短単位検索システムと OpenCHJ 形式形態論情報付与ツールの構築

松本 茂雄（慶応義塾大学文学部）

Development of a Short Unit Word Search System and a Morphological Annotation Tool for the OpenCHJ Format

Shigeo Matsumoto (Keio University Faculty of Letters)

要旨

テキストデータを短単位で検索できる Web アプリケーション『短単位検索システム』と、それに対応する形態論情報付与ツール『OpenCHJAnnotator』を開発した。従来、短単位での検索には国立国語研究所の『中納言』が主に用いられてきたが、検索対象は収録されたコーパスに限定される。そこで、『OpenCHJAnnotator』により手元のテキストを OpenCHJ 形式に自動変換し、『短単位検索システム』に読み込ませることで、中納言に未収録のテキストデータについても短単位検索を実行できるようにした。両ツールの連携により、データ変換から検索までを一貫して実行できるとともに、短単位での検索結果を検証可能な形で報告することが可能となる。また、両ツールの構築に際しては生成 AI モデルが活用されており、言語資源分野における AI 技術活用の実践例ともなっている。

Abstract

A web application, the "Short Unit Word Search System", that enables the searching of text data by short unit words, and a corresponding morphological annotation tool, "OpenCHJAnnotator", have been developed. Previously, searches based on short unit words primarily relied on "Chunagon", provided by the National Institute for Japanese Language and Linguistics. However, its use has been limited to specific corpora, making it difficult to utilize users' own text data. To address this limitation, "OpenCHJAnnotator" was implemented to automatically convert users' own text data into the OpenCHJ format, enabling the "Short Unit Word Search System" to process text data not included in "Chunagon". The integration of both tools enables seamless execution from data conversion to searching, while also providing verifiable reports of short unit word search results. Furthermore, generative AI models were utilized in the development of both tools, presenting a practical example of AI technology application in the field of language resources.

１．はじめに

テキストデータを UniDic に基づく短単位で検索できる Web アプリケーション『短単位検索システム』と、同システムで利用可能な OpenCHJ 形式のデータに変換するための『OpenCHJAnnotator』を開発した。開発の目的は、手元にあるテキストデータの短単位検索を実行できるようにするとともに、検索結果の再現性を担保することにある。従来、国立国語研究所の『中納言』を除けば、短単位での解析を行うことは困難であり、結果の再現性が保証されるとは限らなかった。こうした背景のもと、本研究では、2 つのツールを連携させて利用することで、短単位検索の実行と、その検索結果を再現するための仕組みを構築した。以下、その詳細について、設計と実装を中心に述べる。

２．既存ツールと開発の背景

２.１　『中納言』

短単位に基づく検索を実行できるツールとして、これまで主として用いられてきたのは、国立国語研究所のコーパス検索 Web アプリケーション『中納言』である。複数の検索方法のうち短単位検索では、形態論情報（語彙素、書字形、品詞など）に基づく詳細な検索が可能である。短単位とは、現代語において意味を持つ最小の単位を、短単位認定基準に基づいて結合させ（もしくは結合させずに）認定した単位である (小椋 2006)。コーパスを短単位で解析するときには、MeCab[1]などの解析器とともに、解析用辞書 UniDic が用いられる (伝ほか 2007)。UniDic を用いることで、日本語のテキストを短単位に分かち書きしたうえでの自動解析が可能となる (岡 2019)。『中納言』では、UniDic であらかじめ短単位自動解析され、一部について人手で形態論情報が付与されたコーパスが収録されている。このように、短単位検索を可能とするテキストデータを用意する際は、MeCab と UniDic を組み合わせた解析による形態論情報の付与が、現時点における有効な方法の 1 つとなっている。

２.２　形態素解析ツール

２.２.１　『Web 茶まめ』

MeCab と UniDic を組み合わせて自動解析を実行できる既存ツールに『Web 茶まめ』がある (堤・小木曽 2023)。このツールでは、Web ブラウザ上にテキストを直接入力したり、テキストファイルをアップロードすることで、テキストを解析できる。UniDic は、現代語用 UniDic と古文用 UniDic に大別されるが、画面上に各種の UniDic が一覧で表示されており、テキストの時代やジャンルに適した辞書を選択することが可能である。出力項目には、語彙素、書字形、品詞などが含まれる。出力形式は、HTML 形式でのブラウザ表示や CSV 形式などでのダウンロードに対応している。UniDic で分かち書きされ、形態論情報が付与されるため、短単位での語の出現頻度を項目別に集計するといったことに応用できる。

２.２.２　『TagAnt』

オペレーティングシステム (OS) 上で形態素解析を実行できるグラフィカルユーザインタフェース (GUI) ツールに『TagAnt』がある (Anthony 2024)。テキストの直接入力とファイルのアップロードに対応しており、解析結果は GUI 上に即座に表示される。現行のバージョン (Version 2.1.1) は spaCy[2] ベースで多言語に対応しており、日本語選択時のトークナイザには SudachiPy[3] (解析用辞書 sudachidict_core) が用いられる。デフォルトの分割モード A では短単位相当での分かち書きが可能である。解析時に、品詞 (part-of-speech; POS) や原形 (lemma) などを付与することができ、結果はテキストファイルで出力される。ただし、UniDic を解析に用いる機能は備えていない。使用言語と出力形式を設定するだけで解析処理を実行でき、解析結果を即座に視認できることから、ユーザの学習コストが低く作業効率の高いツールであると評価できる。

２.３　開発の背景

テキストを定量的に分析した結果を検証可能な形で提供するためには、テキストデータと、その分析に用いたシステムが公開されていることが望ましい。郡司 (1997) は、言語学において、「同じデータを用いれば誰でも同じ結果が出るような研究方法をとっていることが要求される」と述べ、第三者が追試できることの重要性を指摘している。これを実現する手段の 1 つとして、『中納言』の検索結果を、検索条件式とともに報告する方法がある。検索条件式を用いれば、常に同じ検索結果が得られるからである。一方、『中納言』のコーパ

