

Chapter 0

人工知能をはじめよう

データサイエンティストにとって、人工知能(AI)は必須の知識です。そこでここでは、人工知能とは何か、なぜ人工知能を学ぶのか、そして人工知能はどのように発展してきたのかを見ていきましょう。

0.1

人工知能は知らない間に — これもAI、あれもAI、きっとAI

※人工知能というコトバ

人工知能、これは英語の「Artificial Intelligence」の訳語です。人工知能という言葉は、20世紀半ばごろにアメリカではじめて使われました。その後人工知能の技術はどんどん発展していき、今では私たちの生活になくてはならない技術になっています。もっと言えば、人工知能の技術は、私たちの身近なところだけではなく、現代社会を支える重要なインフラ技術となっています(図0.1)。

※人工知能って何?

では、人工知能とは何でしょうか?実は、人工知能に対するキチンとした定義は、まだありません。人工知能研究の歴史は100年にも満たないため、まだ、何をもって人工知能とするかについて誰もが納得するような定義が得られていないのです。

それでも、図0.2に示した(i)および(ii)の2つは、「人工知能」の定義としてしばしば目にします。

図0.2の定義(i)は、人工知能をその言葉の通り、「人工の知能・知性」とする定義です。おそらく、多くの人はこの定義に賛成されるのではないのでしょうか。

この種の人工知能は、フィクションの世界の中には大昔から存在しており、

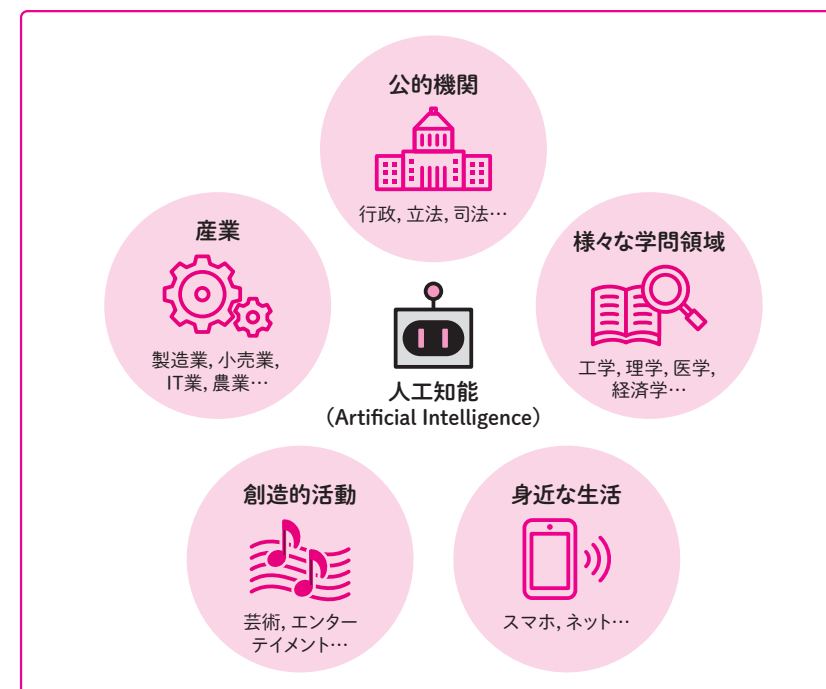


図0.1 人工知能の技術は、さまざまな分野で利用されている

- (i) 人工の知能・知性
- (ii) 生物や人間の知的挙動にヒントを得た、ソフトウェア技術

図0.2 人工知能の2つの定義

人形やモノがしゃべりだして知的にふるまうというおはなしが、神話や昔話に良く出てきます。近年では、人間に作られたロボットやアンドロイドが活躍するSFがたくさんあります。

Chapter 0

人工知能をはじめよう

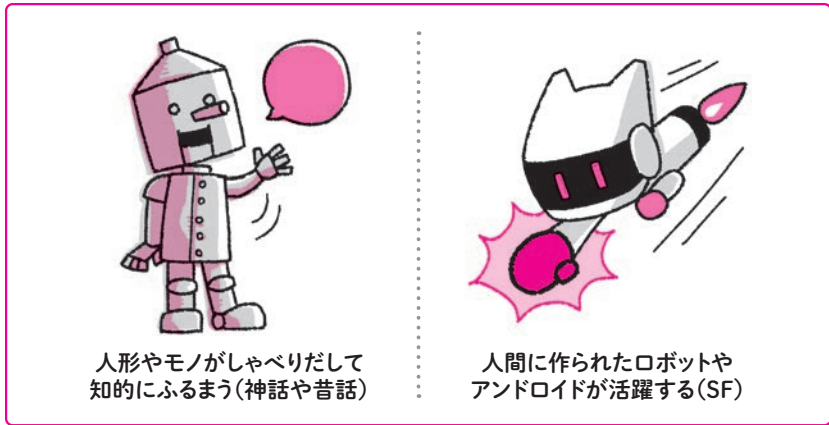


図0.3 定義(i) 人工知能は「人工の知能・知性」

ただ、残念ながら、現時点ではそういった「人工知能」はフィクションの世界の中だけの存在であり、現実には存在しないようです。

定義(i)に対して、定義(ii)はぐっと具体的な定義です。定義(ii)では、人工知能をソフトウェア技術、つまり、プログラムやデータの作り方であるにとらえています。ここでは、プログラムを作る際に、生物や人間の知的なふるまいを観察して、それをヒントにして賢いプログラムを作るやりかたを与えるのが人工知能である、と定義しています。

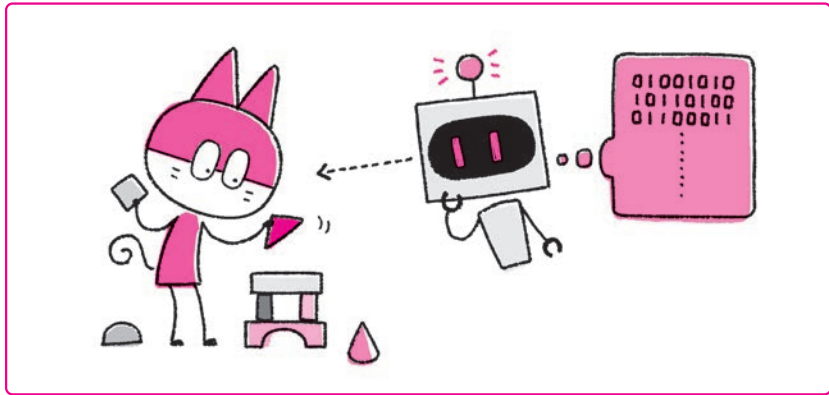


図0.4 定義(ii)人工知能は「プログラムの作り方」

★人工知能の応用例

人工知能の研究者や人工知能を扱う技術者の多くは、定義(ii)の、「人工知能は生物や人間の振る舞いにヒントを得たプログラミング技術である」とすることに、反対しないと思います。

実際、定義(ii)の立場に基づく人工知能研究によって、さまざまな応用システムが生み出されています。表0.1に、こうして生み出された人工知能の応用例を示します。

表0.1 人工知能の応用例

技術の名称	説明	応用例
自然言語処理	自然言語(日本語や英語など)をプログラムで処理する技術	音声によるスマホ操作 機械翻訳
探索	データのなかから目的とする項目を高速に探し出す技術	ネット検索 表計算ソフト
機械学習	機械すなわちプログラムが学習によって賢くなる技術	制御 自動運転 画像認識
深層学習(ディープラーニング)	大規模な人工ニューラルネットによる機械学習技術	画像理解 文書理解
生成AI	深層学習を文書や画像の生成に利用する技術	チャット 画像生成動画生成

これらについては、次章で改めて取り上げます。

0.2 人工知能はなぜ必要なのか? — 人工知能を学ぶと社会がわかる、今がわかる、そして未来がわかる

★ソフトウェア技術としての人工知能

人工知能研究の成果は、ソフトウェア技術としてさまざまなプログラムの作成に生かされています。表1に示したような応用例の他にも、実は、ワープロ(かな漢字変換)や、Java言語やPython言語のようなプログラミング言語、あるいは専門家の振る舞いを真似て人間の仕事を助けるエキスパートシステムなど、現代社会の中であたりまえに使われているさまざまなソフトウェアに、人工知能の研究成果が生かされています。

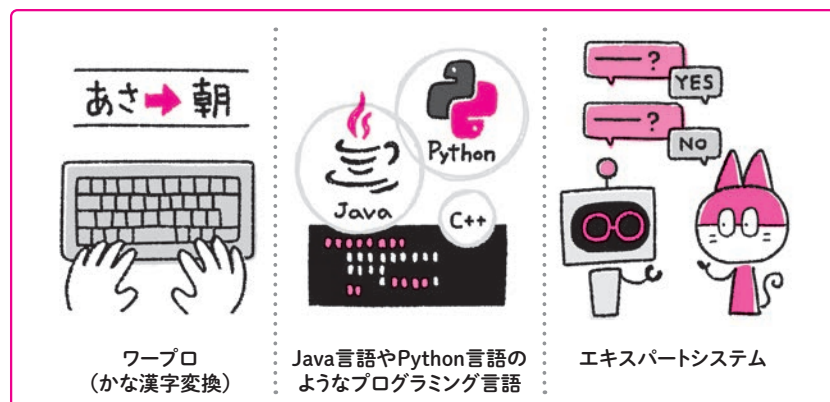


図0.5 社会をささえる人工知能研究の成果

ソフトウェア技術としての人工知能を学ぶことで、社会を支えるソフトウェアについて理解することができ、結果として、現在そして未来の社会を知ることができます。

※ データサイエンスのツールとしての人工知能

データサイエンティストの立場からは、人工知能はデータ解析の重要なツールとしてとらえることができます。データサイエンスの世界には、重要な武器が2つあります。一つは**統計学**です。統計学は、従来の数学の枠組みから、データの解析や評価を行います。もう一つの武器は**人工知能**です。人工知能、特に機械学習の手法は、統計学だけでは解析しきれないようなデータに対しても、統計学とは異なる立場から解析を行うことが可能です。この意味から、人工知能はデータサイエンティストが使う重要な武器であると言えるでしょう。

※ 人工知能は数学？ —データサイエンスは文理融合、数学は必要

データサイエンスはデータを扱う学問です。データには、数字で表現されたものの他に、例えば文字や文章、画像、音声、動画などさまざまな形式があります。しかしデータを扱うときには、データを数字として処理することが基本となります。したがって、この時、統計学を適用するにせよ、人工知能を用いるにせよ、どうしても数学の知識が必要になります。

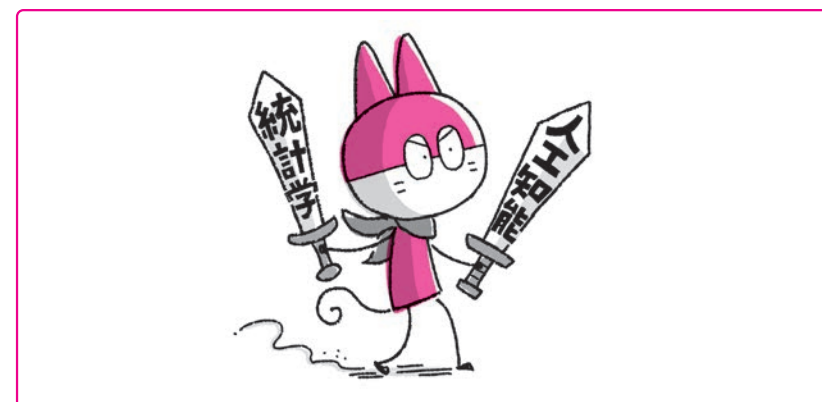


図0.6 人工知能はデータサイエンティストが使う重要な武器の一つ



図0.7 データサイエンスには数学が必須

このような意味から、データサイエンスは「文系と理系の性質を併せ持った文理融合の学問である」、と言われています。

ただし本書では、数学による表現は最小限に留めて代わりにプログラムによる表現等を用いることで、なるべく数式は使わないで説明を進めることにします。

0.3

とりあえず、歴史
—— 人工知能はここから学びましょう

★20世紀中ごろ 人工知能の誕生

「人工知能」という言葉は20世紀中ごろに初めて使われるようになりましたが、人工知能の考え方自体はそれ以前からありました。ただ、人工知能を実現するためにはコンピュータが必要ですから、人工知能を具体化する技術はコンピュータが発明された以降に発達しました。

