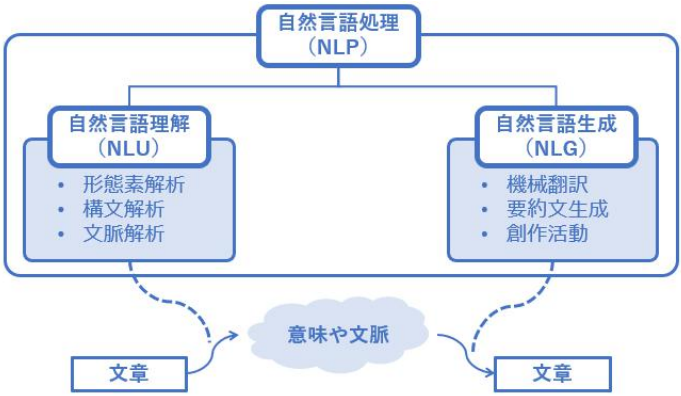
3章
自然言語処理入門

01

自然言語処理で何ができるのか？

前章では、AIや機械学習の仕組みについて理解を深めました。本章は生成AIへの序章として、自然言語処理について紹介します。自然言語処理はNLP（Natural Language Processing）とも呼ばれ、我々が使う自然言語（日本語や英語といった話し言葉）で書かれた文章をAIが読み取り、実行する処理です。自然言語処理の技術は、テキストの解析から意味の理解、さらには生成まで、幅広い領域にわたっています。この分野の発展には、統計学的手法や機械学習、最近では深層学習が大きく貢献しています。自然言語を処理することにより、AIは質問に答えたり、文章を要約したり、さらには人間の言葉で新しい文章を生成することが可能になります。 自然言語処理(NLP)は大きく自然言語理解(NLU)と自然言語生成(NLG)に大別されます。簡単にまとめると、文章から文脈や意味合いを抽出する処理がNLUであり、文脈や意味合いから文章を生成する処理がNLGとなります（図X-X)。



図X-X 自然言語処理

- 自然言語理解（NLU：Natural language understanding）
  - 自然言語理解は、AIが人間の言葉を理解し、その意味や文脈を把握するプロセスです。形態素解析、構文解析、文脈解析などを行って文章の内容を理解する処理であって、文章中の意図や情報を正確に解釈することを目的とします。NLUのアプリケーション例としては、文章分類、文脈認識、感

情分析、情報抽出などがあります。ユーザーが入力した自然言語のプロンプト（クエリを含む）に基づいて正確な情報を提供したり、テキストから特定の情報を抽出したりするのに役立ちます。
    - Google検索エンジンのアルゴリズムに使用されるようになったBERTが有名です
- 自然言語生成（NLG：Natural Language Generation）
    - 自然言語生成は、入力された文章の続きを生成したり文章の変換を行う処理であって、AIが自然言語で新しいテキストを作り出すプロセスです。NLGの用途として、機械翻訳、要約文生成、AIによる創作活動（例えば、物語や詩の作成）などがあります。
    - OpenAIのGPTなどが有名です

自然言語理解（NLU）の使いどころ

音声認識

スマートフォンやスマートスピーカーなどのデバイスでは、NLU技術を用いてユーザーの発話からコマンドを認識し、音楽の再生、ニュースの読み上げ、天気予報の提供、スマートホームデバイスの制御などを行います。多少の表現の揺らぎがあっても、ユーザーが実行したい処理を意図通りに実行したり、知りたい内容に対して回答する事が可能になります。

情報抽出と知識管理

NLU技術は、大量の文書やテキストデータから重要な情報を抽出し、構造化するのにも用いられます。インターネット検索はもちろん、企業において契約書、報告書、学術論文などから必要な情報を迅速に見つけ出し、知識の管理と活用を効率化が可能となります。

感情分析

NLUの応用例として、感情分析があります。これは、文章に含まれる感情や意見を識別する技術です。ソーシャルメディアの投稿、製品レビュー、顧客フィードバックなどから、ポジティブ、ネガティブ、ニュートラルなどの感情を自動で判定します。企業は感情分析を活用して市場のトレンドを把握したり、顧客満足度を評価することが可能となります。

自然言語生成（NLG）の使いどころ

機械翻訳（MT: Machine Translation）

ある文章に対して自然言語処理を行い、別の言語に変換するものです。生成 AI 技術は翻訳ツールとして開発されたものではありませんが、ある文章を特徴量空間に埋め込み、別の言語として取り出す[1]という所作を行うことによって、翻訳でも高い能力を発揮します[2]。

コンテンツ生成

特定のデータに基づき、人間にとって読みやすい概要を自動で作成する応用例です。近年では Google 検索結果に生成 AI による検索体験 (SGE: Search Generative Experience) が追加されています。同社は、目的の情報をすばやく簡単に見つける新たな方法と位置づけています[3]。

図 X-X SGE の例

---

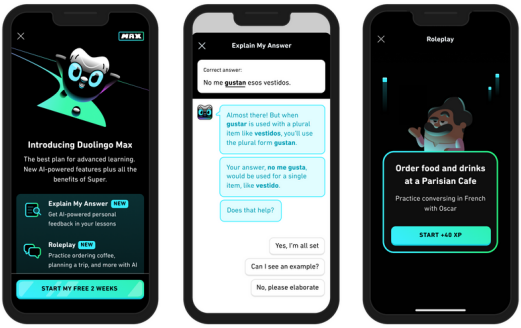
[1] 特徴量空間に埋め込む話は第一章でも紹介しました。

[2] 例えば 2023 年に公開された論文（https://doi.org/10.48550/arXiv.2301.08745）を紹介します。この論文は 2023 年 1 月に第 1 版が公開され、その際には「データが豊富なヨーロッパ言語等においては商業翻訳製品（Google 翻訳など）と競合する性能を発揮する一方で、データが少ない言語や遠い言語では大幅に遅れをとっている」旨の結論でした。その後に ChatGPT-4 が公開されたことによって論文も更改され、2023 年 11 月に公開された第 4 版では「GPT-4 では翻訳性能は大幅に向上し、遠い言語であっても商業翻訳製品と遜色ない性能である」旨の結論が述べられています。なお、当該論文における翻訳性能の評価には BLEU スコアが使用されています。BLEU スコアは機械翻訳結果の精度評価指標として広く利用されており、自動翻訳と人間が作成した参考翻訳との違を測定するものです。参考翻訳文と同一の単語列を含む出力文が高いスコアを算出するため、出力文において文法的な誤りが存在しても高いスコアを算出してしまうという欠点もあり、人間が翻訳文を評価した場合には異なるスコアが算出される場合があります。GPT4 が人間の翻訳者を完全に置換でき得るかという点については注意して考える必要があるでしょう。

[3] 同機能は 2023 年 8 月から日本国内でも試験提供が開始されました（https://japan.googleblog.com/2023/08/search-sge.html）。

チャットボットと仮想アシスタント

チャットボットについてはChatGPTを想像するとよいかもしれません。自然言語生成（NLG）技術を利用して、人間との会話をシミュレートする応用例です。 世界で1億人以上のユーザーを有する言語学習サービス「Duolingo」は2023年にOpenAI社のGPT-4を採用し、言語学習者が対話型AIチャットボットを通して会話力を鍛える機能を提供しています。学習者はチャットボットからフィードバックや励ましの言葉を受け取ることもできます[4]。



図X-X 言語学習チャットボットの例

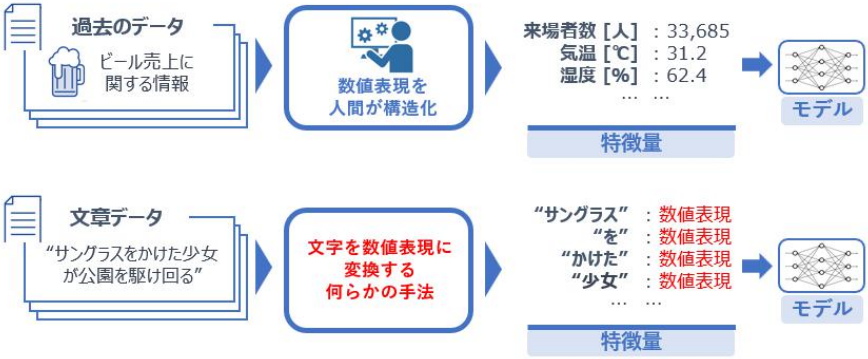
どのように機械学習すればいいの？

自然言語処理は我々の身近なアプリケーションに広く応用されているというイメージを持っていただけたと思います。ここで、第二章で学んだように、あらゆる機械学習モデルは数字で処理を行います。自然言語を何らかのカタチで数字表現に変換する必要があります。次節では、文章を数字表現に変換する技法を紹介します。