---

[1] https://taku910.github.io/mecab/

[2] https://spacy.io/

[3] https://github.com/WorksApplications/SudachiPy

スに含まれていないテキストを対象とする場合は、調査方法が研究者ごとに異なるため、検索結果の再現性が保証されないという課題がある。もし、独自のテキストに対しても、短単位のような斉一な単位で検索を行い、検索条件式のような結果を再現できる機能を持つシステムがあれば、『中納言』へのコーパス収録を待たずとも、『中納言』と同様に、検証可能な形で検索結果を報告できる。ここに『短単位検索システム』を開発する意義が認められる。

『中納言』のコーパスの 1 つである「オープン CHJ」が公開され、その形態論情報の形式が規定されたことも、本システムの開発を後押しする要因となった。このコーパスは、オープンライセンスのテキストデータに基づいて構築され、形態論情報もオープンデータとして公開されたものである。そのデータ形式を、ここでは OpenCHJ 形式と呼ぶ。OpenCHJ 形式は、UTF-8 (BOM なし) 、LF 改行、タブ区切りで、13 フィールドを持つデータである[4]。タブ区切りであるため、XML 形式で求められるようなタグによるマークアップを省略できる。それによりデータ量も抑えられ、計量的な分析に適した形式となっている。また、「オープン CHJ」で採用された形式であることから、今後、汎用性の高い形式として広く利用されることが期待される。したがって、『短単位検索システム』に対応するデータとして OpenCHJ 形式を採用することとしたが、同形式へ直接変換する既存のツールは見当たらなかった。そこで、その目的を実現する新たなツールとして、『OpenCHJAnnotator』を開発することとした。

３．設計と実装

３.１　システムの全体像

『短単位検索システム』と『OpenCHJAnnotator』は、連携させて利用することを想定している。連携時の処理の全体像を示すと図 1 の通りである。

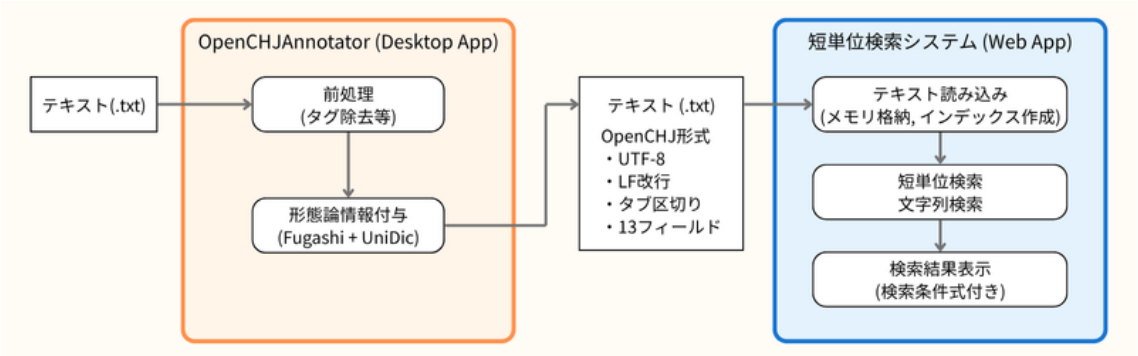

図 1　『OpenCHJAnnotator』と『短単位検索システム』の連携

手元のテキストを『OpenCHJAnnotator』によって OpenCHJ 形式に変換する。そのテキストをそのまま『短単位検索システム』に読み込ませることで、短単位による検索結果や検索条件式の表示が可能となる。

開発期間は 2025 年 3 月から 2025 年 6 月である。開発プロセスにおいては、後述する通り生成 AI モデルを活用した。以下、両ツールごとに設計と実装を記述する。

３.２　『短単位検索システム』の設計と実装

３.２.１　システム構成

『短単位検索システム』は Web ブラウザ上で動作する Web アプリケーションである。『中

[4] https://github.com/togiso/OpenCHJ/

納言』を参考に設計を行っているが、ソースコードはすべて独自に作成し、主として JavaScript により記述した。検索方法として短単位検索と文字列検索のいずれかを選択でき、検索対象としてユーザ自作のテキスト (OpenCHJ 形式) の読み込みに対応する。

テキストに含まれ得る個人情報や著作権への配慮から、本システムは、テキストの読み込みや検索処理を、すべてクライアントサイドの Web ブラウザ上で実行するように設計した。読み込まれたテキストデータは、Web ブラウザのメモリに一時的に保存されるのみで、SQL クエリや API 連携などを通じた外部への送信を前提としない。一方、読み込み可能なデータ量は、Web ブラウザを実行するコンピュータのメモリ容量に依存するため、ファイル読み込み時に合計 60MB の容量制限を設けた[5]。本システムと『中納言』との主な違いを表 1 に示す。

表 1　『短単位検索システム』と『中納言』の比較

	『短単位検索システム』	『中納言』(BCCWJ の場合)
データ処理方式	クライアントサイド処理	サーバサイド処理
データベース	ブラウザ内メモリ	SQL Server
コーパス規模	約 50 万語 (上限 60MB)	1 億語超
検索方法	短単位、文字列	短単位、長単位、文字列、位置
検索条件指定	検索フォーム、検索条件式	検索フォーム、検索条件式、履歴
自作データの検索	可 (OpenCHJ 形式)	不可
検索結果表示件数	全件表示	最大 500 件まで (ランダム表示)
検索結果ダウンロード	不可 (未実装)	可
ユーザ登録	不要	要

※『現代日本語書き言葉均衡コーパス (BCCWJ) 』の仕様は小木曽・中村 (2011) に基づく。

３.２.２　検索機能

短単位検索では、検索フォームへの文字列入力またはドロップダウンリストの選択により検索条件を指定する。文字列入力は「書字形出現形」「語彙素」の検索で用いる。ドロップダウンリストは「品詞」「活用型」「活用形」の検索で用い、『中納言』の大分類[6]に相当する項目がリスト表示される。キーとなる条件に関して、前方もしくは後方共起条件を追加でき、そのうえで各条件に短単位条件を追加できる。検索を実行すると、検索結果とともに、その検索結果を再現するための検索条件式が表示される。また、この検索条件式は『中納言』と同様の形式であり、『中納言』でもそのまま利用できる。