コンピュータが発明されたのは、実は20世紀中ごろです。世界最初のコンピュータがなんであるかは議論の分かれるところですが、例えばアメリカで開発されたENIACというコンピュータは1946年に稼働を始めています。ただしアメリカでは、それより以前の1942年にABCというコンピュータが実験的に稼働しています。イギリスやドイツでも、同時期にコンピュータが相次いで稼働しています。

こうした中、1950年にイギリスの数学者アラン・チューリングは、コンピュータに知能・知性があるかどうかについて考察した論文を発表しています。チューリングはこの論文の中で、チャットのような手段で行う会話において、会話の内容からは相手が人間かコンピュータか区別できないならば、そのコンピュータは知能・知性を持っていると判断して良いと主張しています。この考え方による知能・知性の判断方法を、**チューリングテスト**と呼びます(図0.8)。

チューリングテストにはさまざまな批判もあります。例えば、人間と見間違えてしまうようなチャットボットは比較的簡単に作成することができます。しかし、1950年に発表されたことを考えると、チューリングテストはコンピュータの知能・知性についての初期の考察として重要です。

★1950年代 人工知能研究の黎明期

人工知能、Artificial Intelligenceという言葉が初めて現在の意味で使われるようになったのは、1956年にアメリカのダートマス大学で開催された**ダートマス会議**がきっかけであったと言われています。

ダートマス会議は、「会議」と言っても、コンピュータやソフトウェアを研

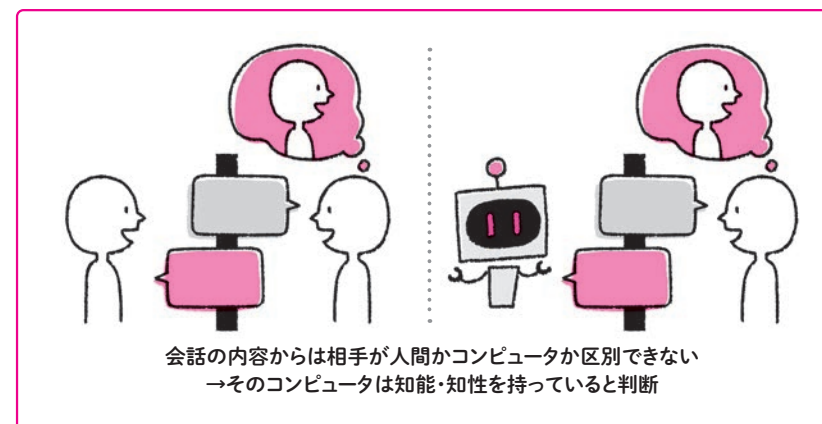


図0.8 チューリングテスト

究する研究者が集まって研究成果を発表する、夏季セミナーのような学術的集会でした。そこでは、知的なコンピュータプログラムの実現方法に関連するさまざまな研究が紹介され、研究者どうしが議論を重ねました。

ダートマス会議を主催した中心メンバーに、当時新進気鋭の研究者であった**ジョン・マッカーシー**や**マービン・ミンスキー**がいました。彼らはやがて人工知能研究の大御所となるのですが、ダートマス会議開催のためにマッカーシーが起案した提案書の中に、人工知能、Artificial Intelligenceという言葉が使われています。ダートマス会議は、人工知能研究を立ち上げて促進した、人工知能研究における重要な出来事であったと言えるでしょう。

★1960年代から1970年代 人工知能研究の発展

ダートマス会議の後、1960年代から70年代にかけて、人工知能研究は大きく発展します。例えば、現在生成AIとの関係で注目されている**チャットボット**の研究は、1960年代に行われた**ジョセフ・ワイゼンバウム**による**イライザ(ELIZA)**の研究が、その起源とされています。また1970年代には、英語で命令することでロボットアームを動かして積み木の世界を操作する**シュルドゥルー(SHRDLU)**というシステムが、**テリーウィノグラード**によって開発されています。

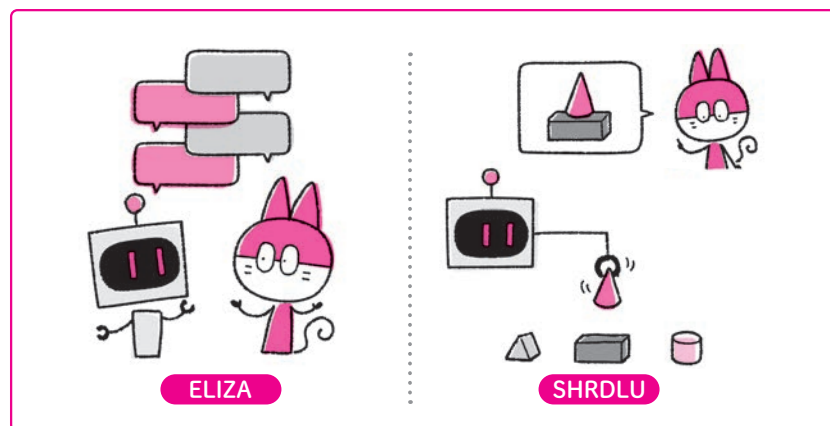


図0.9 ELIZAとSHRDLU

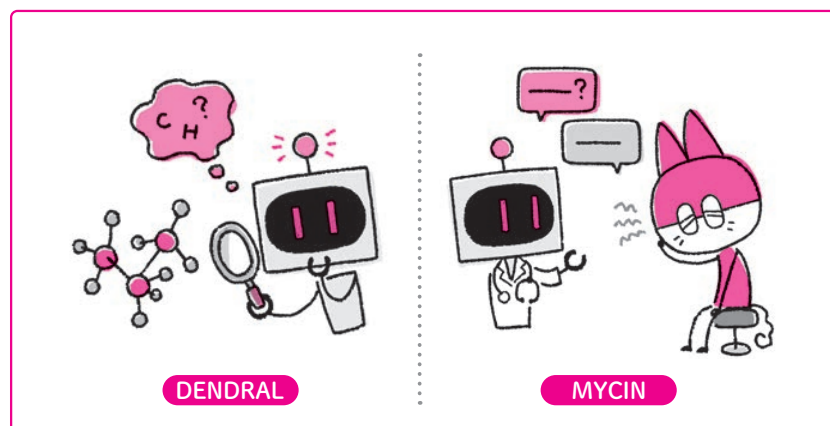


図0.10 エキスパートシステム

1970年代には、エキスパートシステムが実用化されています。世界初のエキスパートシステムと言われるデンドラル(DENDRAL)は1960年代後半に開発されました。DENDRALは、化学物質の組成を推論するエキスパートシステムです。また、1974年にはマイシン(MYCIN)と呼ばれる、感染症の診断に関するエキスパートシステムが開発されています。

以上のような成果を挙げたにもかかわらず、人工知能研究はその後「冬の時代」と呼ばれる低迷期を迎えます。これは、当時のコンピュータの貧弱な処理

能力では実用的な人工知能プログラムは作成しづらく、産業応用が進めにくかったことに起因すると思われます。

★1980年代

1980年代に入ると、人工知能研究は再び盛り上がりを見せます。例えば、70年代に開発されたエキスパートシステムが、医療や金融などの分野で実用化されます。また、身体を持ったロボットの研究も盛んになり、体を持つことが知能・知性の発現に必須だという**身体性認知科学**の考え方が示されました。

この時代には、**ニューラルネット**の研究が盛んに行われました。ここで言うニューラルネットは、生物の神経細胞の働きを数式でシミュレートした、人工ニューラルネットのことを意味します。

ニューラルネット自体は、実はコンピュータが発明されるよりも古い時代から研究されていましたが、1950年代に盛んに研究された後、一旦研究が沈静化していました。しかし1980年代には、工業的な応用も含めて、ニューラルネットの研究が盛んに行われるようになりました。これは、ニューラルネットの学習手法である**バックプロパゲーション(誤差逆伝播)**が広く知られるようになったことに加えて、コンピュータの性能が上がったことが理由です。

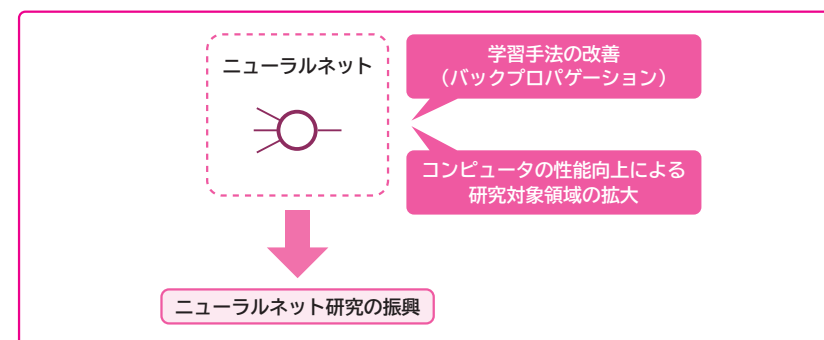


図0.11 ニューラルネット

この後、人工知能研究は再び冬の時代を迎えます。この時も、当時のコンピュータの能力で解決可能な人工知能研究の課題が一通り研究されつくされことで、人工知能研究は低迷期を迎えました。

★2010年代 深層学習の台頭

21世紀に入り2010年代に**深層学習**の有用性が示されると、第3次人工知能ブームとも呼ぶべき時代を迎えます。深層学習は、ニューラルネットを大規模化した**ディープニューラルネット**を使った人工知能の手法です。

深層学習は、初めは、**画像認識**、すなわち与えられた画像に何が移っているのかを判断する問題領域で、他の方法と比較して抜きん出た能力を示すことで注目されました。その後深層学習は画像認識だけでなく、音声の認識や自然言語処理への応用がなされ、それぞれの分野で高い性能を発揮しています。

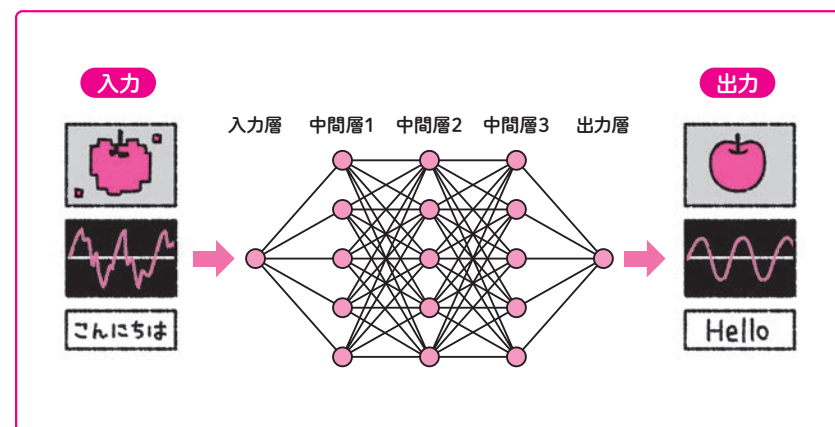


図0.12 深層学習

★2020年代 生成AIの誕生

2020年代になると、深層学習を利用した自然言語処理において、指示に従って自然な文章を生成することができる**ディープニューラルネット**が開発されました。これが**生成AI**です。ChatGPTやGemini、Copilotといった生成AIは、人間とのやりとりによってあたかも人間が書いたようななめらかな文章を生成することができます。さらに生成AIは、文章だけでなく、画像や動画なども生成することができます。

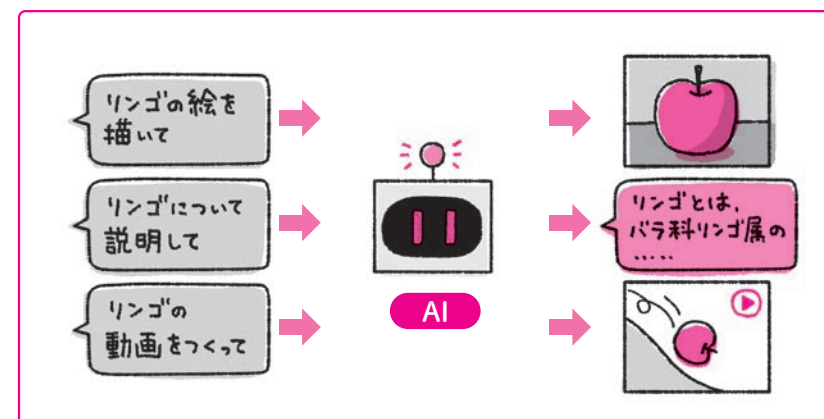
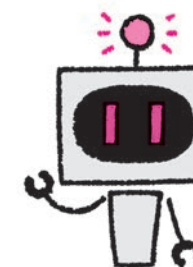