[4] 同機能は2023年3月に「Duolingo Max」として公開されました（https://blog.duolingo.com/duolingo-max/）。一方で2023年時点では一部の言語に限り提供されており、米国や英国などの一部地域でiOS版のみの提供となっています。

02 離散化 ～文章を区切る技術～

第一章で言及した通り、自然言語から成る文章は本質的に連続的で複雑なデータであり、時に長い文脈を持ち、一部を切り出しただけでは意味を成さないこともあります。一方で、機械学習モデルで自然言語を処理する際にはモデルが受け取れる形式で受け渡さなければなりません。例えば過去のビール売上に関連する数字で表された特徴量を扱う場合、最初に「来場者数」「気温」といったデータを構造化[5]してモデルに入力すればよい一方、文章データの場合は連続する文章表現を直接モデルに入れられないので、適当に文章を区切った上で数値表現に変換する必要があります。 すなわち、固定長の数字データ[6]に変換し、コンピューターが処理しやすい形式に変換（離散的な要素に分割）する必要があります。



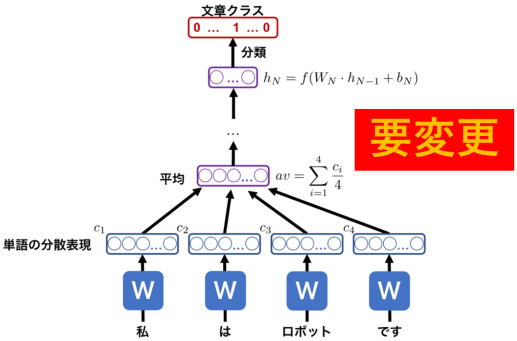
図X-X 文章を離散化するイメージ

[5] 構造化とはデータを整理して体系的に配置するプロセスで、端的に言えばExcelやCSVファイルのように「列」と「行」の概念を持たせることです。構造化データは検索、集計、比較などが容易になり、機械学習モデルへの直接的な入力が可能となります。一方で、日常で目にする契約書、帳簿、提案書といったフォーマット（構造定義）が無いデータは非構造データと区分されます。

[6] 機械学習モデルは幾つかの特徴量を基にして推論を行います。例えば「来場者数」と「気温」という2種類の特徴量から「売上」を予測するように学習したモデルに対して、推論段階で「湿度」という新たな特徴量を追加して推論を実行することはできません。様々な長さの文章がありますが、文章の長さに応じてモデルの中身を変えるわけにはいかないので、任意の文字列を所定の長さの数字（固定長の数字データ）に変換します。

文章を区切る

文章を数字に変換するにあたって、まず文章を適当に区切ることが必要です。文字を適当な単位に区切り、この各単位を特徴量とすれば、先に学んだ機械学習で処理ができそうです。このように適切な単位に分割する事をトークナイズ（tokenize）[7]と呼びます。英語の場合は、各単語はスペースで区切られているので機械的に処理することが可能ですが、日本語の場合は明示的な区切り記号がないので工夫が必要です。 本節では、具体的な区切り方について幾つかの手法を紹介します。



図X-X 文章を区切るイメージ

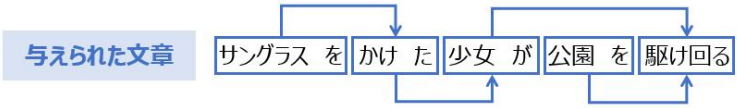
形態素解析

図X-Xの自然文章（日本語）を考えた時に、要素ごとに空白で区切りを入れます。この各要素のことを形態素（morpheme; 言語で意味を持つ最小単位）と呼びます。端的に言えば、語彙や文法をルール化し、「少女」は名詞といった具合で、ルールに基づいて分割を行います[8]。 形態素を品詞単位で区切ると、例えば名詞や形容詞といった特定の品詞だけ抜き出して分析を行ったり、係り受けにより文章を解析することができそうです。係り受けとは「文中の

[7] 日本語では「分かち書き」とも。BERTやGPT等の機械学習モデルでは、このような処理をエンコード（encode）と表現されます。

[8] 最もシンプルな手法は辞書のようにルールを定義するやり方なのですが、2000年前後からは日本語文書の単語出現頻度を考慮して機械学習を行う形態素解析手法（Mecab、Kuromoji、JUMAN、Chasen、など）が現れました。昨今は深層学習を利用した手法（JUMAN++、など）も提案されています。

言葉同士の関係性」の事で、係り受け解析とは文中の各要素（単語ないし文節）がどのように関連し合っているかを解析する技術を指します。これは、文を構成する要素間の「係り受け関係」を明らかにすることで、文の意味構造を理解するための重要な手段です。日本語では動詞や助詞などが、文節間の関係性を示す重要な因子となり、これらを解析することで文の全体的な意味を捉えることができます。 例えば「サングラスをかけた少女が公園を駆け回る」という文章において、「少女が」が「駆け回る」に係る主語であること、「公園を」が「駆け回る」に係る目的語であることなど、各文節がどのように他の文節に関連しているかを解析します。この解析により、文の構造を理解し、より複雑な自然言語処理タスクへの応用が可能となります[9]。



トークン	文節の係り受け関係	
サングラス	サングラスを	→ かけた
かけ	かけた	→ 少女が
少女	少女が	→ 駆け回る
公園	公園を	→ 駆け回る

図X-X 係り受け解析

サブワード分割

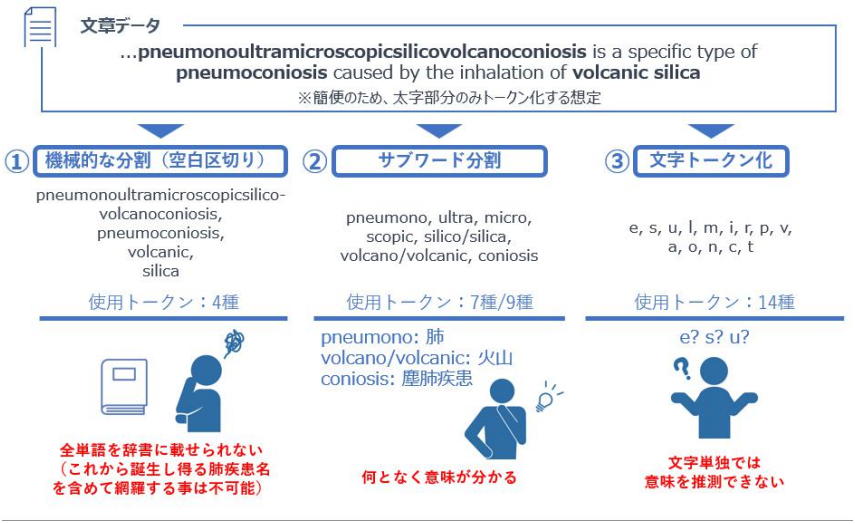
先ほどのように形態素や文節で文章を区切ってしまえば離散化自体は可能そうですが、世の中にある全ての単語を列挙することは非現実的です[10]。更に、学習時に登場しなかった未知語に対しては上手く対処できません[11]。 こ

[9] 図 X-X 中の係り受け解析には自然言語処理ライブラリ「GiNZA」（ギンザ）を用いました。「GiNZA」は株式会社リクルートのAI研究機関であるMegagon Labsと国立国語研究所との共同研究により生まれた日本語自然言語処理オープンソースライブラリです。先進的な自然言語処理ライブラリ「spaCy」をフレームワークとして使用しており、日本語の形態素解析器「SudachiPy」を使用しています。（https://www.recruit.co.jp/newsroom/2019/0402_18331.html）

[10] 仮にこのようなことが出来たとしても、ほぼ無限の語彙数（vocaburary）が必要となり、計算量が増えてしまいます。

[11] 例えば、「マイナポイント」という語句が辞書に登録されていないと「マイナ」「ポイント」（これは2013年に公開された"unidic-mecab-2.1.2"という辞書を用いた場合の分割例です。マイナポイント事業が開始されたのは2019年でしたので、辞書作成当時は未知語でした）といった具合で、意図せずに分割されてしまうことが考えられます。日本語形態素解析における未知語処理手法として、既知語からの派生ルールを用いる方法などが提案されていたりもします（https://doi.org/10.5715/jnlp.21.1183）。