文字列検索では、短単位に限らず 30 字までの文字列を検索語として指定する。検索オプションとして検索語にワイルドカードまたは正規表現を用いることができる。

当初の検索処理の実装では、JavaScript がシングルスレッドで実行されることから、読み込んだデータ量次第では検索処理と描画処理が競合し、描画処理が遅延する現象が見られた。そこで、検索インデックスの作成と、Web Worker[7]による並行処理を導入した。テキストデータ読み込み時に形態論情報からトークン (短単位) ごとのインデックス (書字形出

[5] この制限下においても、OpenCHJ 形式の公開データセット『源氏物語 形態論情報データ (OpenCHJ)』の全ファイル (約 50 万語) を読み込むことが可能である。

[6] 中納言では品詞などの下位分類として大分類・中分類・小分類が設けられている。本システムではユーザ操作上の簡便性を考慮して、大分類に相当する項目のみを採用した。

[7] Web Worker は、Web ブラウザが提供する Web API の 1 つで、JavaScript のメインスレッドとは独立した別スレッドでの処理を実行する。

現形、語彙素、品詞など) を作成し、Web Worker の検索処理をメインスレッドから分離してバックグラウンドスレッドで並行的に実行する。これにより描画処理の遅延が解消された。

検索速度を評価するため、『源氏物語 形態論情報データ (OpenCHJ)』の約 50 万語を対象に、Web Worker が担う検索処理の実行時間を Chrome DevTools を用いて測定した。ただし、検索結果の描画処理に係る実行時間は除いた。表 2 に示す各検索条件について、それぞれ 10 回の測定を行い、平均実行時間 (ミリ秒) と標準偏差 (SD) を算出した。

表 2　検索処理の評価結果

検索方法	検索条件	検索結果 (件)	平均実行時間 (ミリ秒)
短単位検索	書字形出現形「時」	237	114.33 (SD 2.64)
	語彙素「為る」	3,523	115.25 (SD 1.62)
	品詞「動詞」と共起条件	10,581	147.29 (SD 5.51)
文字列検索	文字列「のたまへば」	239	150.23 (SD 3.16)

※「動詞」の共起条件は、前方 1 語「助詞」、後方 1 語「助動詞」とした。

※実行環境: Windows 11 Pro (24H2), Intel Core i7-10700F CPU (8 コア, 2.90GHz), 32.0GB RAM, Google Chrome 138.0.7204.158 (Official Build)

実行環境にもよるが、検索対象が 50 万語程度であれば、検索処理が概ね 0.2 秒以内で完了することを確認した。上記の実装に加え、クライアントサイド処理の利点として、外部サーバの応答時間やインターネット回線の速度に依存しない点も、高速な検索を実現する要素といえる。なお、本アプリケーションをダウンロードのうえ、ローカル環境で簡易 HTTP サーバを動作させれば、オフラインでの使用も可能である。

３.２.３　ユーザインタフェース

ユーザインタフェース (UI) は、『中納言』を参考に設計した。本システムの検索動作が『中納言』に近い仕様となっていることから、『中納言』の UI と整合している方がユーザビリティを確保できると考えたためである。

図 2　『短単位検索システム』の検索画面

短単位検索での共起条件の追加 (図 2)、KWIC 形式での検索結果表示 (図 3)、前後文脈における形態論情報のツールチップ表示、モーダルウィンドウでの詳細表示などがその例である。

検索条件

検索結果11件 (1-11件を表示)
検索対象語数：108,178　記号・補助記号・空白を除いた検索対象語数：98,063　検索対象サンプル数：1　(1ファイルで該当)

ファイル名	開始文字位置	終了文字位置	前文脈	キー	後文脈	語彙素	品詞	活用型	活用形	書字形出現形	発音形	語種	成立年	形式	作者	性別	作品名	サブコーパス名
kokoro	48570	48590	いう|警告|を|与え|た|の|で|ある|。|他|の|懐かしみ|に|応じ|ない|(先生)|は|、|他|を	軽蔑	(する)|前|に|、|まず|自分|を|軽蔑|し|て|い|た|もの|と|みえる|。|私|は|無論|先生	軽蔑	名詞-普通名詞-サ変可能			軽蔑	ケーベツ	漢	1914	小説	夏目漱石	男	こころ	青空文庫
kokoro	70290	70310	」|先生|は|その|日|これ|以外|を|語ら|なかっ|た|。|六|私|は|それ|から|時々|(先生)|を	訪問	(する)|よう|に|なっ|た|。|行く|たび|に|先生|は|在宅|で|あっ|た|。|先生|に|会う|度数	訪問	名詞-普通名詞-サ変可能			訪問	ホーモン	漢	1914	小説	夏目漱石	男	こころ	青空文庫
kokoro	84860	84880	た|事|が|ない|の|です|」|七|私|は|不思議|に|思っ|た|。|しかし|私|は|(先生)|を	研究	(する)|気|で|その|宅|へ|出入り|を|する|の|で|は|なかっ|た|。|私|は|ただ|その|まま	研究	名詞-普通名詞-サ変可能			研究	ケンキュー	漢	1914	小説	夏目漱石	男	こころ	青空文庫
kokoro	136020	136040	奥|さん|と|二人|差向い|で|話|を|する|機会|に|出合っ|た|。|(先生)|は|その|日|横浜|を	出帆	(する)|汽船|に|乗っ|て|外国|へ|行く|べき|友人|を|新橋|へ|送り|に|行っ|て|留守|で|あっ	出帆	名詞-普通名詞-サ変可能			出帆	シュッパン	漢	1914	小説	夏目漱石	男	こころ	青空文庫
kokoro	158990	159010	。|先生|は|それ|を|奥|さん|に|隠し|て|死ん|だ|。|(先生)|は|奥|さん|の|幸福|を	破壊	(する)|前|に|、|まず|自分|の|生命|を|破壊|し|て|しまっ|た|。|私|は|今|この|悲劇	破壊	名詞-普通名詞-サ変可能			破壊	ハカイ	漢	1914	小説	夏目漱石	男	こころ	青空文庫