図0.13 生成AI

まとめ

- 人工知能という言葉には、「人工の知能・知性」という意味のほかに、「生物や人間の知的挙動にヒントを得た、ソフトウェア技術」という意味があります。
- 人工知能は現代社会を支えるソフトウェア技術であり、さまざまな分野で利用されています。
- データサイエンスの世界では、人工知能は統計学と並ぶ重要な技術です。
- 20世紀なかばに始まった人工知能研究は、ソフトウェア技術の分野においてさまざまな成果を挙げています。



Chapter 1

あれもこれもAI 生活に 密着する人工知能技術

人工知能技術は、社会を支えるインフラから私たちの身近な部分まで、さまざまな領域に浸透しています。この章では、こうした人工知能技術をいくつか取り上げて、人工知能技術の基礎を説明します。

1.1

「Hey Siri」で始まる自然言語処理 —— 自然言語(ヒューマン)とコンピュータの言語

★「Hey Siri」と「アレクサ、おやすみ」

私たちの生活に密着した人工知能の応用技術の一つに、**AI音声アシスタント**と呼ばれるソフトウェアがあります。具体的には、スマホで使えるシリ(Siri)や、スマートスピーカーと呼ばれるデバイスで利用できるアレクサ(Alexa)などがあります。AI音声アシスタントを用いると、音声でスマホやスマートスピーカーに話しかけるだけで、情報検索や機械の制御などの機能を利用することができます。



図1.1 「Hey Siri」と「アレクサ、おやすみ」

私たちはこうしたAI音声アシスタントを当たり前のものとして利用していますが、実はその裏では、さまざまな人工知能技術が利用されています。AI音声アシスタントが行っている、言語の処理に係る技術を、**自然言語処理**の技術と呼びます。自然言語処理は人工知能を構成する一つの領域です。

★自然言語と人工言語

自然言語処理について説明する前に、前提として、「自然」言語とは何かを説明します。人工知能の世界では、英語や日本語のような言語を**自然言語**と呼びます。自然言語は、人類やある種の生物の集団が、その歴史とともに自然発生的に生みだして発展させてきた、意思疎通や記録、あるいは思考のための手段です。単に「言語」と言うと、普通それは自然言語を意味します。

ちなみに、自然言語以外の言語、つまり自然発生的に生まれてきたのではなく人為的に作成された言語のことは、自然言語に対して**人工言語**と呼ばれます。例えば、C言語やPythonのような**プログラミング言語**や、人為的に作成された言語であるエスペラント語などは、人工言語の例です。

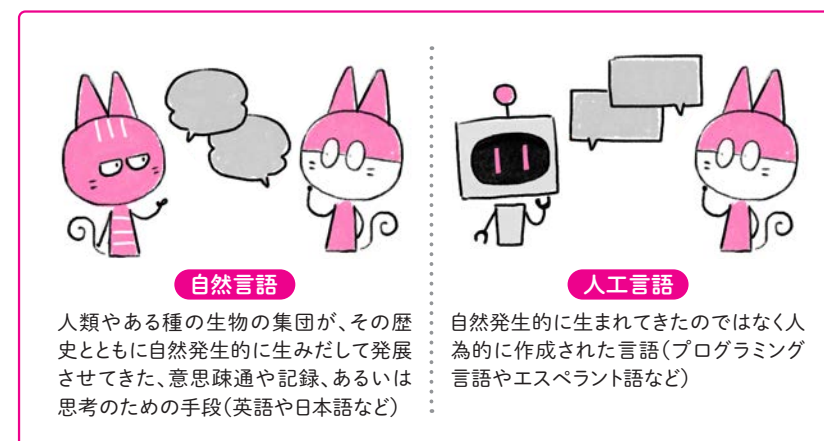


図1.2 自然言語と人工言語

人工言語の世界で自然言語処理技術というと、英語や日本語を処理する技術を意味します。コンピュータは本来英語や日本語を理解しませんが、自然言語処理技術を用いることで、コンピュータで英語や日本語を処理することができ

Chapter 1

あれもこれもAI生活に密着する人工知能技術

るようになります。

※自然言語処理技術の例

自然言語処理技術によって作成された自然言語システムの例を表1.1に示します。

表1.1 自然言語処理システムの例

名称	説明
音声入力	音声を聞き取って文字データに変換する。AI音声アシスタントの入力技術でもある。
AI音声アシスタント	音声でスマホやスマートスピーカーに話しかけるだけで、情報検索や機械の制御などの機能を利用することができる。
機械翻訳	自然言語で記述された言語を、他の自然言語に変換する。例えば英語のWebサイトを読み込んで日本語に翻訳したり、日本語で書いた文章を中国語に翻訳したりする。
文字読み取り (OCR)	手書きあるいは紙に印刷された文字を、カメラやスキャナを用いて画像として読み込み、そこに書かれている文字を判別して文字データに変換する。
チャットボット	人間が入力した文を解析して、その返答を人間に与える。無目的な対話を行うもの他に、特定の情報についての情報検索を行うなど目的を持った対話を行うチャットボットもある。
類似文書検索	利用者から入力された文書を手掛かりに、関連する文書を検索する。
文章の要約や生成	利用者の指示に従って、文章を要約したり、新たな文章を生成する。

以下、表1.1に示した自然言語処理システムについて説明します。

※音声入力とAI音声アシスタント

音声入力は、人間の発する音声をマイク等で捉えてコンピュータに取り込み、音声に含まれる自然言語の情報を抽出する技術です。音声入力を用いると、音声を聞き取って文字データに変換することで、キーボードやタッチパネルを経由せずに音声で文字を入力することができます。また、入力した情報に従ってコンピュータやスマホを操作することも可能です。

※機械翻訳

機械翻訳 (machine translation) は、英語から日本語、あるいは日本語から中国語への翻訳を、コンピュータプログラムが自動的に行う技術です。機械翻訳

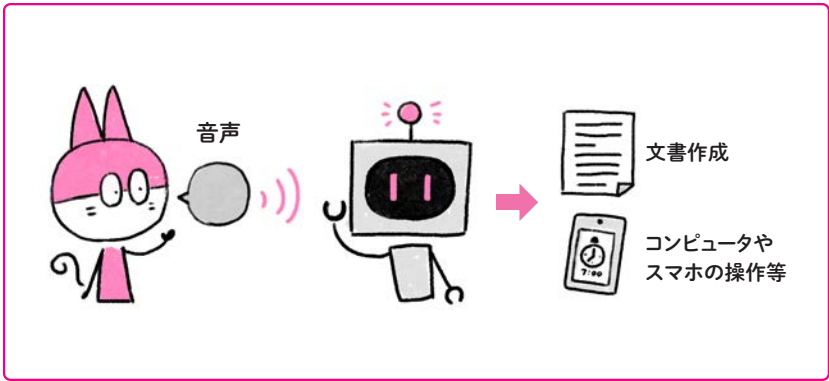


図1.3 音声入力

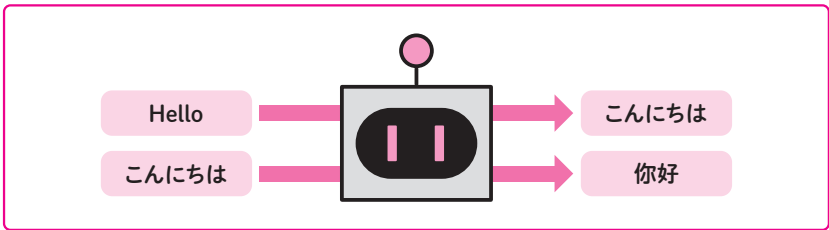


図1.4 機械翻訳

という言葉のうちの「機械(machine)」は、この場合にはコンピュータのプログラムと言う意味を持ちます。

※文字読み取り

文字読み取り (OCR, Optical Character Reader) は、カメラやスキャナを用いて読み込んだ文字の画像を解析して、文字情報を取り出す技術です。対象とする文字は、印刷された文字の他、人間が手書きした文字も含まれます。問題の難しさとしては、印刷された文字の方が易しく、手書き文字の認識はそれよりも難しい問題です。

文字読み取りの技術は古くから研究されていて、例えば郵便で使われる郵便番号の読み取りについては、1960年代から実用的に利用されています。郵便番号は手書きされるのが普通ですから、郵便番号の読み取り問題は印刷された

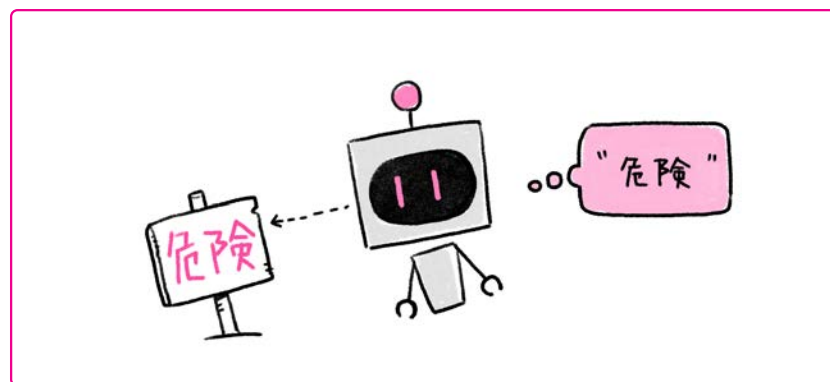


図1.5 文字読み取り(OCR)

文字の読み取りよりも難しい問題です。ただし郵便番号は0から9までの数字しか扱いませんから、たくさんの種類の文字を読み取らなければならない一般の文字読み取りの問題と比べると、その分だけ問題が易しくなっています。

※チャットボット

ワイゼンバウムのイライザに端を発したチャットボット研究は、現在、生成AIという形で大きな発展を遂げています。チャットボットの構成には、自然言語処理技術をはじめとするさまざまな人工知能技術が利用されています。

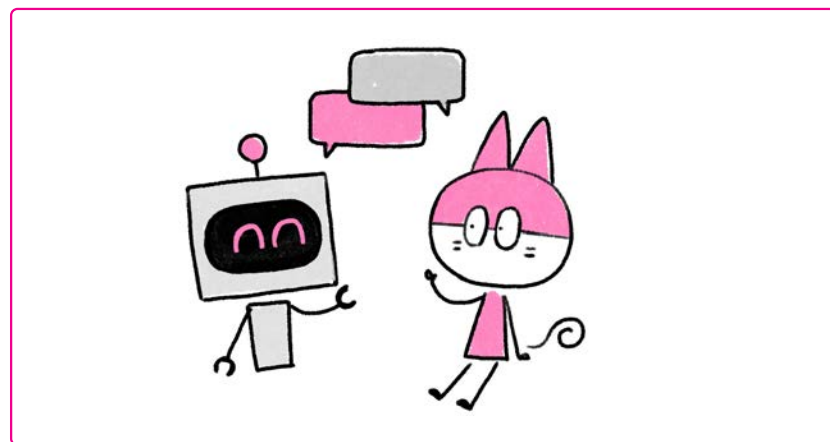


図1.6 チャットボット

チャットボットは、人間の入力した文を読み取る文解析部と、解析結果から応答文を作成する文生成部から構成されます。生成AIを用いると、解析と生成を一連の流れとして構成することが可能です。

※類似文書検索

類似文書検索は、利用者から入力された文書を手掛かりに、その文書と類似する文書を検索する技術です。2つの文書の類似性を数値的に計算することで、類似した文書を探し出します。