ういった背景から考案されたのが、単語を更に小さな単位（サブワード）に分割するサブワード分割です。極端な話ですが、英語文章を全てサブワードに分解すると、大文字と小文字のアルファベットで計52種になります。ここまで分割すると、全ての英文は52種の語から構成されることになりますので、語彙数（vocaburary）を圧縮できます[12]。論文1本でも学習させれば、全アルファベットが使用されているでしょうから、基本的に未知語（というより未知の文字）に遭遇することはなくなるでしょう。例えば、図X-Xのような具体例を考えます。



図X-X サブワード分割のイメージ

この例では、次の旨の英文が与えられたと仮定します。

- **超微視的珪質火山塵肺疾患は火山塵の吸引によって引き起こされる塵肺の一つである**

英語文章は空白文字で文章が分かれているので、機械的に分割してトークン化する手法[13]（図中①）が考えられます。与えられた文章を空白文字で区

[12] このような手法を、character 分割、文字トークン化、"character-level tokenization"、"character-based tokenization algorithm"などと言います。
[13] ヨーロッパ系言語（ドイツ語、ロシア語、英語、等）のトークン化を行うライブラリとしてmosetokenizer 等が有名で、基本的には空白文字で区切られます（例："Hello, AI!"を"Hello"+","+"AI"+"!"に分割する）

切ってトークン化した後、今回は次の４つの単語に注目しすることにします。

- **超微視的珪質火山塵肺疾患（pneumonoultramicroscopicsilicovolcanoconiosis）**[14]
- **塵肺疾患（pneumoconiosis）**
- **火山の（volcanic）**
- **珪質（silica）**[15]

この世に存在する全ての単語を掲載した辞書があれば、その辞書に含まれている単語全てをトークンとしてモデルに与えられます。一方で、超微視的珪質火山塵肺疾患のような低頻度語を含めて網羅すれば膨大な辞書となってしまい、且つ未知の単語に対しては対応できなくなってしまうという欠点があります。

次に、注目する４単語についてアルファベットで１文字ずつ分割する事を考えます（図中③）。４つの単語に含まれるアルファベットは14種類ですので、14種のトークンが得られます。この手法はシンプルですが、各トークンに注目すると「e」や「s」単独になり、元々の単語の意味を損失してしまいます。このように文字ごとに区切ってしまうと単語やフレーズのような、より大きな意味の単位が失われるため文脈を捉えるのが難しくなり、文章の意味を理解することが困難になります。例えば日本語においても、「火山」という単語を考えたとき、この単語単独で"volcano"という意味を持ちますから、「火」（"fire"）と「山」（"mountain"）という２つに分割しない方が、文章中における本来の意味を保持する事が出来そうです。 そこで予め用意した大量単語を学習して、その中で隣り合う頻度が高い文字列を結合するルールを作っておけばどうでしょう。例えば、"**pneumo**nia"（肺炎）、"**pneumo**thorax"（気胸）など、学習した文章中において**pneumo**[16]）の出現割合が高ければ、単独のトークンと定義する、といった具合です。このように分割する手法がサブワード分割（図中②）であり、元々の単語の意味をある程度保ちつつ、複雑な単語（未知の単語[17]）に対しても効果的に分割で

[14] 同疾患は45文字の英単語で表現されます。英語辞書として著名なOED（Oxford English Dictionary）に掲載されている中で最も長い単語の一つです。
[15] "volcanic silica"で「珪質（シリカ; SiO2）の火山塵」を表わします。
[16] "pneumo-"は、肺や空気に関する単語に使われる接頭辞です。

きます。

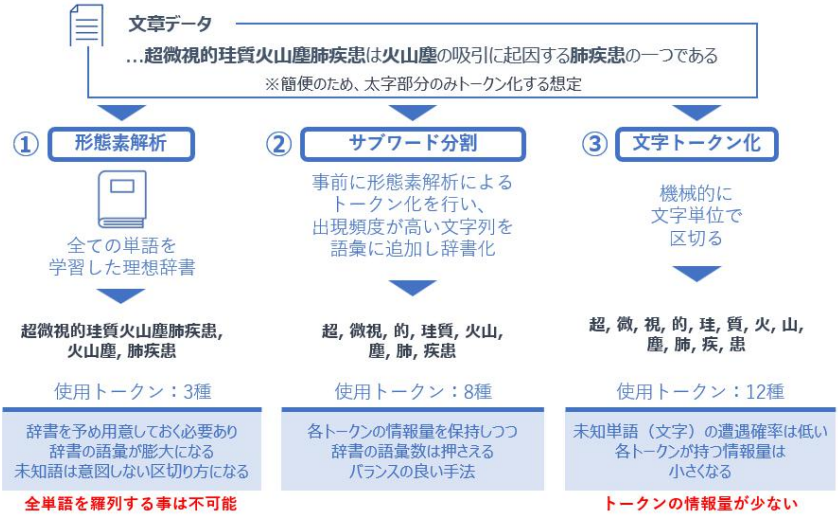
実際には、機械学習によって適切なサブワード単位が決定されます。単語単位に分割し、分割された各単語に対して、サブワード分割手法（BPE[18]、WordPiece[19]など）を用いてサブワード単位に分割します。れらの手法は主に教師なし学習が用いられていて、文章データさえあればトークナイザを構築できます。サブワード単位に分割された語を言語モデルへの入力に用いることで、低頻度語に対して頑健性を向上させます。

次に日本語でサブワードを学習する場合を考えましょう。英文の場合は空白文字で機械的に区切ることが出来ましたが、日本語の場合は形態素解析を行って文章を区切る必要があります。図X-Xのように、形態素辞書を用いて単語単位に分割する点が、英文との違いです。図X-Xでは、先に提示した英文の日本語版を例として用います。簡便のため、「超微視的珪質火山塵肺疾患」「火山塵」「肺疾患」の3単語に注目します。

[17] 仮に"pneumonoultramicroscopicsilicovolcanoconiosis"という単語が学習データの中に入っていなかった（未知語であった）場合でも、pneumono（肺）、ultra（超）、microscopic（微視的）、silico（珪素）、volcano（火山）、coni（塵）、osis（疾患）といったサブワードに分割できます。各サブワードには（少なくともアルファベット1文字以上の）意味が含まれているため、分割前の単語に含まれた意味をある程度保持する事が期待できます。

[18] BPE（Byte Pair Encoding; バイトペア符号化）とは、文字の繋がりを見て、頻出する文字同士を結合させる手法です。要するに、単語内の繋がりやすいアルファベットの組み合わせをサブワードとするものです。日本語においては元々の語彙数がアルファベットよりも段違いに多いので、そのペアを探して登録する事を繰り返すと、語彙数が逆に大くなってしまうことが知られています。

[19] 考え方としてはBPEに近いです。WordPieceは最も頻度の高いシンボルペアを選択するのではなく、一度語彙に追加された学習データの尤度を最大化するものを選択します。



図X-X 日本語におけるサブワード分割のイメージ

日本語の場合には文章を区切りるに当たって形態素解析が必要です（図中①）。ここで全ての単語が網羅されている形態素辞書があるとすれば、「超微視的珪質火山塵肺疾患」を単一のトークンとして認識できますが、全単語を網羅した形態素辞書は非現実的です。日本語の場合も大量単語を学習し、隣り合う頻度が高い文字列を結合するルールを作り、それに基づいてサブワード分割を行うこと（図中②）によって、各トークンの情報量を保持しつつ辞書の語彙数を抑える事が期待出来るのです。

SentencePiece

日本語の場合はサブワード分割を学習する際にトークンに分割する必要があり、最低限の辞書は必要になってしまいます。それなら最初から形態素に分けずにトークン分割してしまおう、という考え方に基づく手法がSentencePieceです。この手法は言語に関する事前知識を必要とせず、生のテキストデータから直接サブワード単位を学習します。英語圏言語で用いられている空白スペースも文字として扱うので、日本語や中国語のように、空白文字区切りではない言語と同様に扱うことが可能です。 この手法はGoogle社が開発し、GitHub上でオープンソースとして公開[20]されているため、研究者や開発者は自由に利用できます。 SentencePieceは、特にトランスフォー