図 3　『短単位検索システム』の検索結果画面

一方で、独自に 4 つの機能を実装した。1 つめは、検索フォームにおいて短単位条件を追加する場合に、AND 条件以外に、OR 条件または NOT 条件を選択できる機能である。『中納言』では、検索フォームからは AND 条件のみ指定可能で、OR 条件や NOT 条件を指定するには検索条件式を利用するほかなかった。一般に論理演算子は NOT、AND、OR の優先順位で適用されるところ、『短単位検索システム』でもこの順位に従うように設計している。ただし、本システムの検索条件式は、演算子の出現順序が UI 上での追加順のままであり、優先する演算子を括弧で括る機能を実装していない。そのため、条件式における演算子の出現順序や、演算子の複数回の出現などにより、検索動作が異なる場合がある。

2 つめは、検索条件式から UI 上の検索フォームを再現する機能である。検索条件取込ボタンをクリックし、ユーザが検索条件式を貼り付けたりすることで、その式を出力する検索フォームが再現される。通常は検索条件式よりも検索フォームを利用するユーザが多いと想定され、この変換機能はニーズが高いと考えられる。ただし、検索条件式の括弧内にさらに括弧が含まれる入れ子構造の式には対応していない。

3 つめは、検索結果を全件表示する機能である。『中納言』では、検索結果の表示は 500 件までに制限され、検索するたびにランダムに表示内容が変わる仕様となっている。本システムでは、読み込むデータ量の制限から全件を表示することに支障がなく、ユーザの自作データの利用を前提としているため、この機能を設けてユーザの利便性を高めるようにした。

4 つめは、前後文脈におけるハイライト表示機能である。『中納言』では前方・後方共起条件に合致した語が括弧で括られて表示される。この表示は、ユーザによっては必ずしも視認性が高いとはいえず、本システムでは括弧で括ったうえで黄色の背景色を追加して目立たせるようにした。ただし、この機能については、ユーザがハイライトの表示方法を選択できるようにすべきかもしれない。

３.２.４　『中納言』との比較による検索結果の検証

『短単位検索システム』は OpenCHJ 形式に対応するように設計しているので、同じデー

タを対象とすれば、『中納言』のコーパス「オープン CHJ」の検索結果と一致することが期待される。これを検証するため、『短単位検索システム』と『中納言』で同じデータを用いるようにして[8]、以下の (1) から (6) までの 6 つの検索条件式について、それぞれの検索結果件数と、形態論情報の内容が一致するかどうかを調査した。以下にその結果を示す。

(1)
キー: 語彙素="有る"

この基本的な検索条件式では、『短単位検索システム』222 件、『中納言』222 件で検索結果件数が一致した。形態論情報の内容に関して、『短単位検索システム』の検索結果画面を図 4 に、『中納言』の検索結果画面を図 5 に示す。なお、比較のために列の表示を調整し、開始文字位置で昇順にソートしたうえで抜粋してある。

検索結果222件 (1-20件を表示)
検索対象語数：26,311　記号・補助記号・空白を除いた検索対象語数：22,458　検索対象サンプル数：6　(6ファイルで該当)

開始文字位置	前文脈	キー	後文脈	語彙素	品詞	活用型	活用形	発音形	語種
180	羅生|門|芥川|龍之介|ある|日|の|暮方|の|事|で	ある	。|一人|の|下人|が|、|羅生|門|の|下|で|雨やみ|を|待っ|て|い|た|。|広い|門	有る	動詞-非自立可能	五段-ラ行	終止形-一般	アル	和
250	高瀬|舟|森|鴎外|高瀬|舟|は|京都|の|高瀬|川|を|上下|する|小舟|で	ある	。|徳川|時代|に|京都|の|罪人|が|遠島|を|申し渡さ|れる|と|、|本人|の|親類|が|牢屋敷|へ	有る	動詞-非自立可能	五段-ラ行	終止形-一般	アル	和
680	と|決意|し|た|。|メロス|に|は|政治|が|わから|ぬ|。|メロス|は|、|村|の|牧人|で	ある	。|笛|を|吹き|、|羊|と|遊ん|で|暮し|て|来|た|。|けれど|も|邪悪|に|対し|て	有る	動詞-非自立可能	五段-ラ行	終止形-一般	アル	和
1060	、|大きな|円柱|に|、|蟋蟀|が|一|匹|とまっ|て|いる|。|羅生|門|が|、|朱雀|大路|に	ある	以上|は|、|この|男|の|ほか|に|も|、|雨やみ|を|する|市女|笠|や|揉|烏帽子|が|、	有る	動詞-非自立可能	五段-ラ行	連体形-一般	アル	和
1060	で|暮し|て|来|た|。|けれど|も|邪悪|に|対し|て|は|、|人|一|倍|に|敏感|で	あっ	た|。|きょう|未明|メロス|は|村|を|出発|し|、|野|を|越え|山|越え|、|十|里|はなれ	有る	動詞-非自立可能	五段-ラ行	連用形-促音便	アッ	和