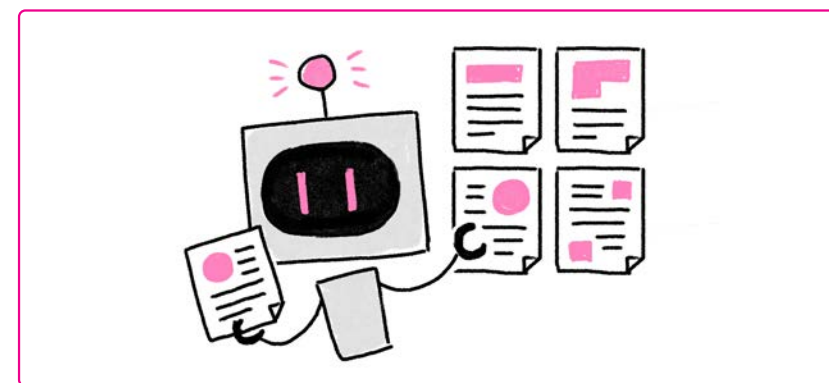


図1.7 類似文書検索

※文章の要約と生成

自然言語処理技術を応用すると、文章の要約や生成、つまり、利用者の指示に従って、文章を要約したり、新たな文章を生成したりすることが可能です。

文章の生成は、生成AIの得意分野です。また生成AIを用いると、文章の要約も行うことが可能です。

※自然言語処理の手続き

自然言語処理はさまざまな処理過程を経て実現されます。図1.9に、自然言語処理の一般的な処理過程を示します。

図1.9で、人間から与えられる入力には、音声データと文字データの2種類があります。

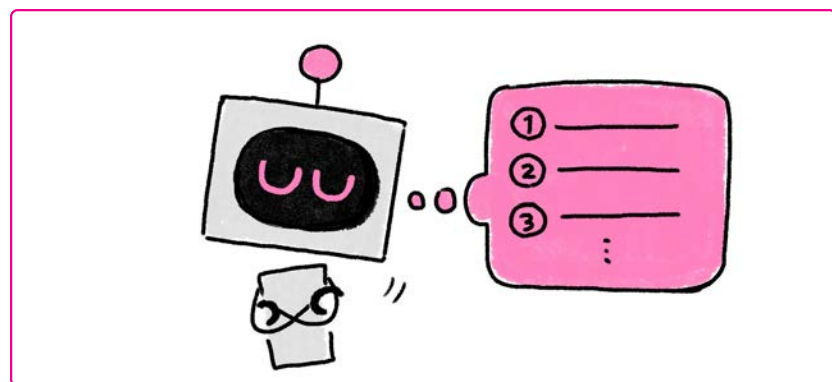


図1.8 文章の要約と生成

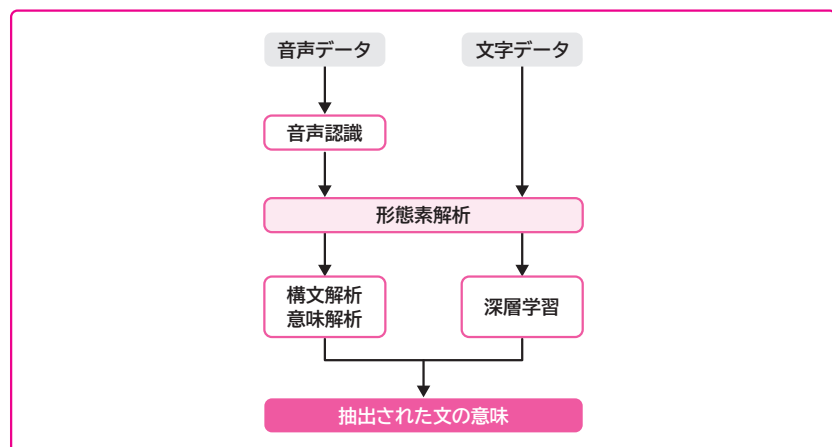


図1.9 自然言語処理の一般的な処理過程

音声データはマイク等を通してコンピュータに入力される時系列信号、すなわち時間の経過とともに変化するアナログ信号です。音声データはデジタルデータに変換することでコンピュータに取り込みます。この過程を**A/D変換**（アナログ/デジタル変換）と呼びます。取り込んだデジタルデータをプログラムで解析することで、文字データに変換します。

文字データは、はじめに適当な長さの文字の集まりに切り分けられます。この操作を一般に**形態素解析**と呼びます。ここで形態素とは、文を構成する部品

のことであり、いわゆる単語は形態素の一種です。形態素解析では、単語レベルでの区切りを与えるほか、さらに細かい**サブワード**のレベルで区切りを与える場合もあります。

伝統的な自然言語処理では、形態素解析によって形態素に分割された文を、さらに**構文解析**、**意味解析**といった解析段階を経て、意味を取り出します。現代的な深層学習ベースの自然言語処理では、形態素の並びを深層学習の手法で直接学習して、その学習結果を元の文の意味であるとして出力します。

※ 深層学習と自然言語処理

本書の後半で示すように、深層学習では、処理対象のデータは必ず数値でなければなりません。このため、自然言語処理に深層学習を適用するためには、文字による文の表現を、数値による表現に変換しなければなりません。このため深層学習を用いた自然言語処理においては、形態素解析によって得られた単語やサブワードの並びを、複数の数値による表現に変換してから、深層学習の手法を適用します。このような表現方法を単語の**分散表現**と呼びます。分散表現の計算には、例えばWord2Vecという方法があります。

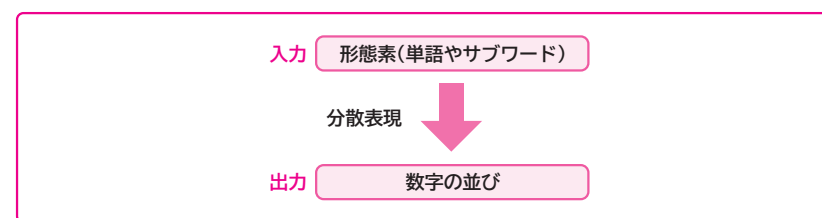


図1.10 分散表現 文字による表現から数値による表現への変化

② プログラミング例題 チャットボットのプログラム

自然言語処理の例題として、Pythonを使ってチャットボットを作成してみましょう。実行例1のようなチャットのできるチャットを考えます。

🔍 実行例1 チャットボットchat.pyの動作例(下線部は人間の入力)

さくら:こんにちは!
あなた:こんにちは、さくらさん。
さくら:わたしはさくらです。あなたのお名前は?
あなた:わたしは佐藤です。
さくら:そうですね〜
あなた:今日の天気はどうですか?
さくら:今日はこちらは晴れていますよ。
あなた:こちらは曇りです。
さくら:そうですね〜
あなた:明日雨が降るかどうかわかりますか?
さくら:うーん、良く分かりません・・・
あなた:予報では晴れみたいですけど。
さくら:そうですね〜
あなた:そろそろ時間です。さようなら。
さくら:それではお話を終わしましょう

実行例1で、「さくら:」から始まる文はチャットボットの出力です。また、下線部は人間がキーボードから打ち込んだ入力文です。

実行例1のような動作を行うチャットボットを作る簡単な方法は、チャットボットが人間の入力を見張っていて、あらかじめ登録しておいた単語、すなわちキーワードが見つかったら、用意しておいたそれに対する応答文を出力することです。これは、ワイゼンバウムのELIZAでも取り入れられている手法です。

表1.1に、実行例1の応答に必要な条件を示します。表1で最初の行を見ると、例えばチャットボットの名前である「さくら」という単語が入力されたら、「さくら:わたしはさくらです。あなたのお名前は?」と出力することが分かります。

表1.1に従って作成したチャットボットのプログラムであるchat.pyを、プログラムリスト1に示します。

表1.1 人間からの入力に含まれる単語と、それに対する応答文の組み合わせ

人間からの入力に含まれる単語(キーワード)	応答文
「さくら」	さくら:わたしはさくらです。あなたのお名前は?
「天気」	さくら:今日はこちらは晴れていますよ。
「?」	さくら:うーん、良く分かりません・・・
「バイバイ」又は「さようなら」	(対話を終了する)
上記以外	さくら:そうですね〜

📄 プログラムリスト1 chat.pyプログラム

```
01  """
02  チャットボット chat.py
03  使い方 >python chat.py
04  """
05
06  # メイン実行部
07  # 入力と応答
08  print("さくら:こんにちは!")
09  while True :
10      inputline = input("あなた:")
11      if "さくら" in inputline :
12          print("さくら:わたしはさくらです。あなたのお名前は?")
13      elif "天気" in inputline :
14          print("さくら:今日はこちらは晴れていますよ。")
15      elif "?" in inputline :
16          print("さくら:うーん、良く分かりません・・・")
17      elif ("バイバイ" in inputline or "さようなら" in
18            inputline) :
19          break # 対話を終わります
20      else : # 対話を続けます
21          print("さくら:そうですね〜")
```

```

22 print("さくら:それではお話を終わしましょう")
23
24 # chat.pyの終わり

```

chat.pyプログラムでは、人間からの入力を受け取り、その中に含まれる単語を調べて、表1に示した応答文を出力します。このために、プログラムの9行目から20行目のwhile文によって、入力と条件判定を繰り返しています。その後、22行目のprint()関数の呼び出しによって終了のメッセージを出力してプログラムを終了します。

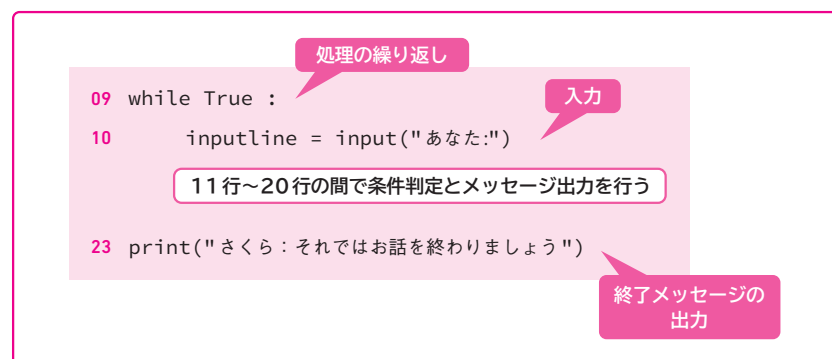


図1.11 chat.pyプログラムの概要

11行から20行の間の条件判定は、if ~ elif ~ elseの連鎖によって表1の動作を実現します。例えば表1で、キーワード「さくら」を見つけて応答文を出力する部分は、プログラムでは次のように記述しています。

```

11 if "さくら" in inputline :
12     print("さくら:わたしはさくらです。あなたのお名前は?")

```

11行目のinputline変数には、人間から入力された入力文が格納されています。11行目では、in演算子を用いて、入力文の中に「さくら」というキーワードが含まれているかどうかを調べています。

13行目と15行目のelifは、キーワード「天気」とキーワード「?」に対応する処理を記述しています。また17行目のelifは、「バイバイ」又は「さようなら」と入力されたら、breakを使ってwhileの繰り返しが終了して、22行目の終了メッセージ出力に進む働きがあります。これら以外の入力の場合には、19行目のelseによって定型的な応答文「そうなんですね〜」が出力されます。

1.2

Google検索はAIのかたまり — AIを意識しなくてもAI

★Web検索のしくみ

Web検索は、膨大な情報が存在するWWWの世界の中から、必要な情報を瞬時に探し出してくれるシステムです。Web検索のサービスは、GoogleやYahoo!、Bingなどが提供しています。

Web検索システムの概要を図1.12に示します。Web検索システムは、利用者からの検索要求に対して、実際のWebサイトを検索しているわけではありません。Web検索システムは、あらかじめ世界中のWebサイトを訪問して集めておいたデータを対象として、検索エンジンとよばれる検索ソフトウェアを用いて対象データを探しています。

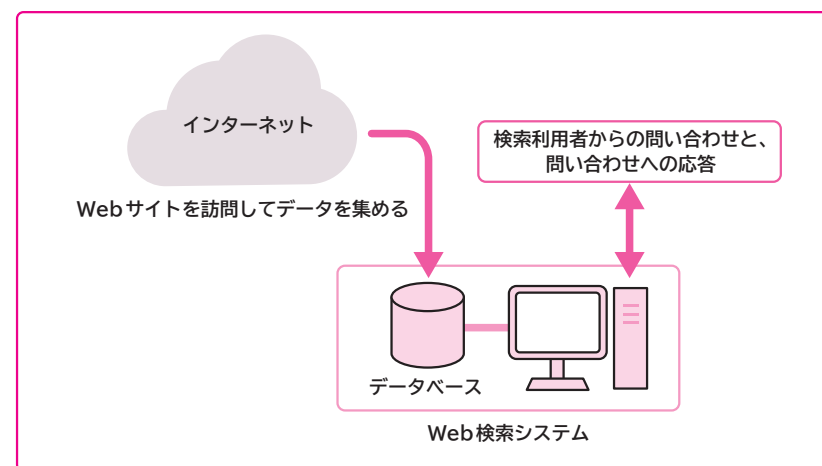


図1.12 Web検索システムの概要

世界中のWebサイトを訪問してデータを集めるためには、**クローラ**あるいはロボットと呼ばれるプログラムを利用します。クローラは、自動的にWebサイトを訪問して、必要なデータを抽出し、Web検索システムのデータベースにデータを登録します。