マーベースのモデル（例えば、BERTやGPT）の前処理として使用されています。

まとめ

形態素による係り受け解析は、助詞や名詞のデータベースに基づく、人が作成したルールベースの処理によって文章を解析します。自然言語の文法的構造を理解する上で重要であると考えられています。このようなルールベースのアプローチは、特定の言語の文法規則や特性を詳細に反映することができるため、高い精度での解析が可能です。一方でこの手法は、ルールの作成とメンテナンスに多大な労力が必要であり、未知の表現や言語の変化に対応するためには定期的な更新が必要となります。

BERTやGPT等の現代的な機械学習モデルでは、Encode後のトークン間の関連性を学習して、人が作成したルールに依存せずに自ら係り受けの学習（に近いこと）をしていると考えることができます。このように自ら特徴を学習するには、Self-Attentionという要素技術が重要なのですが、こちらについてはTransformetに関する説明の章で述べます。 OpenAIのGPTシリーズでは、tiktokenというトークナイザー[21]が使用されています。図X-Xに「こんにちは。今日の天気は？」を分解した様子をしまします。「こんにちは」は単一のトークンとして処理されていますが、それ以外の文字は単独で区切られています[22]。

[20] GitHubはソフトウェア開発のプラットフォームで、個人や企業を問わず幅広いユーザーがコードを共有しています。8,000万件以上ものプロジェクトが管理されており、SentencePieceは次のURLにてApache2.0ライセンス下で公開され、尚も継続的に開発とメンテナンスが行われています（https://github.com/google/sentencepiece）。

[21] 高速BPEトークナイザーで、トークナイザーの挙動はOpenAI社のWebページ上で確認することが出来ます（https://platform.openai.com/tokenizer）。

[22] GPT-4やGPT-3.5では、「cl100k_base」というエンコーディングが使用されています。例えば、"こんにちは。今日の天気は何ですか？"という文章をトーカナイズすると、次の12トークンに分割されます。 トークン：['こんにちは', '。', '今', '日', 'の', '天', '気', 'は', '何', 'です', 'か', '？'] これらの各トークンにはIDが割り振られており、モデルに入れる際には文字ではなく、これらのIDで識別されることになります。（トークンID： [90115, 1811, 37271, 9080, 16144, 36827, 95221, 15682, 99849, 38641, 32149, 11571]）

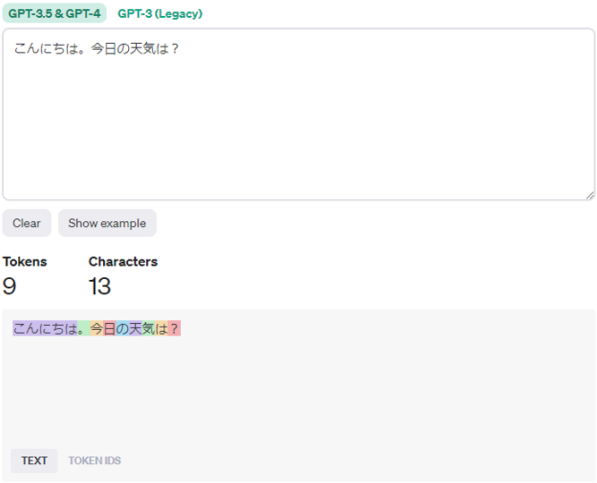


図 X-X GPT における文章の区切り方

人間の感覚からすれば、「今日」や「天気」という語句は単独で取り出した方が意味を成しそうですが、多言語の大量文章を学習した結果、これらは分けた方が都合が良いという結果になったのでしょう。現代的な機械学習モデルで使用している Self-Attention が強力なのは、人が作成 したルールに依存せずに、広域的な単語（Encode した後のトークン単位）間の関連性を学習可能である点です。例えば係り受けを学習するにあたって、人間が作成したルールに基づいて分割された単語間の係り受けを学習する限り人間の性能を超えることは難しいでしょう。ルールベースの形態素解析ではなく、更に細かいトークン単位での学習を行う大きな理由です。 Self-Attention メカニズムを採用した現代的な大規模言語モデルは、単語やフレーズの分割方法に対する人間の直感的な理解を超える能力を持っています。これらのモデルは、大規模なデータセットから複雑な言語パターンを学習することで、より洗練された言語理解と処理を実現します。この進歩は、人間が作成したルールベースのシステムでは到達できない新たな可能性を開くものであり、自然言語理解の分野における今後の研究や応用に大きな影響を与え続けるでしょう。

参考：前処理手法

ここでは、参考として文章を離散化する際に用いられる前処理手法を紹介します。実際には、分析対象となる文章に合わせて複数の手法を組み合わせて適切な分析を行います。

正規化（表記揺れを是正）

文字種の統一、つづりや表記揺れの吸収といった、同じ意味を持つ単語表現を統一する処理をします。扱う単語種類を減少させることが出来るので、後続の処理における計算量やメモリ使用量の観点から見ても重要な処理です。

- 文字種の統一
  - 文字種の統一ではアルファベットの大文字を小文字に変換する、半角文字を全角文字に変換する、といった処理を行います。たとえば「Cat」の大文字部分を小文字に変換して「cat」にしたり（ケースノーマライゼーション）、「ﾈｺ」を全角に変換して「ネコ」にします。このような処理をすることで、文字種の区別なく同一の単語に変換することで情報を一貫して扱うことが可能になります。Unicode正規化[23]と呼ばれる手法が有名です。
- 数字の置き換え
  - 数字の置き換えでは文章中に出現する数字を別の記号に置き換えます。例えば、ある文章中に「2017年10月10日」のような数字を含む文字列が出現したとしましょう。数字の置き換えではこの文字列中の数字を「0年0月0日」のように変換してしまいます。自然言語処理のタスクにおいては、文章に含まれる数字自体は役に立たないことが多いからです。 例えば、ニュース記事のジャンルを分類するタスクを考えてみましょう。この場合、記事の公開日付はジャンル分類に直接影響を与えないため、数字の置き換えを行っても問題ありません。他方で、経済ニュースの分析では記事に記載されている具体的な数字（例えば株価や経済指標）が重要な役割を果たす場合には、数字の置き換えは適切ではないかもしれません。自然言語処理における数字の扱いは、タスクの性質に大きく依存します。数字を無視することで情報の損失が生じる可能性があるため、数字を置き換えるかどうか、またどのように置き換えるかを慎重に検討する必要があります。例えば、文脈に応じて数字を保持するか、特定のタグで置き換えるか、あるいは完全に除去するかなど、タスクの目的に応じて最適な処理方法を選択

[23] Unicodeとは、様々な言語や記号に番号（文字コード）を割り当てて定義した標準規格。全角半角の統一以外にも、「①」と「1」といった等価な文字表記を統一する処理も実行可能です。

することが重要です。

- 辞書を用いた単語の統一
  - 辞書[24]を用いた単語の統一では、各単語を代表的な表現に置き換えます。たとえば、「オレンジ」と「orange」という表記が入り混じった文章を扱う時に、どちらかの表現に寄せて置き換えてしまいます。これにより、これ以降の処理で2種類の単語を同じ単語として扱えるようになります。置き換える際には文脈を考慮して置き換える必要があることには注意する必要があります。

ストップワード(stop word)

ストップワードとは、自然言語処理を実行する際に一般的で役に立たない等の理由で処理対象外とする単語のことです。たとえば、助詞や助動詞などの機能語(「は」「の」「です」「ます」など)が挙げられます[25]。これらの単語は出現頻度が高い割に役に立たず、計算量や性能に悪影響を及ぼすため除去されます。 ストップワードの除去には様々な方式がありますが、この記事では以下の2つの方式を紹介します。

- 辞書による方式
  - この方式では、あらかじめ定義されたストップワードのリスト（辞書）を使用して文章からストップワードを除去します。この辞書は、その言語で一般的に使用される前置詞、冠詞、接続詞などの単語を含んでおり、自然言語処理においてほとんど意味を持たないと考えられる単語から構成されます。辞書を用いる利点として、シンプルで実装が容易である点が挙げられます。一方で、辞書が固定されているため、特定のドメインやコンテキストに特有のストップワードをカバーしきれない可能性があります。
- 出現頻度による方式
  - この方式では、テキスト全体での単語の出現頻度を分析し、あまりにも頻繁に出現する単語や、逆にほとんど出現しない単語をストップワードとして扱います。頻繁に出現する単語は、一般的に内容の理解にあまり寄与しないと考えられるため、ストップワードとして除外されます（例えば、「の」「あの」「この」など）。一方、非常に稀にしか出現しない単語も重要