図 4　『短単位検索システム』での語彙素「有る」の検索結果

222 件の検索結果が見つかりました。
検索対象語数：26,311　記号・補助記号・空白を除いた検索対象語数：22,458　検索対象サンプル数：6

サンプル ID	開始位置	前文脈	キー	後文脈	語彙素	品詞	活用型	活用形	発音形出現形	語種
60N羅生 1915_11001	180	羅生|門 # 芥川|龍之介 # ある|日|の|暮方|の|事|で|	ある	|。 # 一人|の|下人|が|、|羅生|門|の|下|で|雨やみ|を|待っ|て|い|た|。 # 広い|門	有る	動詞-非自立可能	五段-ラ行	終止形-一般	アル	和
60N高瀬 1916_11001	250	高瀬|舟 # 森|鴎外 # 高瀬|舟|は|京都|の|高瀬|川|を|上下|する|小舟|で|	ある	|。 # 徳川|時代|に|京都|の|罪人|が|遠島|を|申し渡さ|れる|と|、|本人|の|親類|が|牢屋敷|へ	有る	動詞-非自立可能	五段-ラ行	終止形-一般	アル	和
60N走れ 1940_11001	680	と|決意|し|た|。 # メロス|に|は|政治|が|わから|ぬ|。 # メロス|は|、|村|の|牧人|で|	ある	|。 # 笛|を|吹き|、|羊|と|遊ん|で|暮し|て|来|た|。 # けれど|も|邪悪|に|対し|て	有る	動詞-非自立可能	五段-ラ行	終止形-一般	アル	和
60N羅生 1915_11001	1060	、|大きな|円柱|に|、|蟋蟀|が|一|匹|とまっ|て|いる|。 # 羅生|門|が|、|朱雀|大路|に|	ある	|以上|は|、|この|男|の|ほか|に|も|、|雨やみ|を|する|市女|笠|や|揉|烏帽子|が|、	有る	動詞-非自立可能	五段-ラ行	連体形-一般	アル	和
60N走れ 1940_11001	1060	で|暮し|て|来|た|。 # けれど|も|邪悪|に|対し|て|は|、|人|一|倍|に|敏感|で|	あっ	|た|。 # きょう|未明|メロス|は|村|を|出発|し|、|野|を|越え|山|越え|、|十|里|はなれ	有る	動詞-非自立可能	五段-ラ行	連用形-促音便	アッ	和

図 5　『中納言』での語彙素「有る」の検索結果

上の両図で共通する列においては、前文脈と後文脈での区切り記号「|」「#」の表示を除き、

[8] 『短単位検索システム』に「OpenCHJ-Aozora 形態論情報データ」を読み込ませて対象データとし、『中納言』では「オープン CHJ」(中納言 2.7.2、データバージョン 2025.03) のうち「近代小説-青空文庫/総研大 2024　全ての部」を対象データとして選定した。前後文脈の語数は「20」、共起条件の範囲は「文境界をまたがない」とした。『短単位検索システム』では「検索条件取込」機能で検索条件式を取り込んだ。

形態論情報の内容（前文脈、キー、後文脈、語彙素、品詞、活用型、活用形、発音形、語種）は完全に一致した[9]。

(2)
キー: 品詞 LIKE "動詞%"
AND 前方共起: 品詞 LIKE "接頭辞%" ON 3 WORDS FROM キー
AND 前方共起: 品詞 LIKE "名詞%" ON 2 WORDS FROM キー
AND 前方共起: 品詞 LIKE "助詞%" ON 1 WORDS FROM キー
AND 後方共起: 品詞 LIKE "助動詞%" ON 1 WORDS FROM キー
AND 後方共起: 品詞 LIKE "助動詞%" ON 2 WORDS FROM キー
AND 後方共起: 品詞 LIKE "助詞%" ON 3 WORDS FROM キー

これは、前方・後方共起条件を 3 件ずつ追加し、7 つの条件を用いて共起語を検索する条件式である。『短単位検索システム』2 件、『中納言』2 件であり、検索結果件数は一致した。また、形態論情報も一致した。

(3)
キー: (品詞 LIKE "動詞%" AND NOT 活用形 LIKE "連用形%")
AND 前方共起: (品詞 LIKE "助詞%" OR 品詞 LIKE "副詞%") ON 1 WORDS FROM キー
AND 後方共起: (品詞 LIKE "助動詞%" AND 書字形出現形="られ") ON 1 WORDS FROM キー

これは、前方・後方共起条件を 1 件ずつ追加した 3 つの条件に、条件ごとに異なる論理演算子 (NOT, OR, AND) が 1 つ含まれる条件式である。結果は、『短単位検索システム』19 件、『中納言』19 件で件数は一致した。また、形態論情報も一致した。

(4)
キー: (語彙素="有る" AND 活用形 LIKE "連用形%" OR 書字形出現形="あり" AND NOT 書字形出現形="有っ")

これは、1 つの条件に 3 種の論理演算子が 1 つずつ含まれる条件式である。結果は、『短単位検索システム』76 件、『中納言』76 件で件数は一致した。また、形態論情報も一致した。

(5)
キー: (品詞 LIKE "感動詞%" AND NOT 語彙素="ああ" AND NOT 語彙素="さあ" AND NOT 語彙素="うん" AND NOT 語彙素="いや" AND NOT 語彙素="おい")

これは、1 つの条件に 1 種の論理演算子が 2 つ以上含まれる条件式である。この例では NOT 演算子 1 種が 5 つ含まれている。結果は、『短単位検索システム』43 件、『中納言』43

[9] 『短単位検索システム』では、文境界は認識されるものの、文脈中の区切り記号の表示は未実装である。また、語彙素読みによる検索機能は省略した。

件で件数は一致した。また、形態論情報も一致した。

(6)
キー: (語彙素="有る" AND 活用形 LIKE "連用形%" AND NOT 書字形出現形="有っ" AND NOT 書字形出現形="あっ" OR 書字形出現形="あり" OR 書字形出現形="ある")

これは、1 つの条件に 3 種の論理演算子が含まれ、同種で 2 つ以上含まれるものがある条件式である。結果は、『短単位検索システム』163 件、『中納言』212 件であり、件数に不一致が見られた。差異の 49 件は NOT 条件で除外されるべき件数に相当する。したがって、何らかの理由により、『中納言』で NOT 条件が適用されなかったものと推測される。その原因は明らかではないが、この検証の範囲内では、検索条件式内に NOT 演算子とそれ以外の論理演算子が存在し、かつ NOT 演算子が 2 つ以上含まれる場合に、NOT 条件が適用されない場合があるものと考えられる。本システムの検索条件式が、NOT、AND、OR の優先順位を括弧で括って明示していないことも、不一致の要因の 1 つかもしれない。

以上の検証において、キー条件や、前方・後方共起条件といった基本的な検索条件において、『短単位検索システム』と『中納言』で同じ検索結果が得られることが確認された。また、それらの条件に論理演算子が追加された条件では、演算子が 2 つ以上含まれていても 1 種のみの場合、もしくは 2 種以上の演算子が含まれていても各演算子が 1 つのみの場合は、両者の検索結果は同じであった。ただし、(6) の結果の通り、検索条件式において NOT 演算子とそれ以外の演算子が含まれ、かつ NOT 演算子が 2 つ以上含まれる場合に、両者で結果に不一致が見られた。