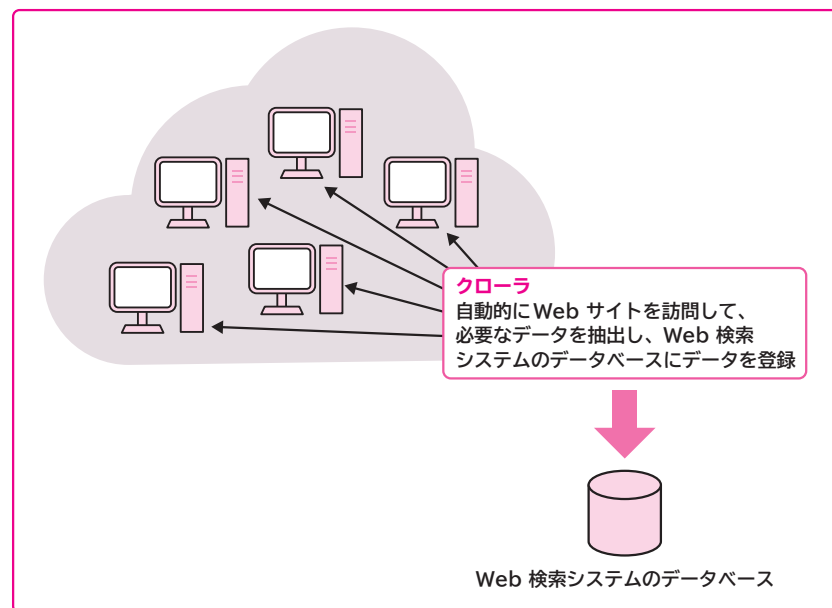


図1.13 クローラの動作

Webサイトの内容は繰り返し更新されるのが普通ですから、クローラも繰り返してWebサイトの情報を集めて回らなければなりません。また、訪問の対象は世界中の膨大な数のWebサイトです。クローラは効率よく膨大なデータを処理するために、さまざまな人工知能技術が利用しています。検索エンジンも、膨大な対象データから一瞬でデータを探し出すために、人工知能研究で培われたさまざまなソフトウェア技術を利用しています。

★エージェント技術

クローラは、WWWのリンクをたどることで世界中のWebサイトを自律的

に探訪します。クローラのように自律的に動作するソフトウェアを**ソフトウェアエージェント**と呼びます。ソフトウェアエージェントは、機械の体を持ったエージェントであるロボットに対して、実際の体は持たずにプログラムだけで構成したエージェントです。

ソフトウェアエージェントには、クローラのようにネットワーク上で稼働するプログラムであるネットワークエージェントの他、メールに対して代理で返答するメールエージェントや、決められた条件でデータバックアップを行うバックアップエージェントなどがあります。

★探索の技術

クローラがWebサイトを訪問する際、クローラはWebのリンクを利用して、次から次へとWebサイトをアクセスします。この時、すべてのWebサイトをもれなく訪れようとすると、一定のアルゴリズムに従って訪問を続ける必要があります。

例えば図1.14で、WebサイトSから訪問を始めて、矢印で示したリンクをたどることですべてのサイトを効率よく訪問するには、どのような順番で訪問を進めれば良いでしょうか。

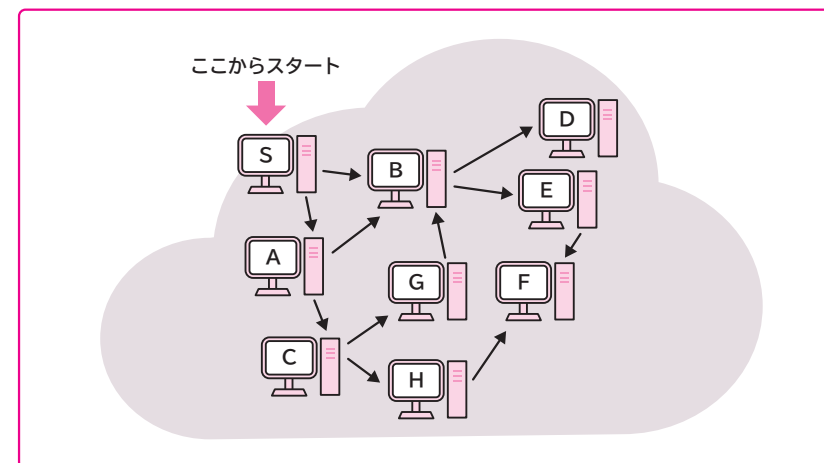


図1.14 Webサイトの探索

図1.14のような処理を、探索と呼びます。人工知能研究の歴史においては、探索は初期に研究された研究課題の一つです。探索は、例えば地図上でのルート探索や、ロボットの制御手順の探索、あるいはボードゲームの手順の探索など、さまざまな問題に応用可能です。以下では、いくつかの探索手法について紹介します。

※横型散策

図1.14の例で、WebサイトSから訪問を始めて矢印で示したリンクをたどり、探索先のWebサイトを順に並べて線で接続すると、図1.15のような図が出来上がります。このような、探索の過程を示した図を、探索木と呼びます。

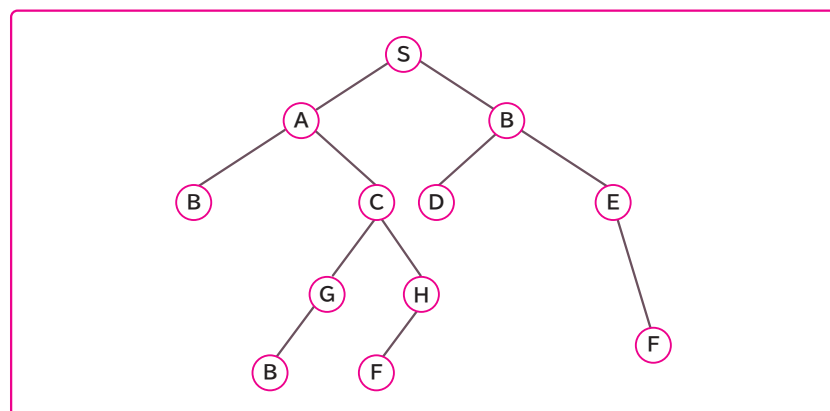


図1.15 探索木の例

探索木は木(tree)と呼ばれるデータ構造の一種です。図15で、アルファベットで示した探索対象を節点(node)と呼び、特に一番上に書かれた探索を開始する節点を根(root)と呼びます。図1.15の探索木では、根は節点Sです。根とは逆に、探索木の一番下に配置される節点を葉(leaf)と呼びます。さらに、図1.15の例には出現しませんが、一般の探索木では、探索のゴールとなる目標節点が存在します。例えば地図を使って出発点から目的地(ゴール地点)までの道順を探索するルート探索の場合には、目的節点は目的地(ゴール地点)に対応します。

探索木を作成して探索を実行する過程では、根から初めて次の節点を探して、木を成長させます。図1.14の例で言えば、探索木を作成する手続きの初期状態においては、根である節点Sだけが存在する木となります(図1.16)。

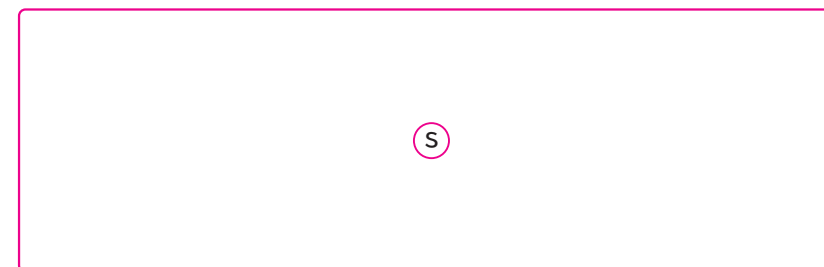


図1.16 探索木の初期状態(根である節点Sだけが存在)

節点Sから次の節点を求めることを、展開と言います。探索は、展開を繰り返して探索木を成長させる過程です。

節点Sを展開するには、図1.14に戻って、WebサイトSのリンク先のWebサイトを探します。Sのリンク先にはWebサイトAとWebサイトBがあります。そこで、節点Sを展開して節点Aと節点Bが見つかったことを、図1.17のように記述します。

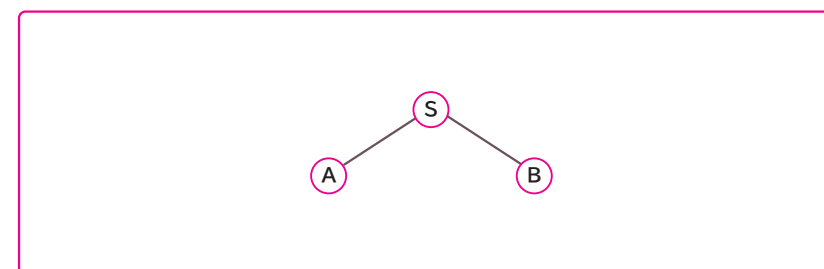


図1.17 節点Sを展開(節点Aと節点Bが見つかる)

次に、節点Aを展開して節点Bと節点Cを得て、同様に、節点Bを展開して節点Dと節点Eを得ます(図1.18)。

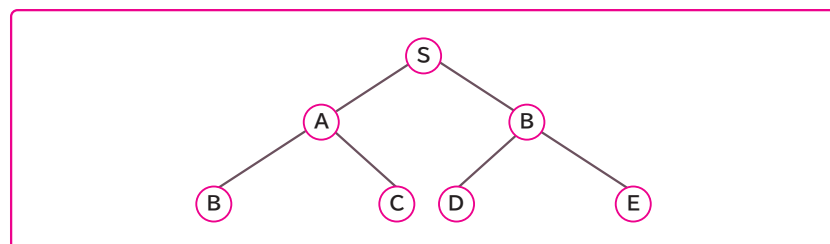


図1.18 節点Aと節点Bを展開

このように、同じ深さ(木の縦方向の位置)となる節点を優先して、探索木の横方向(木の幅の方向)に対して展開する探索方法を、**横型探索**と呼びます。横型探索は、**幅優先探索**とも言います。横型探索は、探索対象をもれなく探索することのできる探索アルゴリズムです。

ちなみに、図1.18からさらに探索を続ける場合を考えます。図1.18で、左端の節点Bは、既に節点Sの展開時に展開済みです。このため、繰り返し(ループ)を避けるために、ここでは節点Bは展開しません。

また節点Dは、図1.14のWebサイトの接続関係図を見ると、行き止まりで、次のリンク先はありません。このため、節点Dは展開できません。

残りの節点Cと節点Eは展開可能です。そこで結果として、節点Cと節点Eを展開して、図1.19のような探索木を得ます。

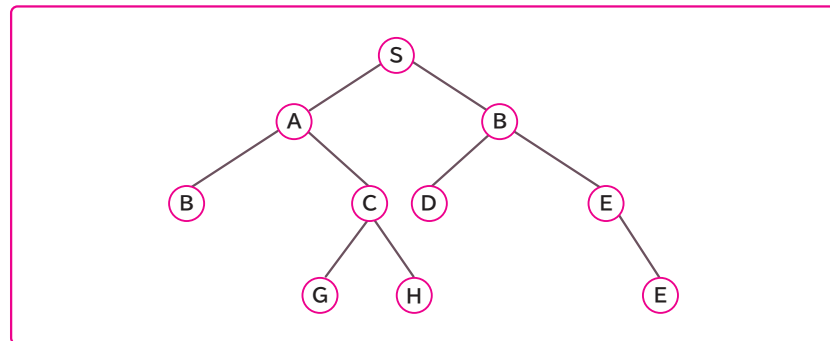


図1.19 節点Cと節点Eを展開(節点Bと節点Dは展開不要)

さらに展開可能な節点Hについて展開を施すと、図15に示した探索木が完成します。

縦型探索

図1.17の状態、横型探索では、同じ深さとなる節点を優先して展開するため、節点Aと節点Bを並行して展開します。しかし別の方法として、探索木のより深い方向に先に展開を進める方法で探索を行う方法も考えられます。この方法を**縦型探索**といいます。縦型探索は、**深さ優先探索**とも呼びます。