[24] 有名な辞書としてWordNetが挙げられます。単語間の意味的関係を体系的に整理した大規模な辞書データベースで、単語を「シノニムセット（synsets）」と呼ばれる同義語のグループに分類し、これらのグループ間の様々な意味関係（例えば、上位語/下位語関係、部分/全体関係など）を定義しています。異なる単語が同じ概念を指している場合や、同じ単語が複数の意味を持つ場合、WordNetを使用してこれらの単語や意味の関係を特定し、文脈に応じて表記ゆれを是正することが可能です。
[25] 英語のストップワードとしては、"the"、"a/an"、"is"、"at"、"which"、"on" などの前置詞、冠詞、接続詞が挙げられます

な意味を持たない場合が多いため、除去の対象となることがあります。与えられたデータに対して柔軟にストップワードを定義できますが、単語の出現頻度だけを基準にして機械的に除去してしまうと、重要な情報を持つ可能性のある単語を除去してしまうリスクも伴います。 また、複数の文章中における各単語の出現頻度に基づいて単語の重みを調整するTF-IDFといった手法をもあります（後述）。

まとめ

自然言語で書かれた文章の前処理には、先述した正規化やストップワードの除去といった処理以外にも複数の手法が考えられます。各言語に特有なルールを適用する必要もあるかもしれません。例えば日本語では、ひらがなとカタカナ、漢字の使用に関する正規化が必要になる場面もあるでしょう。 文章データから分析に不要な表現の揺らぎを取り除き、データを機械学習モデルやその他の分析手法で扱いやすい形に整形する処理は分析精度の向上に不可欠です。どのような前処理手法を適用するか、またその程度は、次ような観点に基づいて慎重に選択する必要があります。

- 文章データの性質
  - 文章で扱われている言語や専門分野によって、適切な前処理手法が異なります。例えば、技術文書や医療記録などの特定のドメインでは、稀有な専門用語であっても重要な意味を持つと考えられるため、これらをストップワードとして除外すべきではないでしょう。
  - SNSの投稿、ニュース記事、学術論文など、テキストの形式や出典によって、言語の使用法やスタイルが異なるため、前処理のアプローチも変わってきます。例えば、絵文字や顔文字は、その感情的な表現に意味があったり、何かを代替して表現する場合があるため、どのように処理を行うかは検討に値します。
- 分析の目的
  - 分析の目的に応じて特定の単語やフレーズが重要になったり、逆に無視すべき場合もあります。例えば、感情分析では否定語が重要な役割を果たすため、これらを除外しないようにする必要があります。

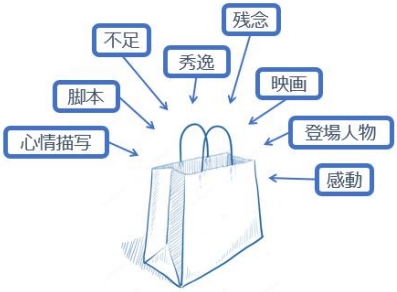
03 単語文章行列 ～BOWとTF-IDF～

前節では文章をトークンに分割するイメージについて理解を深めました。一方でこのままでは尚も数字ではないため、数値的に解析可能な形式に変換するため（モデルに入力するため）には何らかの処理が必要そうです。各トークンに対して、例えば文書に出現する単語の頻度を数値化すれば、各トークンの特徴を表現できそうです。このように作成したデータを単語文書行列（document-term matrix）と言い、文章を数値化する上で基本的な手法です。本節では、文章データを構造化データに変換する手法として BOW と TF-IDF を紹介します。

Bag of Words (BOW)

Bag of Words（BOW）モデルは、テキストデータの分析においてよく用いられる手法の一つであり、文章内に含まれる単語の出現回数に基づいて固定長のベクトルとして表現します。例えば、次の３つの映画レビュー文書があった場合、BOW モデルは各文書に含まれる全てのトークンを収集し、各トークンがその文書に何回出現するかをカウントします。

- ① 脚本が秀逸で映画全体を通して感動した
- ② 映画の登場人物の心情描写が不足しており映画全体としては残念だった
- ③ 登場人物の心情描写が不足していたのは残念だが、脚本は秀逸で感動する映画



図X-X Bag of wordsのイメージ

図X-Xのように、全文章から収集した全種類のトークンをバッグに入れるイメージです[26]。各文章に出現するトークンは、このバック内のトークンで網羅されているはずですから、バッグ内のトークンを各列にとって、各文章における出現回数を集計して表にまとめてみます。この表内は数字の羅列ですので、文字表現を数字表現に変換する事が出来ました（図X-X)。

	脚本	秀逸	映画	全体	登場人物	心情描写	不足	残念	感動
文章①	1	1	1	1	0	0	0	0	1
文章②	0	0	2	1	1	1	1	1	0
文章③	1	1	1	0	1	1	1	1	1

文章①のベクトル表現： [1, 1, 1, 1, 0, 0, 0, 0, 1]
文章②のベクトル表現： [0, 0, 2, 1, 1, 1, 1, 1, 0]
文章③のベクトル表現： [1, 1, 1, 0, 1, 1, 1, 1, 1]

ベクトルの成分の大きさに色付けを行った可視化例

文章①のベクトル表現： 1 1 1 1 0 0 0 0 1
文章②のベクトル表現： 0 0 2 1 1 1 1 1 0
文章③のベクトル表現： 1 1 1 0 1 1 1 1 1

図X-X Bag of wordsで数字表現（ベクトル表現）を獲得するイメージ

図X-X中の表を眺めてみると、元の文章を読まなくても次のような分析はできそうです。

[26] 簡便のため、例文中に含まれる「した」「の」といった助詞や助動詞は無視する事にします。実際の分析においても、このような品詞は文章の意味合いを推測する上では相対的に重要性が低く除外される場合があります。

- 文章①には「脚本」「秀逸」「感動」といったトークンが出現するので、脚本に対してポジティブな感想を持っている
- 文章②には「心情描写」「不足」「残念」といったトークンが出現するので、登場人物の描き方に対するネガティブな感想を持っている
- 文章③には「脚本」「心理描写」「秀逸」「不足」「残念」「感動」といった多くのトークンが出現するが、何に対してポジティブ（あるいはネガティブ）な感想を持っているか分からない

上記の通り、BOW モデルは文書を固定長のベクトルに変換する単純で直感的に理解しやすい手法です。一方で全種類のトークンを同じ重要度で考えますので、レビュー結果の特徴が込められていない「映画」というトークンと、レビュワーの感想にである「残念」や「感動」といった、レビューを参考にするあたって重要なトークンとが（出現回数が同じであれば）同値な特徴量として算出されるという短所があります。

TF-IDF

TF-IDF は BOW モデルをさらに洗練させた手法で、文章内におけるトークン出現頻度（TF）だけでなく、そのトークンが各文章の特徴付けにおいてどれだけ重要であるかという点（IDF）を加味して重み付けを行います。

例えば先ほどの映画レビューの文章を考えてみましょう。３つ全ての文章に「映画」というトークンが出現するため、各文章の特徴を考える手がかりとはならず、重要度は低そうです。このように、多くの文書で登場するトークンは個々の文書を特徴付ける単語として重要度合を下げるというコンセプトを定式化した値が IDF（inverse document frequency）[27] です。図 X-X の定義式を使って計算すると、「映画」の IDF 値は 0 と計算され、それ以外のトークンについては 3 文章中、いずれも 2 文章に出現するので、log(3/2)≒0.41 と算出されます。

[27] 別の言い方をすると、トークンの文章間出現頻度（DF）の逆数（inverse）です。全ての文章において共通して出現するトークンの重要度は低く（IDF 値が小さくなる）、出現頻度の低い単語は逆に重要度が上がります（IDF 値が大きくなる）。また、IDF 値の計算式はいくつか存在し、scikit-learn（著名な Python 用の機械学習ライブラリ）では、IDF=log(全文章数/あるトークンが出現する文章数)+1 と定義されています（本文中の式に 1 を加えたもの）。いずれの定義においても、考え方の大筋は同じです。

$$\mathrm{IDF} = \log\left(\frac{\text{全文章数}}{\text{あるトークンが出現する文章数}}\right)$$

	脚本	秀逸	映画	全体	登場人物	心情描写	不足	残念	感動
全文章数	3	3	**3**	3	3	3	3	3	3
トークンが出現する文章数	2	2	**3**	2	2	2	2	2	2
IFD値	0.41	0.41	0.0	0.41	0.41	0.41	0.41	0.41	0.41