３.３『OpenCHJAnnotator』の設計と実装

３.３.１　システム構成

『OpenCHJAnnotator』は、テキストに形態論情報を自動付与し、OpenCHJ 形式で出力するための GUI アプリケーションである。OS 上で動作し、インストールを要しない。短単位での解析を行うためには MeCab と UniDic を組み合わせることが有効である。ところが、MeCab は通常コマンドラインインタフェースから実行するので、不慣れなユーザにとっては学習コストが高くなる。また、サーバを運用しない想定のため、大容量の UniDic はローカル環境へのダウンロードが前提となる。したがって、ユーザが操作しやすい GUI を備え、あらかじめ UniDic を組み込んだデスクトップアプリケーションとして設計した。また、設計段階において、『Web 茶まめ』の持つ UniDic を用いた形態素解析機能と、『TagAnt』の持つ出力結果の GUI 表示機能を統合するようなツールとすることを要件とした。

MeCab を GUI で用いるために、形態素解析器として Fugashi を選定した (McCann 2020)。Fugashi は MeCab を内包する Python のライブラリであり、基本的に MeCab をインストール不要で利用できる。そのため、アプリケーションのソースコードを主に Python で記述した。また、解析用辞書として用いる UniDic に関しては、容量を抑えるため UniDic Lite[10]をデフォルト辞書に選定した。その他の各種 UniDic を利用する場合は、ユーザ自身が設定できるように実装した。

[10] https://github.com/polm/unidic-lite

３.３.２　テキスト入力

OS 上のテキストファイル (.txt) を読み込ませて入力テキストとする。読み込んだデータ量が大きい場合に動作が不安定となる現象が確認されたため、一度に読み込み可能なファイルサイズの合計を 10MB まで (ファイル数は 50 まで) とした。ファイルの入力時に文字コードを自動判定し、Unicode へデコードを行ったうえで内部処理を行い、出力時に UTF-8 (BOM なし) へエンコードを行うように実装している。

３.３.３　テキスト前処理

GUI において、括弧・空白・改行コードの除去、正規表現を用いた文字列の置換などの前処理を任意で実行する。GUI の左側に入力テキストが表示され、右側に前処理済みのテキストが整形テキストとして表示される。このとき、前処理が適切に反映されているかどうかを即座に確認することができる (図 6)。

また、利用するユーザが少なくないとみられるのが「青空文庫」[11]からダウンロードしたテキストファイルである。青空文庫形式のファイルは作業マニュアルに従って作成されており、注記などの記法が共通している。そこで、「《 》」で囲まれたルビの除去、冒頭・末尾の注記情報の除去、JIS X 0213 (第 3・第 4 水準漢字) の文字コードの置換などを、設定により前処理できるようにした。