図1.17の例で縦型探索を行うと、節点Aの展開が終わった後、節点Bの展開はひとまず置いておいて、先に節点Aによって求めた節点を展開します(図1.20)。

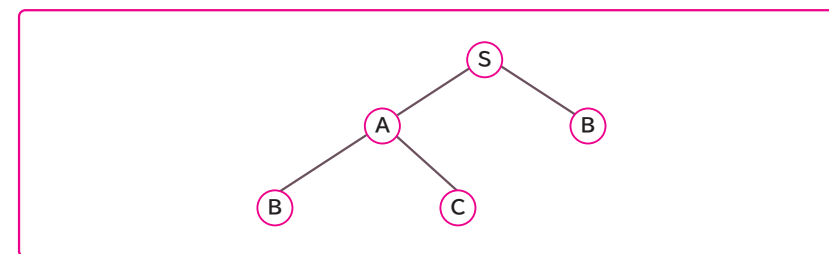


図1.20 縦型探索の例(1) 節点Aの展開

さらに、深さ方向の探索を優先して、展開可能な節点Cを続けて展開します(図1.21)。

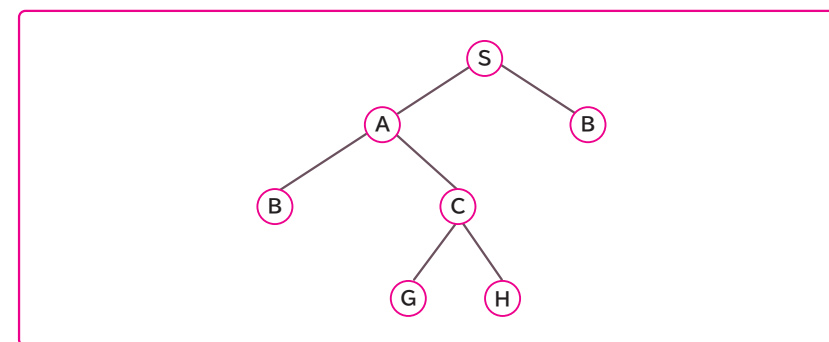


図1.21 縦型探索の例(2) 節点Cの展開

引き続き、深さ方向を優先して節点Gを展開します(図1.22)。

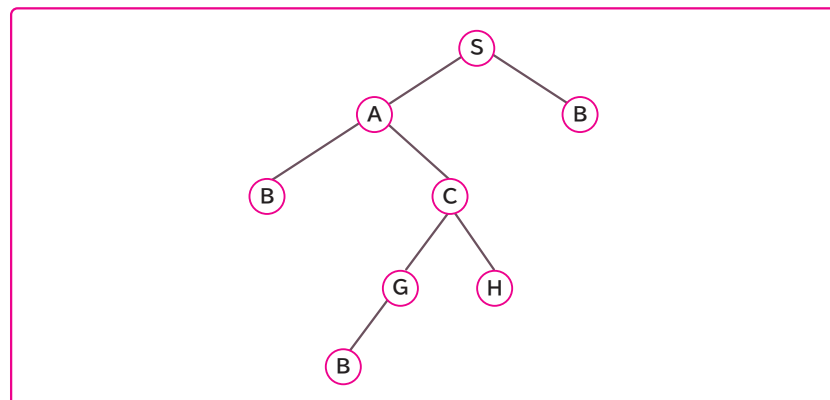


図1.22 縦型探索の例(3) 節点Gの展開

図1.22で、節点Bは既に展開によって出現済みですから、これ以上展開できません。このため、いったん棚上げてある節点Hの展開を行います。

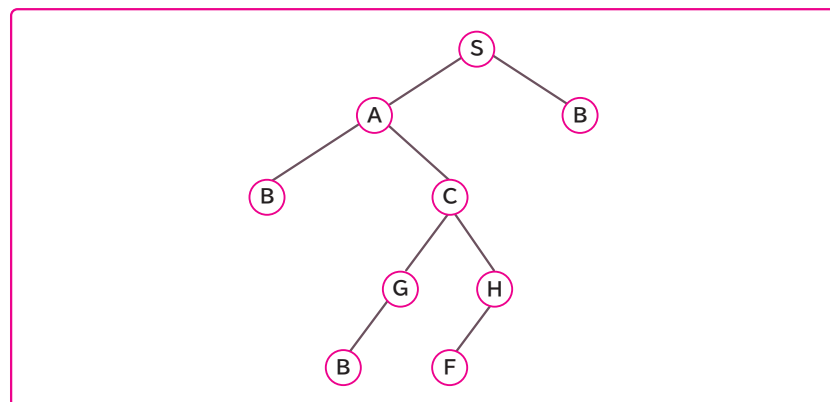


図1.23 縦型探索の例(4) 節点Hの展開

図1.23で展開によって得られた節点Fは行き止まりでこれ以上展開できません。そこで、棚上げてあった節点Bの展開に戻って探索を進めます(図1.24)。

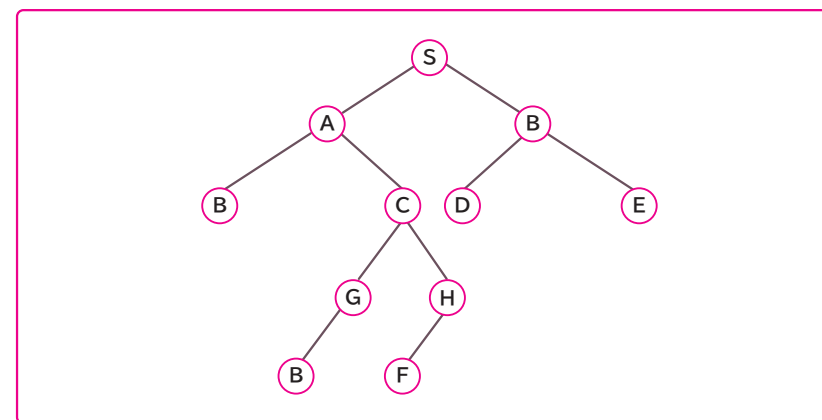


図1.24 縦型探索の例(5) 節点Bの展開

その後展開を進めることで、図1.15に示した探索木が完成します。

以上の例のように、横型探索と縦型探索は、どちらもしらみつぶしに展開を進めることで、同じ探索木を完成させることが可能です。この意味では、どちらのアルゴリズムも同じ処理を行うことになります。

ただし、ルート探索のような目的節点のある探索においては、横型探索では深さの浅い節点を優先して見つけることができます。ルート探索において、深さが浅い節点を見つけられるということは、目的地までの近道を見つけられることを意味します。

縦型探索では、より深さの浅い部分に目的節点がある場合でも、深い部分にある節点を先に見つけてしまう場合があります。つまり、遠回りの道順を見つけしてしまうかもしれません。ただし、探索木の枝分かれの多い場合には、横型探索と比較して縦型探索は目的節点をより素早く見つけられる可能性があります。

★その他の探索技術

横型探索や縦型探索は、網羅的に探索を進めるための探索アルゴリズムです。横型探索や縦型探索では、どのような問題について探索を行っているのかという、問題についての知識を用いません。このため、場合によっては非効率的な探索を行ってしまいます。

例えば、出発点から目的地までの道順を探索するルート探索では、一般に、目的地になるべく近づくように経路を選択すべきです。もちろん遠回りしないと目的地にたどり着けないこともあります。大抵の場合は、なるべく目的地に近づく方向に探索を進める方が効率的です。このように、どのような問題を解いているのかについての知識を用いて探索を効率化する手法を**最良優先探索**と呼びます(図1.25)。

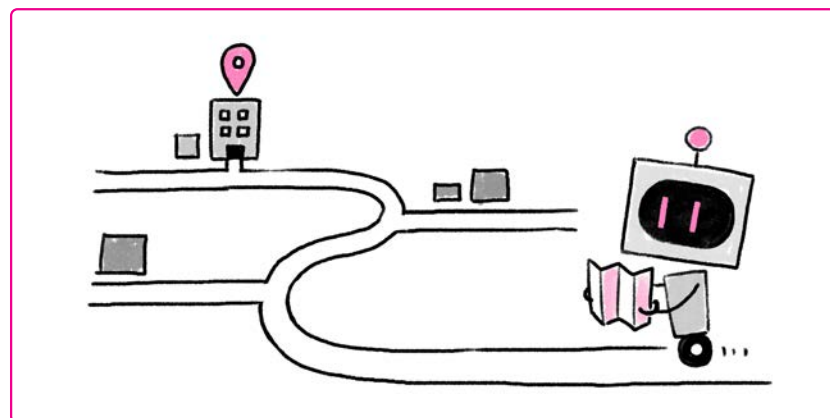


図1.25 最良優先探索 「良さそう」な方から先に探索する

探索問題では、見つかった答えの効率が問題になる場合もあります。ルート探索の例で言うと、出発点から目的地までのルートは、単に見つかれば良いのではなく、普通は最も距離の短いルートを見つけるべきです。このように、ある評価基準に従って最も良い答えを見つける探索手法を**最適経路探索**と呼びます。最適経路探索は、**分枝限定法**又は**分岐限定法**とも呼びます(図1.26)。

さらに、最良優先探索と最適経路探索は組み合わせることもできます。組み合わせた探索アルゴリズムを、A-アルゴリズムと呼びます。A-アルゴリズムのうちで一定の条件をみたしたものは、必ず最適解を見つけられることが分かっています。この場合のA-アルゴリズムを、特に、A*-アルゴリズムと呼びます。ここで「A*」は「エースター」と読みます。A-アルゴリズムやA*-アルゴリズムは、ルート探索やゲーム、配送ルート問題など、さまざまな問題の解決に利用されます。

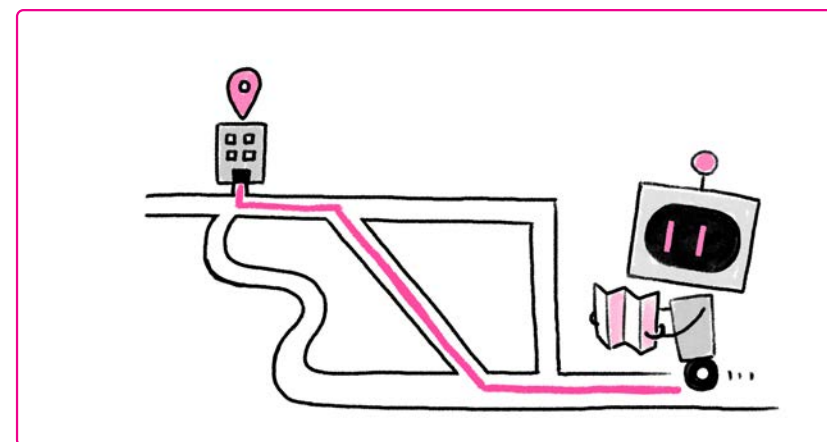


図1.26 最適経路探索 評価の良い経路を探索する

1.3 Amazonのお買い物でつい買いすぎるのはなぜ？ — AIとヒューマンとの駆け引き

★ネットショップのしくみ

インターネットのWebサイトでお店を開いているネットショップは、いつでもどこでも買い物ができ、とても便利なお店です。ネットショップの代表例には、Amazon、Yahoo!ショッピング、あるいは楽天市場などがあります。これらのお店は、WWWのサーバ上にコンピュータプログラムとして構築されています。

ネットショップ側のコンピュータでは、商品の管理や売上の決済処理を行います。それだけでなく、同時に、顧客がネットショップでどのような購買行動を行ったのかの記録をとります。つまり、買い物の履歴をデータとして蓄積します。