図X-X IDF値の計算イメージ

次に、各文章においてどのようなトークンが多く使われているかを定量化してみましょう。図X-Xのように、ある文章に注目した時、その文章における各トークンの出現割合を求めることによって、当該文章を特徴づけるトークンを定量化できそうです。この指標をTF（term frequency）と呼びます。[28]

$$\mathrm{TF} = \frac{\text{ある文章内における、あるトークンの数}}{\text{ある文章内における、全トークンの数}}$$

	脚本	秀逸	映画	全体	登場人物	心情描写	不足	残念	感動
文章①	1/5	1/5	1/5	1/5	0	0	0	0	1/5
文章②	0	0	2/7	1/7	1/7	1/7	1/7	1/7	0
文章③	1/8	1/8	1/8	0	1/8	1/8	1/8	1/8	1/8

図X-X TF値の計算イメージ

全トークンの重要度はIDF値で定量化することができていますので、各文章におけるトークンの重要度合であるTF値を組み合わせれば、各トークンの重要度（多くの文書で登場するほどIDFが低く、登場頻度の少ない特異な（レア度が高い）トークンであるほどIDF値が高くなる）を考慮した上で、各文章の特徴（文章内で出現割合が高ければTF値は高くなる）を表現できそう

[28] 例えば、「この映画は脚本が**感動**的だし、心情描写にも**感動**したし、全体的に**感動**した」といった文章があれば、「感動」というトークンが当該文章中で相対的に出現頻度が高いため、「感動」というトークンがこの文章を最も特徴づけている、という考え方になります。TF値は、この考え方を指標化したものです。

です。このように、両者を掛け合わせた指標をTF-IDFと呼び、BOWと同様にベクトル表現として獲得できます。

図X-X TF-IDF値の計算イメージ

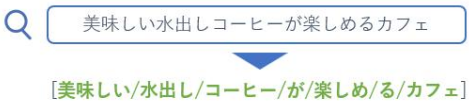
先に紹介したBOWでは文章中に各トークンを含む/含まない、でしか文書の特徴を抽出できなかったのに対し、図X-X中の表を眺めてみると各トークンの重みが評価されているため、より詳細な文書の特徴を抽出できていることが分かるかと思います。 TF-TDFを使うと、一般的なトークン[29]が複数の文章を跨いで高頻度で出現しても、重要度は低く評価され、特定のトピックに特化した単語が強調されるのです。

TF-IDFを用いた検索の例

このTF-IDF値は検索を行う際によく用いられます。例えば、ある地域に出店する複数のカフェで様々なコーヒーが提供されていると想像してください。この地域では「水出しコーヒー」を提供するカフェはごく一部でしか提供されていないとします。仮に、この地域のカフェの口コミが集まる掲示板で

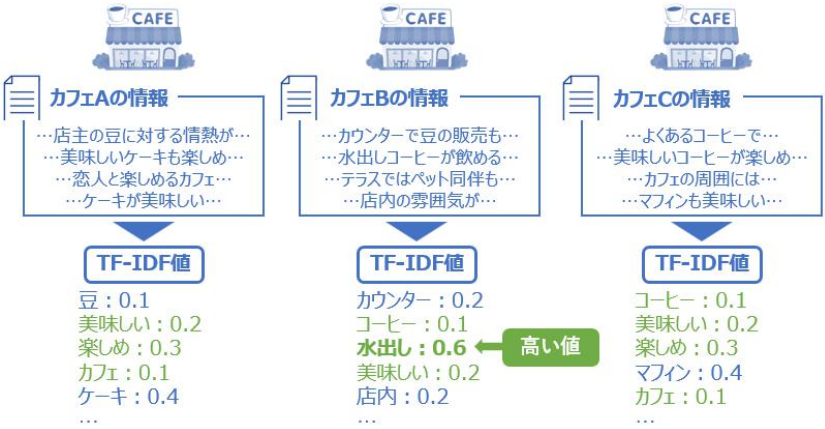
[29] 例えば「その」「この」などの指示語、「した」「の」などの助詞や助動詞、「です」や「ます」の文末敬体などです。「、」「。」といった句読点も、IDF値で低く産出されるでしょう。一方で、これらのトークンについてはストップワードとして分析対象から予め除外しておくのも一手です。

「美味しい水出しコーヒーが楽しめるカフェ」という検索文言で検索を実行したとしましょう。



図X-X 水出しコーヒーを提供するカフェを検索するイメージ

掲示板ではカフェごとに様々なレビューが集まっているので、事前に各カフェのレビューを分析してTF-IDF値を算出しておけば、各カフェの特徴を定量化しておく事ができます。



図X-X カフェの特徴を示すTF-IDF値

カフェを検索するにあたっては、今回の検索語句をトークン化して、そのトークンがレビューに数多く含まれるカフェを抽出すれば良さそうです。図X-Xでは、今回の検索文言に含まれる代表的なトークン[30]を緑色で示しました。このトークンの数だけに注目すると、カフェCがトークンの一致度合いが高そうです[31]。ただし、よく考えてみると「コーヒー」「カフェ」といっ

[30] 先の例と同様に、文章中の「で」「が」「る」といった文字は無視します。
[31] 検索文言に含まれる4種のトークン（「コーヒー」「美味しい」「楽しめ」「カフェ」）がカフェCのレビューに含まれています。カフェBでは、3種のトークン（「コーヒー」「水出し」「美味しい」）しか含まれていませんので、BOWの考え方に基づいて計算するとカフェCの方が類似度は高くなります。

たトークンは、カフェのレビューにおいては頻繁に用いられるため、今回の検索においては重要度合いを下げた方がよさそうです[32]。

ここでTF-IDFの考え方を導入してみましょう。この地域では水出しコーヒーを提供するカフェの数は少ないので、「水出し」というトークンの頻出度合いは低く（つまりIDF値は高く）なり、「コーヒー」や「カフェ」といったトークンは複数のカフェのレビューに登場するのでIDF値は低くなります。図X-X内にある3つのカフェのレビュー分量が同程度だとすると、カフェBにおける「水出し」のTF-IDF値は際立って高くなります。反対に、「コーヒー」という普遍的なトークンについては低い値が算出されます。この緑色で示されたTF-IDF値の合計値が高い順にカフェを提示すれば、検索実行者の意に沿った検索結果となりそうです。

トークンの重要度合を考慮して文章の特徴を抽出するTF-IDFですが、TFの計算過程で文書の長さを考慮できないという短所があります。TFの計算式の分母は「ある文章内における、全トークンの数」ですので、仮に10個のトークンで構成される文章と、1,000個のトークンで構成される文章では、同じトークンであってもTF値が変化してしまいます[33]。このようにTF-IDFでは文書の長さによって計算結果に影響が出てきてしまいます。 この点を改良したBM25という手法[34]があり、同手法を採用した有名な検索エンジンとして、オランダに本社を置くElastic社が提供するElasticsearchがあります。このElasticsearchはWikipedia、Amazon、GitHubといったサイト内検索に用いられており、主要機能はオープンソースで公開されています。

まとめ

本節では、文章データを機械学習モデルに投入する手間のデータ変換手法

[32] 人間の感覚的には「わざわざ**水出し**コーヒーと書く程だから、レビューに**水出し**コーヒーと明示的に記載されているカフェを優先的に検索結果として提示すればよさそうだな」と分かりそうなものです。このようなキーワード（重要単語）に対する人間の感覚を定量化したものがTF-IDF値であるとイメージすると分かりやすいかもしれません。

[33] 例えば、文章Aと文書Bにおいて「水出し」というレアなトークンが各2つ含まれていた場合、TF値はそれぞれ2/10と2/1,000となり、100倍の差が生じます。今回の例では、「水出し」コーヒーを提供していること自体が大きな特徴になりますが、文章Bでは含まれるトークン種が多いがために、文章内における「水出し」というトークンに対する重要度が相対的に低く算出されているのです。