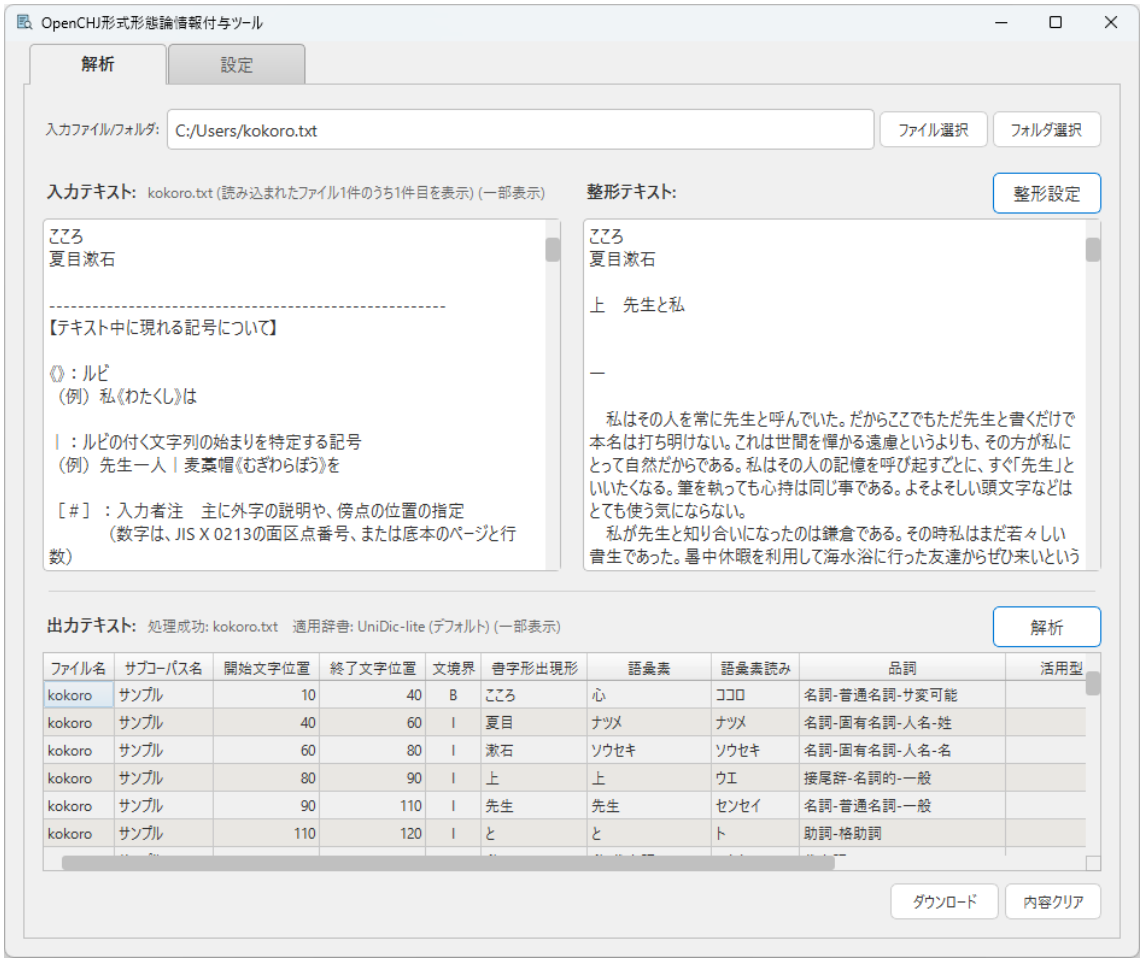

図 6　『OpenCHJAnnotator』のメイン画面

[11] https://www.aozora.gr.jp/

さらに、「タグ特別設定」という設定項目を設けた。この設定は、入力テキストにおいて、「[F あの]」のように括弧内に英字のタグが付いた文字列に対して、出力テキストでの品詞を指定するものである。前処理の段階で、人手によるアノテーションを一部付与する処理といえる。例えば、「あの」「その」のような文字列が「フィラー」であるのか、それとも「連体詞」であるのか、UniDic では適切に判別できないことがある。そうした特定の文字列の品詞をあらかじめユーザが指定したい場合に、表 3 に示すようなタグをテキスト内に挿入することで、UniDic の処理に優先して品詞を付与できるようにした。

表 3　タグ特別設定で「あの」をフィラーに設定する例

テキスト内のタグ	設定方法	設定例
[F あの]	タグの括弧を選択	[ ]
	タグの英字を入力	F
	タグの文字列を入力	あの
	設定したい品詞を選択	感動詞-フィラー

※品詞は、感動詞-フィラー、名詞-固有名詞、連体詞の 3 つから選択できる

出力テキストでは括弧と英字は削除され、内側の文字列が書字形出現形となる。

３.３.４　解析処理

形態素解析では、デフォルトで UniDic Lite が用いられる。その他の UniDic 辞書を適用する場合は、ユーザ自身が国立国語研究所の UniDic のウェブサイトより任意の辞書をダウンロードし、設定画面からカスタム辞書として設定する。また、未知語などを登録したい場合は、ユーザ辞書の設定にも対応している。

解析処理においては、テキストがトークン (短単位) に分かち書きされ、各トークンに OpenCHJ 形式のフィールド情報が一括付与される (表 4)。

表 4　オープン CHJ の 13 フィールド

1. ファイル名 (作者 作品名)
2. サブコーパス名
3. 開始文字位置 (ファイル頭からのオフセット値*10)
4. 終了文字位置 (同上)
5. 文境界 (B=文頭)
6. 書字形出現形 (=キー、表層形)
7. 語彙素
8. 語彙素読み
9. 品詞
10. 活用型
11. 活用形
12. 発音形
13. 語種

フィールド 1 の「ファイル名」には、入力したテキストファイルのファイル名 (拡張子を

除く) が付与される[12]。フィールド 2 の「サブコーパス名」には、設定画面において任意で入力した文字列が反映される。フィールド 3 の「開始文字位置」と 4 の「終了文字位置」は次の規則により位置が決まる。まず、ファイル冒頭のトークンの「開始文字位置」を 10 とする。次に、各トークンの「開始文字位置」に対し、そのトークン文字数に 10 を乗じた値を加算して「終了文字位置」とする。そして、その位置を直後のトークンの「開始文字位置」とする規則である。フィールド 5 の「文境界」に対しては、次の規則に従って文頭を示す B もしくは文中を示す I を付与する。まず、ファイル冒頭のトークンに B を、それ以外のトークンに I を付与する。次に、原則として、句点「。」もしくは閉じ鉤括弧「」」を文末とみなし、その直後のトークンに B を付与する。例外として、「。」」の並びにおける「」」のみ I として扱う。この規則により文境界が B で区切られる。残りの 6 から 13 までのフィールドは UniDic によって自動付与される。なお、名詞などのように、品詞によってはフィールド 10 の「活用型」とフィールド 11 の「活用形」が存在しない場合がある。この場合、両フィールドは空フィールドとして処理されるようにしてある。

３.３.５　テキスト出力

解析処理によって形態論情報が付与された 13 フィールドの結果は、GUI 下側のエリアに、出力テキストのプレビューとして表示される。データ容量が大きい場合に備えて、プレビューには 1,000 行までの表示制限を設けている。この時点でダウンロードボタンを押すと、UTF-8 (BOM なし) 、LF 改行、タブ区切りのテキストファイル (.txt) がユーザ指定のディレクトリに出力される。図 7 は、UniDic Lite を用いて解析したテキストの例である。

```
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　10　30　B　これ 此れ コレ 代名詞　コレ 和
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　30　40　I　が　が　ガ　助詞-格助詞　ガ　和
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　40　60　I　実例 実例 ジツレイ　名詞-普通名詞-一般　ジツレー　漢
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　60　70　I　で　だ　ダ　助動詞　助動詞-ダ 連用形-一般　デ　和
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　70　90　I　あり 有る アル 動詞-非自立可能　五段-ラ行 連用形-一般　アリ 和
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　90　110　I　ます ます マス 助動詞　助動詞-マス　終止形-一般　マス 和
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　110　120　I　。　。　補助記号-句点　記号
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　120　140　B　ある 或る アル 連体詞　アル 和
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　140　160　I　特別 特別 トクベツ　形状詞-一般　トクベツ　漢
ファイル名　ｻﾌﾞｺｰﾊﾟｽ名　160　170　I　な　だ　ダ　助動詞　助動詞-ダ 連体形-一般　ナ　和
```

図 7　出力テキストの例

このようにして出力されたテキストファイルは、そのまま『短単位検索システム』で読み込むことができる。

４．開発における生成 AI の活用

『短単位検索システム』と『OpenCHJAnnotator』の開発プロセスでは、複数の生成 AI モデルをソースコードの生成において活用した。主に利用した生成 AI モデルは、Anthropic 社の Claude 3.7 Sonnet、Google 社の Gemini 2.5 Pro、OpenAI 社の ChatGPT-4o である。基幹部分の実装には Claude を、複雑なディレクトリ構成にかかわる場合は Gemini を、UI のデザイン関連では主に ChatGPT を用いた。この開発プロセスを踏まえての、生成 AI モデル活用

[12] 本来のオープン CHJ の「ファイル名」は、作者と作品名を半角スペースで区切る形式である。本ツールではユーザの簡便な利用を考慮して、両者を区別していない。

