic Theory, 1930-55. *Studies in Linguistic Analysis*. Special Volume of the Philological Society, 1-31.

日高達 (1995).「確率文法」,『情報処理学会誌』, 36, 2, 169-176.

黒橋禎夫 (2023).『自然言語処理〔三訂版〕』, 放送大学教育振興会 .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 26, 3111-3119.

長尾真(編) (1996).『自然言語処理』, 岩波書店 .

岡崎直観・荒瀬由紀・鈴木潤・鶴岡慶雅・宮尾祐介 (2022).『自然言語処理の基礎』, オーム社 .

田辺利文・冨浦洋一・日高達 (2000).「係り受け文脈自由文法とその日本語への適用」,『情報処理学会論文誌』, 41, 1, 36-45.

付録

A　条件付き確率

n-gram 言語モデルや PCFG で記述する条件付き確率について簡単に説明しておく。事象 x が発生する確率を $p(x)$ と記し，x の全事象は n 個（$x_1, x_2, \cdots, x_n$）とする。$p(x)$ に関して，次のようなことが成り立つ。

(44)
i. $p(x) \geq 0$
ii. $p(x) \leq 1$
iii. $p(x_1) + p(x_2) + \cdots + p(x_n) = 1$

中高の教科書によく出てくる例で，物理的に偏りのない正六面体で，ランダムになるように正しく振るサイコロであれば，各目が出る確率は $p(x)$=1/6 である。

統計的厳密さは下がるかもしれないが，もう少し身近な例でも考えてみよう。気象庁の 1991-2020 年のデータによると，猛暑日（最高気温が 35℃以上の日）は観測地点全体で全国平均 7.5 日だそうである。国内で猛暑日にあたる確率を p(猛暑日)とすると, 7.5 を 365 で割った p(猛暑日) ～ 0.020 程度と推測される。さて，地点をより限定した場合，この確率はどうなるだろうか。夏の暑さでよく知られた「京都」に着目した際，猛暑日にあたる確率は当然，p(猛暑日)とは異なるはずである。「京都にあって（京都にいる，ということがすでに発生済）」という意味の条件

が加わっている。このように条件が加わった確率のことを条件付き確率といい, 次のように表す。

(45) $p(y \mid x)$

縦棒の右の x が条件で, y が事象である。これまでの例でいえば「京都にあって」が x,「猛暑日」が y である。条件をバーの右に書くので留意されたい（高校数学では，(45)を $p_x(y)$ と書く）。気象庁のデータによると，京都では平均 19.3 日も猛暑日がある。すると，p(猛暑日 | 京都)〜 0.053 となる。なお，p(猛暑日 | 滋賀)〜 0.013，暑い印象があるかもしれない沖縄で猛暑日にあたる条件付き確率は p(猛暑日 | 沖縄)〜 0.0005 である。夏の気温に関しては，沖縄の方が京都よりも快適そうであることがわかる。条件付き確率は次のような関係にもある。

(46) $p(y \mid x) \times p(x) = p(x, y)$

左辺は「x が発生したときに y が発生する確率（条件付き確率）」と「x が発生する確率」の積で, それが右辺の「x と y が同時に発生する確率（同時発生確率）」に等しい。p(京都, 猛暑日)は「京都で猛暑日である」同時発生確率で，それは京都にある確率と，京都にあって猛暑日となる条件付き確率の積である。さらにもう少し複雑にしてみよう。「京都で猛暑日で雨の日である」確率 p(京都, 猛暑日, 雨)を条件付き確率で書いてみよう。

(47) $p(\text{雨} \mid \text{京都}, \text{猛暑日}) \times p(\text{猛暑日} \mid \text{京都}) \times p(\text{京都}) = p(\text{京都}, \text{猛暑日}, \text{雨})$

ここから，たとえば n 個の事象の同時に発生する確率は，条件付き確率と次のような関係になることがわかる。

(48)
$$\begin{aligned} &p(x_n \mid x_1, \cdots, x_{n-1}) \times p(x_{n-1} \mid x_1, \cdots, x_{n-2}) \times \cdots \times p(x_2 \mid x_1) \times p(x_1) \\ &\quad = p(x_1, x_2, \cdots, x_n) \end{aligned}$$