[34] BM25はElasticsearchにおける標準の類似度ランキング（関連性）アルゴリズムです。BM25のアルゴリズムについては本書では紹介しませんが、Elastic社のBlogで紹介されています（https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables）。

として、BOWとTF-IDFを紹介しました。これらの手法は、文章に含まれるトークンの出現順序に関係なく処理を行うため、文脈情報が無視され、文の意味を正確に捉えるのは難しい場合があります。例えば、「山より海が好き」という文と「海より山が好き」という２つの文章を考える場合、これら２文章は同じベクトル表現になります。これは、両文ともに「海」「山」「より」「が」「好き」という単語が１回ずつ出現するためBOWで作成したベクトルは両文章で一致し、両文章共に合計トークン数が一致するのでIF-IDF値も同一になるためです。このように、文脈や単語間の関係性を無視するため、文の意味を完全に捉えることができないという欠点もありますが、文章分類や検索、さらには感情分析など、多岐に渡る身近なアプリケーションで応用されます。

第４章では、実際にTF-IDF手法を用いて文章分類問題を解いてみましょう。

MEMO　文字の揺らぎをどう吸収するか

本文中で紹介したTF-IDFの考え方はシンプルながら検索システムにおいては有効に作用します。一方で、分割単位がトークンなのでトークンの表記揺れを上手く吸収する工夫が必要です。例えば"swimming"は泳ぐことを表す"swim"の活用形の一つですが、綴りは異なれど同一のトークンとして扱った方が検索する場面では便利そうです。このようにトークンの語幹（今回の例では"swim"）を取り出す処理をステミング（stemming）といい、その単語の原型に変換します。動詞の活用形に留まらず、複数形を単数形[35]に変換するなどの処理を行います。Wikipediaの検索機能を使って"Swim suit"で検索してみると、検索にヒットした語句が太字で強調された状態で検索結果が一覧表示されます。太字の語句を確かめると、"**S**wim"ではなく"simmeres"が強調されるなど、文頭の大文字小文字や活用系の差分に捕らわれすに目的の記事をトップに表示していることが分かります。

[35] 機械的なステミング処理では、単純に過去形の"ed"や進行形の"ing"、複数形の"s"や"es"を語尾から削除する処理が実行されます。不規則な複数形（"mouse"/"mice"、"foot"/"feet"など）や不規則動詞（"go"/"went"/"gone"など）については対応できない場合もあります。

図X-X 英語版Wikipediaで"Swim suit"で検索した例

実は皆さんが身近に用いている検索システムの裏側には、このような工夫が実装されており、検索結果画面を注意深く観察してみると工夫の断片を類推することができるかもしれません[36]。

最近では深層学習を用いた手法が台頭しており、より複雑な文書やクエリの関連性を捉えることが可能になっています（Google社の検索機能は、Transformerがベースの自然言語処理モデルが用いられていると言われています）。それでもなお、TF-IDFはその理解のしやすさと速度的な優位性から、多くのアプリケーションで基礎的な技術として利用され続けています。

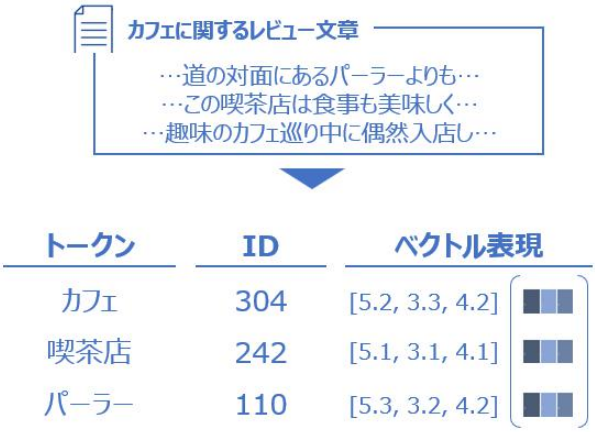
[36] Wikipediaのサイト内検索にはElasticsearchが用いられており、本文中で紹介したステミング機能を実行する"Stemmer token filter"が実装されています（https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-stemmer-tokenfilter.html）。

04 word2vec（Skip-gram, CBOW）

前節で紹介したBOWやIF-IDFは、トークンの出現頻度に基づいて文章データをベクトルデータに変換しました。同じように、トークン単位でベクトル化する手法を考えます。各単語をベクトル化して構造データとして扱うことで、より詳細に各単語や文章の意味を捉えることが期待できます。

単語の埋め込み

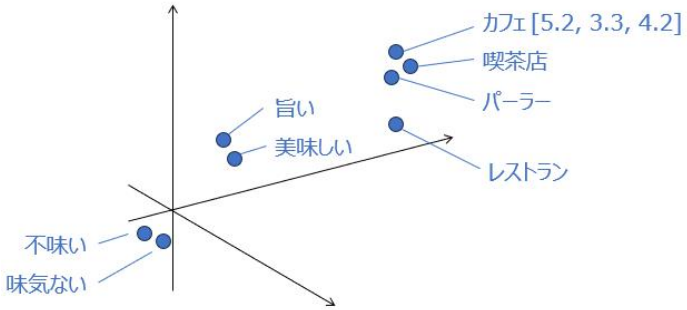
例えば、「カフェ」「喫茶店」「パーラー」といったトークンがあった時、これらは実質的に同じ意味を表していると考えられますので、同じような値を持つベクトルに変換しておいた方が、丸っきり別のデータとして扱うよりも都合がよさそうです。図X-Xのように、ベクトル表現が無い状態では、文章から抽出した各トークンに識別番号（ID）を割り当てる事しかできません[37]。



図X-X トークンをベクトルで表現するイメージ

[37] 図中の"304"と"242"は、それぞれ「カフェ」と「喫茶店」を一意に識別するために便宜的に割り振っただけで、数字自体には意味がありません。機械学習モデルは数字表現であれば入力として受け取ることができるので、"304"や"242"という値を入力する事が出来ますが、値自体も意味を持たせることによって、より詳細な分析が出来そうです。

ここで、各トークンの意味を表現する3次元ベクトルを獲得したと考えましょう（具体的なベクトル化の考え方は後述します）。各トークンのベクトル表現は3つの成分（3つの値）で構成されているので、図X-Xのように3次元グラフ上に描画できます。



図X-X トークンのベクトル表現を3次元空間上に描画

図X-Xを見ると、ベクトル成分が近い値を持つ「カフェ」「喫茶店」「パーラー」は同じような場所に点が存在します。仮に「レストラン」というトークンがあったとすれば、これら3者と同様に飲食店の一つの形態ですので、さほど遠くはない位置に現れる事でしょう。

対して、「旨い」「美味しい」といったポジティブな表現や、「不味い」「味気ない」といったネガティブな表現は、先ほどの飲食店を表すトークン群とは毛色が違う[38]ので、3次元空間上でも別の場所に出現します。ポジティブな表現とネガティブな表現とが混在していますが、これらのトークンの持つ意味合いは逆ですので、3次元空間上でも反対側（ないし遠くの位置）に出現します。ベクトル表現を用いて各トークンの意味合いを数字化する事で、意味的な近さや意味付けの方向性を定量的に評価できるため、少なくともトークンIDの状態よりも多くの情報を持たせることが可能となります。 このように単語やフレーズを多次元空間上の表現に変換する事を「埋め込み」（embedding）と呼びます[39]。

[38] カフェなどは飲食店を示す一般名称ですが、旨いなどは主観的な評価を表す形容詞ですので、意味合いという面でも、品詞という面でも異なります。

単語を埋め込む典型的な埋め込み手法として、word2vec[40] があります。この手法は、単語の意味は周辺の要素（文脈）によって形成される、という分布仮説 (distributional hypothesis)[41] に基づいて、単語を特徴量空間に埋め込みます。分布仮説とは、ある文章内において興味の対象となる単語と、その単語近くにある語句とは、何らかの意味的関係性があるという仮説です。例えば「犬」と「猫」は異なる動物を指しますが、似たような文脈（例えば、「ペット」「飼う」「かわいい」といった表現が出現する文脈）で使われることが多いため、意味的に関連があると捉えます。一方、「犬」や「猫」と「自動車」は全く異なる文脈で使われることが多く、意味的な関連は遠いと言った具合です。