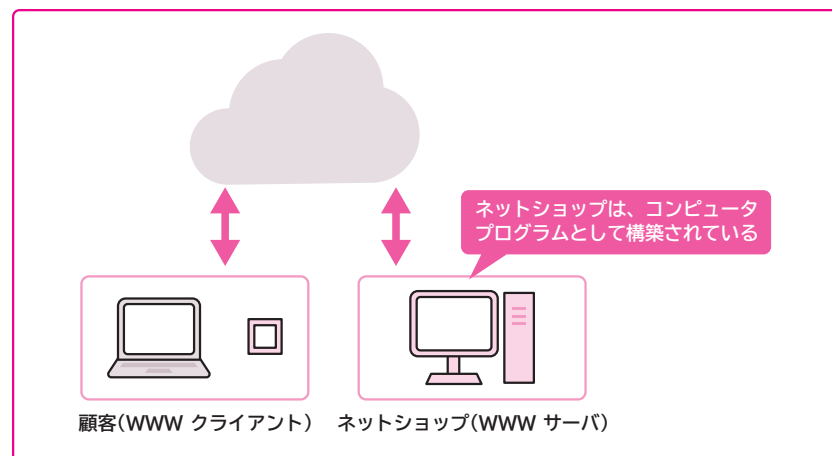


図1.27 ネットショップ

★買い物履歴の学習

こうして蓄積した買い物履歴のデータは、おすすめ商品の提案に使うことができます。

例えば、ある顧客Aさんが、スマホ用の充電器(USB給電の電源)を買ったとしましょう。この時、Aさんは充電器と同時に、USBケーブルも購入したとします。

このことから、ネットショップのシステムは、充電器とUSBケーブルが同時に売れる場合があることを学習することができます。そこで、他の顧客Bさんが充電器を購入しようとしていたら、ネットショップのシステムは「USBケーブルも一緒にいかがですか?」と勧めることができます(図1.28)。

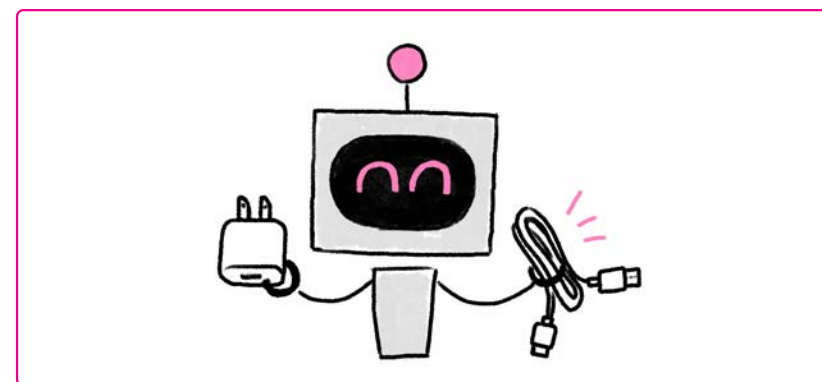


図1.28 ネットショップのお勧め表示

こうした仕組みは、過去の経験を知識として一般化することで実現することができます。つまり、ネットショップのシステムが個々の買い物履歴を蓄積して、その中から充電器とUSBケーブルが一緒に購入されやすいという知識を得るのです。このような挙動は、一般に機械学習と呼ばれます。機械学習は、機械、すなわちコンピュータプログラムが、経験から知識を得る過程です。機械学習は、人工知能の世界では単に**学習**と呼ぶ場合も多いです。

★学習とは?

ここで、本来の意味での学習について考えてみましょう。本来、学習という言葉は、人間を対象として用いるのが普通であり、まれには動物に対しても使われます。ここでは、人間の学習について考えます。

人間が行う学習といっても、さまざまな種類のものがあります。例えば、学校で習う数学や英語の勉強は学習ですし、スポーツや音楽の練習も学習です。また、自転車の乗り方の習得や、道具の使い方を練習することも学習です。

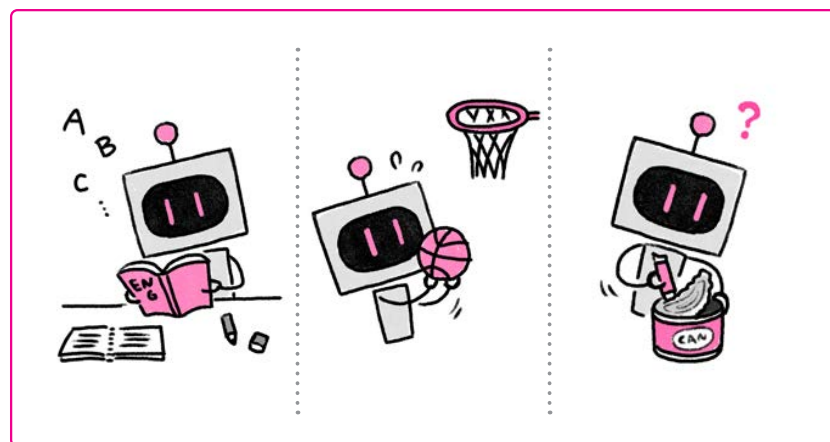


図1.29 さまざまな学習

これらの例では、いずれも、学習者が外界とやりとりをして、その結果から学習者自身の状態をより良い方向へ変化させています。例えば漢字の書き取り練習という学習を行えば、漢字書き取り帳や鉛筆と人間が相互作用することで、人間が漢字を覚えます。人間がピアノとやりとりすることで、人間はピアノの演奏を身に着けます。こうした変化の過程を学習と呼びます。

コンピュータプログラムが学習を行う機械学習でも同様に、プログラムが外界と相互作用することでプログラムの内部状態が変化し、その結果、プログラムの挙動が変わってより良い動作をするようになります。例えばネットショップのシステムが顧客とやり取りすることで、顧客の購買行動について知識を身に着けることができます。この場合、ネットショップのプログラムが学習を行ったことになります(図1.30)。

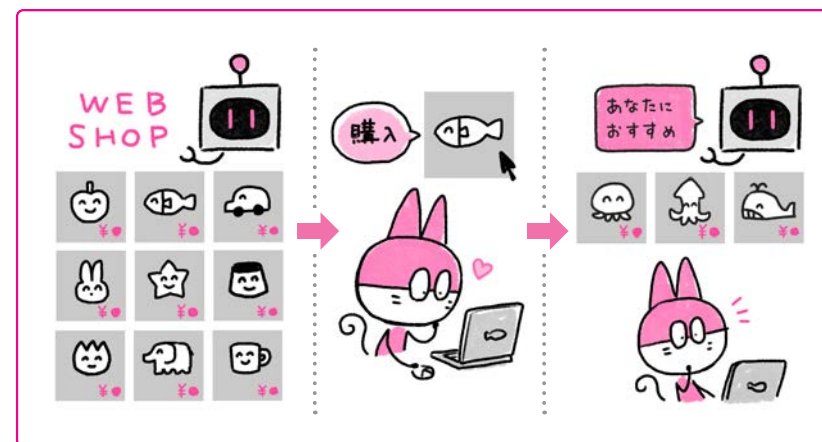


図1.30 機械学習

❓プログラミング例題 人間の入力過程を学習するプログラム

ここでは簡単な機械学習の例を、Pythonのプログラムを使って示したいと思います。

このプログラムでは、人間がプログラムに対して、○か×かどちらかの記号を繰り返して入力します。プログラムは過去の人間の入力系列を学習することで、次に人間が入力する記号が○か×かのどちらかを予測します。

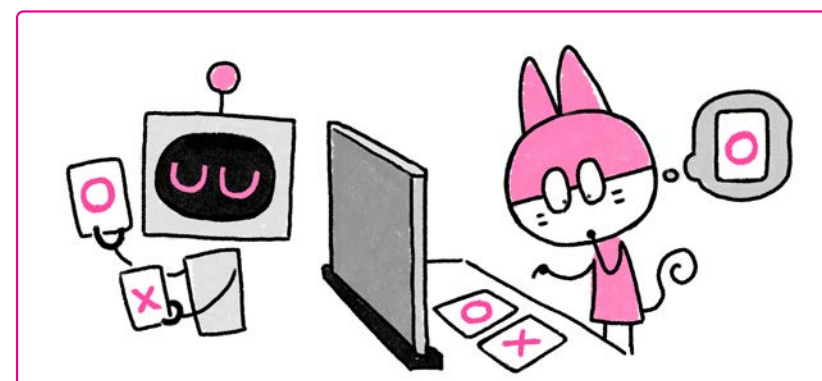


図1.31 簡単な機械学習の例

学習の方法として、プログラムは人間の入力した記号について、○が何回入力されたか、また×が何回入力されたかを記録しておきます。次の入力の予測では、これまで入力された回数が少ない方の記号を選びます。これは、「人間は○と×をバランスよく入力する」という仮説に基づいた選択です。

例えば過去5回の入力で4回○が入力され、×は1回だけであれば、次の人間の入力はおそらく×だろう、と予測します(図1.32)。

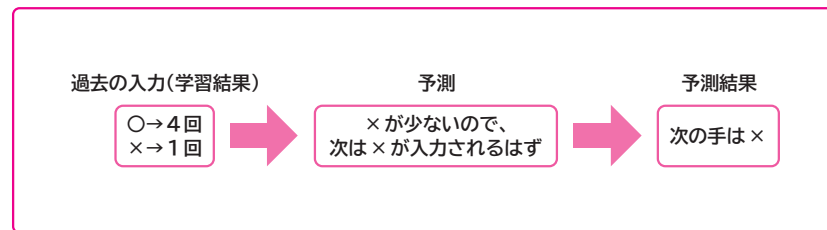


図1.32 ○と×の学習と、次の手の予測

以上の設定で、○×の学習と予測を行うプログラムであるmarubatu.pyを構成します。プログラムリスト2にmarubatu.py プログラムを示します。

プログラムリスト2 marubatu.pyプログラム

```

01  """
02  ○×の学習と予測を行うプログラム marubatu.py
03  使い方 >python marubatu.py
04  """
05  # モジュールのインポート
06  import time # sleep()の利用に必要
07
08  # グローバル変数
09  W = 2 # 表示の待ち合わせ時間(秒)
10
11  # メイン実行部
12  # 変数の初期化

```

```

13  memory = [0, 0] # 過去の手の記憶
14
15  # 入力と応答
16  print("さくら:あなたの入力が○か×かを当ててみます...")
17  while True :
18      time.sleep(W) # 待ち合わせ
19      print("さくら:○か×を思い浮かべてください...")
20      if memory[0] > memory[1] : # 過去の手で○が多い
21          p = '×' # きっと、今度は×
22      else : # ○は多くない
23          p = '○' # きっと、今度は○
24      time.sleep(W) # 待ち合わせ
25      print("さくら:あなたの手を占いました...")
26      time.sleep(W) # 待ち合わせ
27      print("さくら:それでは、思い浮かべた手が○なら0を、×なら1を
    入力してください...")
28      te = int(input())
29      if te == 0 or te == 1:
30          print("さくら:私の占いでは、あなたの手は", p , "でした...")
31          time.sleep(W) # 待ち合わせ
32          memory[te] += 1 # 過去の入力の学習
33          print("さくら:もう一度占いましょう...")
34      else :
35          break
36
37  print("さくら:それでは終わりました...")
38
39  # marubatu.pyの終わり

```

実行例2に、marubatu.pyプログラムの実行例を示します。marubatu.pyプログラムでは、○を入力する場合には数字の0を、×を入力する場合には数字の

1を入力します。実行例では、人間は最初に思い浮かべた手が○だったので、0を入力しています。これに対してmarubatu.pyプログラムは、「占い」によって○を予測した、と出力しています。もちろん実際には占いなどではなく、図1.32に示したアルゴリズムに従った予測結果に過ぎません。1回目はプログラムの予測と人間の入力一致しており、予測が正解であったことが分かります。

2回目の入力では、人間は数字の1を入力することで、思い浮かべた手が×であることを示しています。これに対してmarubatu.pyプログラムは、予測は×であり、今回も予測が当たっています。

3回目の入力では数字の9を入力しています。これに対してmarubatu.pyプログラムでは、0または1以外の数値が入力されるとプログラムを終了します。

実行例2 marubatu.pyプログラムの実行例

さくら:あなたの入力が○か×かを当ててみます・・・
 さくら:○か×を思い浮かべてください・・・
 さくら:あなたの手を占いました・・・
 さくら:それでは、思い浮かべた手が○なら0を、×なら1を入力してください・・・
 0 ← 思い浮かべた手は○
 さくら:私の占いでは、あなたの手は ○ でした・・・
 さくら:もう一度占いましょう・・・ ← 予測も○(正解)
 さくら:○か×を思い浮かべてください・・・
 さくら:あなたの手を占いました・・・
 さくら:それでは、思い浮かべた手が○なら0を、×なら1を入力してください・・・
 1 ← 思い浮かべた手は×
 さくら:私の占いでは、あなたの手は × でした・・・
 さくら:もう一度占いましょう・・・ ← 予測も×(正解)
 さくら:○か×を思い浮かべてください・・・
 さくら:あなたの手を占いました・・・
 さくら:それでは、思い浮かべた手が○なら0を、×なら1を入力してください・・・
 9 ← 0または1以外の数字を入力すると終了
 さくら:それでは終わりました・・・

marubatu.pyプログラムの仕組みを簡単に説明します。

プログラムの6行目では、timeモジュールをインポートしています。

```
05 # モジュールのインポート
06 import time # sleep()の利用に必要
```