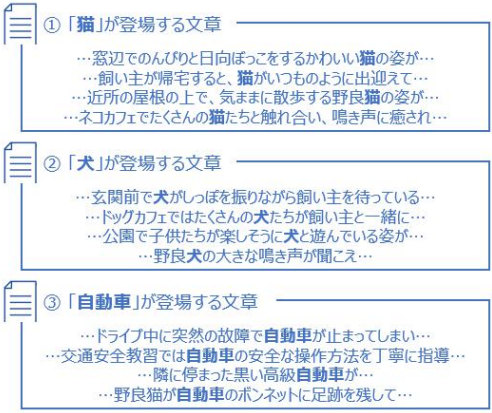


図 X-X 周辺から単語の意味を類推する分布仮説のイメージ

分布仮説のイメージを図 X-X のように示しました。人間であれば図中の太字で記した単語部分を隠しても周囲の文脈から単語を類推することができるか

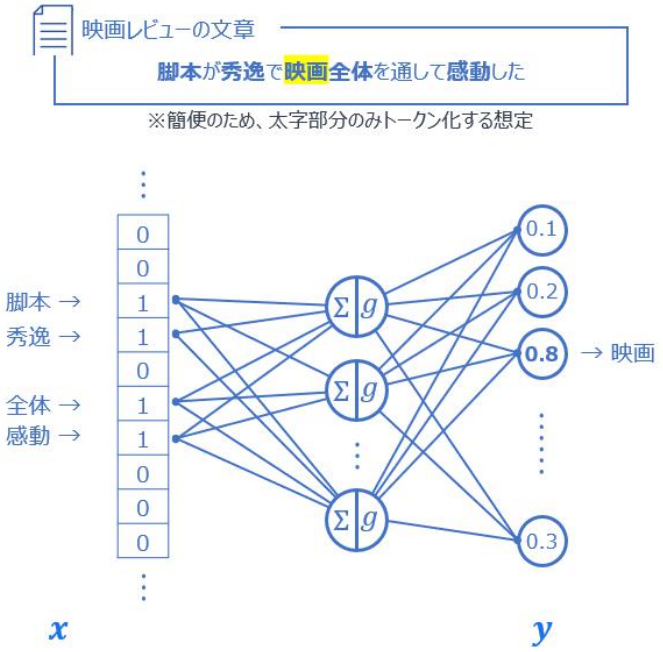
[39] 本書第１章でも登頂量空間に埋め込むイメージを紹介したかと思いますが、本節では具体例として単語の特徴量空間への埋め込みが取り上げられていると考えてください。
[40] 発音すれば察しが付くかもしれませんが、word2vec は"word to vector"の意です。このように単語を数値で置換する数略語（ヌメロニム; numeronym）はプログラム内の関数やソフトウェア名称としても用いられる略記法です。ヌメロニムで置換された数字は、省略された単語の文字数を示している場合や、同等の発音を有する数字を表す場合があります（"to"は２文字ですし、発音的にも"two"と似ています）。仮想環境の管理を行う Kubernetes（クバネティス）という有名なソフトウェアがあるのですが、中の８文字分を省略して"K8s"と短縮表記されることがあります。
[41] 分布意味論（distributional semantics）とも呼ばれます。1950 年代に言語学者の John R. Firth 氏が広めた"You shall know a word by the company it keeps."（単語は、その周辺にある文脈から知ることができる）という考えに基づいています。

と思います（文章①②つについては、犬や猫以外の単語が入るかもしれませんが、「かわいい」「鳴き声」といった表現が周辺に含まれるため、少なくとも動物である事は類推できるかと思います。同様に、文章③についても、自動車ではなくとも周辺の表現から少なくとも乗り物である事は類推できるかと思います）。人間にとっては文脈の意味合いを理解して単語を推測できますが、AIの世界では、このような人間の感覚を数字表現に変換する必要があります。本節では、この意味的な関係性を定量化する手法であるCBOWモデルとskip-gramを紹介します。

CBOW（continuous bag of words）

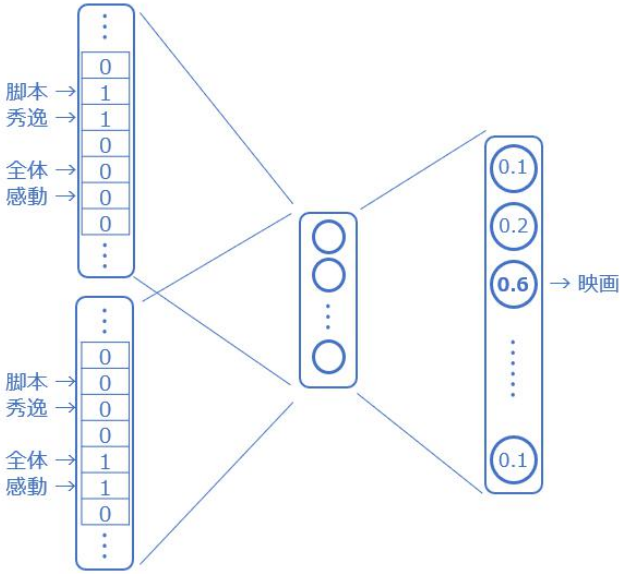
CBOWは興味がある単語の近傍に存在する複数の要素（文脈）から、その単語を予測するように学習する手法です。図X-Xのように、例えば「脚本が秀逸で映画全体を通して感動した」という文章を考えます。今回は簡便のために図中で太字で示した単語のみを考慮するとして、「映画」という単語に注目します。この「映画」の周囲2単語[42]から、「映画」という単語を推論できるようモデルを学習します（学習時には「映画」を正解として、モデルが出力した確率の交差エントロピー誤差が最小となるようにネットワークの重みを調整します）。つまり、CBOWとは周辺語（コンテキスト）から中心語（ターゲット）を推論する、中間層が1層のニューラルネットワークと説明できます。なお、今回は「映画」を中心語として例示しましたが、実際には長い文章中にある単語の数だけ組合せを考慮する事になります。

[42] このように、注目した単語からどの程度離れた単語までを考慮するかという指標をウィンドウサイズと呼びます。今回はウィンドウサイズを2として、注目単語の前後にある2単語ずつ（計4単語）を周辺語（コンテキスト）として取り出します。



図X-X CBOWのモデルイメージ

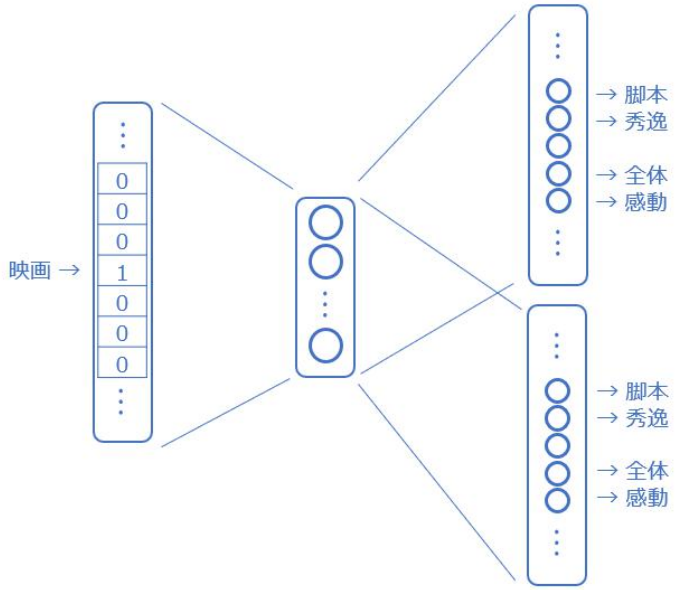
CBOWの模式図は、図X-Xのように簡略化されることもあります。CBOWは注目単語の前後にある周辺語を入力として用いますので、前に出現する周辺語と、注目単語の後に出現する周辺語とで入力が2つに分かれているような描き方となっていますが、先ほどの図X-Xと同等です。



図X-X CBOWの簡略図

skip-gram

skip-gramはCBOWとは逆のアプローチとなっており、中心語から周辺語を予測する手法で図X-Xのように図示できます。



図X-X skip-gramの簡略図

CBOWとskip-gramではskip-gramモデルの方が良い結果が得られるとされており、コーパスが大規模になるにつれて程頻出の単語や類推問題の性能の点において優れた結果が得られる場合があります。 一方、skip-gramは文章の数だけ損失を求める必要があるため学習コストが大きく、CBOWの方が学習は高速です。