実行例2ではわかりませんが、marubatu.pyプログラムではプログラムの出力は1行ずつゆっくりと画面上に出てきます。占いの雰囲気を出すための演出ですが、このような出力の待ち合わせを行うためにtimeモジュールをインポートして、プログラムを一時停止させる働きのあるsleep()関数を利用しています。

marubatu.pyプログラムでは人間の手の入力を学習します。このためには、過去の入力値に関する記憶が必要です。そこでプログラムの13行目では、memory[]という名称のリストを準備しています。

```
12 # 変数の初期化
13 memory = [0, 0] # 過去の手の記憶
```

memory[]には、○と×が過去何回入力されたかを記録します。この時、memory[0]には○が入力された回数を記録し、memory[1]には×が入力された回数を記録します。初期値がいずれも0とします。

17行から35行の間のwhileによる繰り返しは、marubatu.pyプログラムの処理の中心です。

まず20行から23行のif文では、次の手の予測を行っています。予測の方法は、先に述べたように、「これまで入力された回数が少ない方が次の手となる」というものです。20行目のif文は、過去の○の入力回数を格納したmemory[0]が、×の回数を格納したmemory[1]の値より大きいかどうかを調べています。この結果により、過去の入力回数が少ない方が次回の手であると予測します。

```
20 if memory[0] > memory[1] : # 過去の手で○が多い
21     p = '×' # きっと、今度は×
22 else : # ○は多くない
```

```
23      p = '○' # きっと、今度は○
```

その後人間からの入力を受け取り、入力後にプログラムの予測手を示します。次の32行で、人間の手の学習を行います。ここでの学習とは、過去の手を記憶するリストmemory[]の値の更新作業を意味します。

```
32      memory[te] += 1 # 過去の入力の学習
```

marubatu.pyプログラムの学習はごく単純なものです。例えばかな漢字変換の変換候補順の学習と同様の学習であると言えるでしょう。かな漢字変換のシステムは、変換候補を提示する際に過去の履歴を参照して、過去良く使われた変換結果を変換候補の上位に表示します。これは、marubatu.pyプログラムの学習と似たような仕組みです。

もちろん実際には、機械学習の手法として、もっと複雑な方法によるさまざまな方法が提案されています。より複雑な機械学習については、2章以降で見ていくことにします。

1.4

CopilotとGeminiで時短生活——生成AI —— どれが私に合っているのか分からない

★生成AIのできること

0章で紹介したように、**生成AI**は、人間の指示に従って、文章や画像、動画などを生成することができます。例えば、Google社のGeminiという生成AIに次のように問いかけてみます。

「生成AIとは何かを、100文字程度で説明してください。」

すると、Geminiは次のように答えてくれました。

「生成AIは、大量のデータから学習し、テキスト、画像、音声、動画といった新しいコンテンツを**自律的に創造**できる人工知能です。ユーザーの指示に基

づき、オリジナルの情報を生み出します。」

この例のように、生成AIは、人間の指示に従ってあたかも人間が書いたような文章を生成します。生成AIは文章だけでなく、静止画像や動画なども生成することができます。

なお、Geminiはいつでも必ずこのように答えるわけではありません。生成AIの処理では、同じ指示を与えても必ず同じ答えが出るのではなく、答えにランダムさが含まれています。

★生成AIのしくみ

では、生成AIはどのようにして文章などを生成するのでしょうか。実は生成AIは、文の意味を考えて文章を生成しているわけではありません。生成AIは、過去に学習したデータを探索して、関連する単語を順に繋いでいくことで文を作り上げます。この時、単語や文の意味を考えているわけではなく、単に、いかにもありそうな順番で単語を並べているにすぎません。

生成AIは、過去に人間の書いた大量の文章を読み込んで、単語の並べ方を学習します。この学習結果は、**大規模言語モデル**(LLM)と呼ばれます。

大規模言語モデルがどのくらいの規模かという、例えば初期の大規模言語モデルの代表例であるGPT-3という言語モデルでは、およそ4000億単語の分量の文を学習しています。これらの文は、Webサイトや書籍、それにウィキペディアなどの電子データから採録されています。

GTP-3の後継であるGTP-3.5あるいはGTP-4では、公表はされていませんが、その10倍以上のデータを使って学習したと言われています。これだけの量の文章を元にしていて、大規模言語モデルは大変な量の知識を蓄えていることになります(図1.33)。

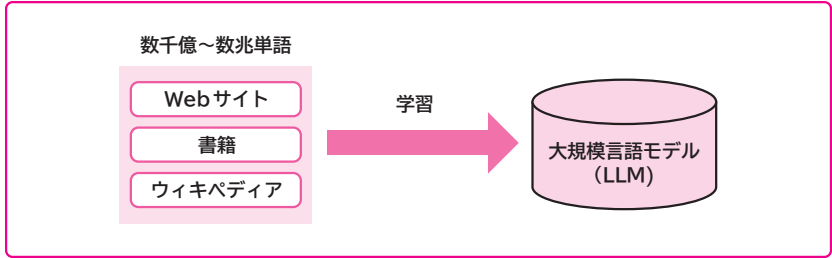


図1.33 大規模言語モデル(LLM)

生成AIは、人間から与えられた指示を解析して単語を拾い上げ、それを手掛かりに大規模言語モデルを検索します。その結果得られた単語を並べることで、人間の指示に対する返答を作成します(図1.34)。

生成AIは単語を並べるだけと説明しましたが、数千億~数兆個もある単語の中から適切な単語を選んで順に繋げる作業は、いくらコンピュータが高速であっても容易ではありません。しかし実際には、生成AIは実用的な速度で文を素早く生成します。このために、生成AIが利用する大規模言語モデルには、特別な仕組みが必要となります。現在用いられている大規模言語モデルでは、ニューラルネット、特にディープニューラルネットの技術が利用されています。ニューラルネットについては、3章で詳しく説明します。

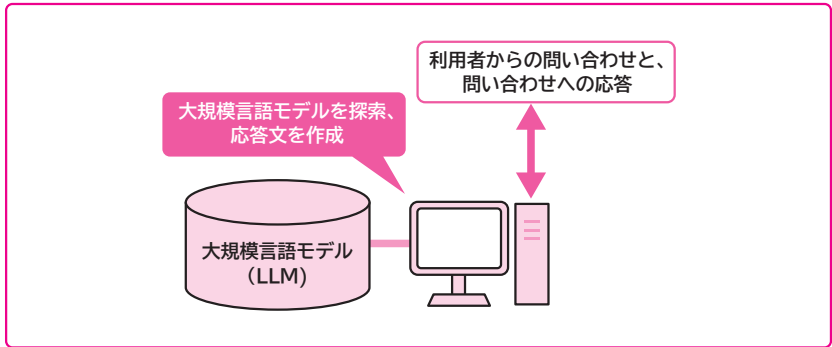


図1.34 生成AIのしくみ

❓ プログラミング例題 単語をつないで文を作る

生成AIが行う文生成の様子を、簡単なプログラムで真似てみましょう。ここでは、あらかじめ単語のつながり方の候補をいくつかプログラムに与えておくことで、単語をランダムにつないで文を生成するプログラムを考えます。これは、生成AIの動作とは異なり学習の過程を全く含んでいませんが、単語をつないで文を作る動作を理解する手助けになるでしょう。

ここでは、例えば以下のような文を作ることを考えます。

私は チャットボットの さくらです 。
私は 元気です 。
あなたは 誰です か。
あなたは 元気です か。

上記の文のうちの、空白で区切った、文字の連続する部分を単語と考えます。例えば最初の文では、「私は」「チャットボットの」「さくらです」「。」を単語と考えます。表2に、それぞれの文を構成する単語を示します。

表1.2 文を構成する単語

1 番 目	2 番 目	3 番 目	4 番 目
私は	チャットボットの	さくらです	。
私は		元気です	。
あなたは		誰です	か。
あなたは		元気です	か。

プログラムは、先頭から順番に、単語をランダムに選んで繋いでいくことで文を作ります。

例えば、先頭1番目の単語として「私は」を選び、2番目の部分の単語には「」(何ものなし)を選びます。3番目は「誰です」、4番目は「か。」を選びます。すると、次の文が出来上がります。

私は誰ですか。

ここでは、表1.2に示した単語を順に繋いで文を作っているのですが、表2は、ある種の言語モデルであると言えます。

以上のような動作をするプログラムであるplm.pyを、プログラムリスト3に示します。

プログラムリスト3 plm.pyプログラム

```
01  """
02  単語の連鎖で文を生成する plm.py
03  使い方 >python plm.py
04  """
05  # モジュールのインポート
06  import random # randint()の利用に必要
07
08  # メイン実行部
09  # 言語モデルの設定
10  plm = ["私は", "私は", "あなたは", "あなたは"], ["チャットボッ
11         トの", "", "", ""],
12         ["さくらです", "元気です", "誰です", "元気です"], [".",
13         "。", "か。", "か。"]]
14
15  # 文の生成
16  n = int(input("さくら:繰り返し回数を入力してください:"))
17  print("さくら:生成を開始します-----")
18  for _ in range(n) :
19      for i in range(4) :
20          print(plm[i][random.randint(0,3)], end="")
21          print()
22  print("さくら:以上で生成は終了です-----")
23
24  # plm.pyの終わり
```

実行例3に、plm.pプログラムの実行例を示します。実行例3では、20回繰り返して、ランダムに文を生成しています。

実行例3 plm.pyプログラムの実行例

```
> python plm.py
さくら:繰り返し回数を入力してください:20
さくら:生成を開始します-----
私は元気ですか。
私は元気ですか。
私はチャットボットのさくらですか。
私はさくらです。
あなたはチャットボットの誰です。
あなたは元気です。
私は元気です。
私はさくらです。
あなたは誰ですか。
あなたはチャットボットの誰ですか。
私はチャットボットの元気ですか。
あなたはチャットボットのさくらです。
私は元気です。
あなたは元気です。
私は元気ですか。
私はチャットボットの元気ですか。
あなたは誰ですか。
あなたは元気です。
あなたはさくらです。
あなたは元気です。
さくら:以上で生成は終了です-----
```

plm.pyプログラムの仕組みを説明します。プログラム冒頭の6行目では、randomモジュールをインポートしています。randomモジュールは、randomモ

ジュール内のrandint()関数を使って整数の乱数を生成するために利用します。

```
05 # モジュールのインポート
06 import random # randint()の利用に必要
```

プログラムの10行目では、文生成に利用するための一種の言語モデルを設定しています。ここでの言語モデルは表2に示した単語の並びです。これをPythonのリストとして表現して、変数plmに格納しています。

```
09 # 言語モデルの設定
10 plm = ["私は", "私は", "あなたは", "あなたは"], ["チャットボッ",
    トの", "", "", ""],
11      ["さくらです", "元気です", "誰です", "元気です"], [".",
    "。", "か。", "か。"]]
```

続いて13行では、文を何回繰り返して生成するかの指示をキーボード入力を受け取ります。

```
13 n = int(input("さくら:繰り返し回数を入力してください:"))
```

15行から18行のfor文によるn回の繰り返して、n行の文を生成します。各回の繰り返しのにおいて、一つの文は4つの単語から構成されます。それぞれの単語はrandint()関数を用いてランダムに選択します(16行から18行)。

```
16 for i in range(4) :
17     print(plm[i][random.randint(0,3)], end="")
18     print()
```

まとめ

- 人工知能技術は、社会を支えるインフラから私たちの身近な生活まで、さまざまな領域に浸透しています
- 人工知能の技術の一つである自然言語処理技術は、例えばAI音声アシスタントや機械翻訳、チャットボットなどに応用されています。
- Web検索では、探索の技術やエージェント技術などの、人工知能技術が応用されています。
- 探索は基礎的な人工知能技術であり、横型探索、縦型探索、最良優先探索、最適経路探索、A-アルゴリズム、A*-アルゴリズムなど、さまざまな方法が提案されています。
- ネットショップでは、人工知能技術、特に機械学習の技術を応用することで、おすすめ商品の提示などのサービスを実現しています。
- 生成AIは、大量のデータを学習した結果である大規模言語モデル(LLM)を利用した、対話応答システムです。