の利点と課題を以下に述べる。

まず、生成 AI を活用する利点として、プログラミング言語習得の学習コスト低減と、コードの自動生成による開発期間の短縮が挙げられる。筆者はプログラミング言語に長けているわけではないが、自然言語によるプロンプト次第で、実行可能なコードを出力することができた。ただし、1 回の指示で意図通りに動作するコードが出力されるとは限らず、複数回のプロンプトの調整と修正作業の反復を要した。大半の時間はデバッグとコードの修正に費やされた。それでも開発期間が大幅に削減されたことは疑いないといえる。この実践例は、専門的な情報工学の知識を持たない者でも、基本的なプログラミングの知識により様々なツールが開発できることを示している。

次に、生成 AI 活用の課題として、生成されたコードの著作権の問題が挙げられる。上記の生成 AI モデルを提供する各社は、モデルが生成したコンテンツの権利等を主張していないため、生成されたコードに著作権が発生する場合には、原則としてユーザ側に帰属することになる[13]。ただし、各社は第三者の権利侵害については責任を負わないとしている。すなわち、生成されたコンテンツには、著作権で保護された既存のコードが含まれる可能性があるため、生成されたコードが第三者の著作権を侵害していないことを確認する責任はユーザ側にあるとされる。この点、開発プロセスにおいては、全コードについて目視により確認を行い、プログラムの動作を確認したが、いずれのコードも新規に開発するツールのために生成し、また適合するように修正を重ねており、権利保護されたコードがそのまま含まれている可能性は低いと判断した。また、両ツールのソースコードを GitHub 上に公開して透明性を確保し、商用利用ではなく研究・教育目的で提供することを明記した。ライセンスは MIT-0 (MIT-No Attribution License)[14]とし、Fugashi や UniDic Lite などのサードパーティライブラリに属するライセンスも併記した。

５．今後の課題

両ツールに関しては、筆者が基本的な動作を確認しているものの、予期せぬ不具合が残されている可能性がある。そのため、今後はユーザからのフィードバックに基づき、バージョンアップによるユーザビリティの向上が不可欠といえる。以下、各ツールの今後の課題について述べる。

まず、『短単位検索システム』に関しては、大容量データへの対応や、検索結果のダウンロード機能の実装、入れ子構造の括弧を含む検索条件式への対応が、主要な課題として挙げられる。大容量データの読み込みに対応するには、データベースの構築およびサーバサイド処理への移行、あるいは OS インストール型のアプリケーションの提供といったアプローチが考えられる。検索結果のダウンロード機能については、出力すべきデータ項目や出力データ形式をどのように設計するかが要点となる。検索条件式における入れ子構造の括弧への対応では、優先する演算子を括弧で括るために UI 上の追加順を内部で並び替える処理が求められる。あるいは、1 つの条件には同種の演算子のみしか追加できない仕様とすることも

---

[13] 生成 AI モデルが生成したコンテンツの権利等に関する各社の利用規約。
Anthropic 社 https://www.anthropic.com/legal/consumer-terms/
Google 社 https://ai.google.dev/gemini-api/terms/
OpenAI 社 https://openai.com/policies/row-terms-of-use/

[14] 著作権表示やライセンス条文の表示義務が無いパブリックドメイン相当のライセンス。

考えられる。

『OpenCHJAnnotator』に関しては、未実装の機能がいくつか残されている。例えば、GUI上でのテキスト直接入力を可能にすること、特殊な文字を前処理できるようにすること、OpenCHJ 形式以外の出力形式に対応することなどである。また、タグ特別設定を行った場合の出力テキストは、単に UniDic で解析された場合のテキストと異なるため、その旨をメタデータに含めて出力することも検討を要する。

６．おわりに

以上、『短単位検索システム』と『OpenCHJAnnotator』の設計と実装について述べた。両ツールを連携させることで、ユーザの手元にあるテキストデータに対して短単位検索を実行し、検索結果を検証可能な形で提示できる仕組みを構築した。また、生成 AI モデルの活用により、プログラミング言語に関する高度な専門性がなくても、学習コストを抑えてツールを短期間で開発できる可能性があることを示した。

謝　辞

本研究は、『中納言』および『UniDic』という存在なくしては実現しませんでした。開発に携わった関係各位に深く感謝申し上げます。また、形態素解析器 MeCab および Fugashi の開発者の皆様にも厚く御礼申し上げます。OpenCHJ のファイル形式は、本システム開発の基盤となりました。開発プロセスにおいては、Anthropic 社、Google 社、OpenAI 社が提供する生成 AI モデルより多大な支援を賜りました。ここに感謝の意を表します。

文　献

岡照晃 (2019).「第 1 章 言語研究のための電子化辞書」伝康晴・荻野綱男編『講座日本語コーパス 7 コーパスと辞書』朝倉書店, pp.1-28.

小木曽智信・中村壮範 (2011).『『現代日本語書き言葉均衡コーパス』形態論情報データベースの設計と実装　改訂版』国立国語研究所内部報告書 (LR-CCG-10-06).

小椋秀樹 (2006).「第 3 章 形態論情報」国立国語研究所『日本語話し言葉コーパスの構築法』(国立国語研究所報告 124), pp.133-186.

郡司隆男 (1997).「第 4 章 言語科学の提唱」松本裕治ほか『岩波講座 言語の科学 1 言語の科学入門』岩波書店, pp.127-165.

堤智昭・小木曽智信 (2023).「複数の UniDic 辞書による形態素解析支援ツール『Web 茶まめ』の実装と運用」『情報処理学会論文誌』Vol.64, No.3, pp.749-757. http://doi.org/10.20729/00225271

伝康晴・小木曽智信・小椋秀樹・山田篤・峯松信明・内元清貴・小磯花絵 (2007).「コーパス日本語学のための言語資源―形態素解析用電子化辞書の開発とその応用―」『日本語科学』22, pp.101-123.

Laurence Anthony. (2024). TagAnt (Version 2.1.1) [Computer Software]. Tokyo, Japan: Waseda University. https://www.laurenceanthony.net/software/TagAnt.

Paul McCann. (2020). fugashi, a Tool for Tokenizing Japanese in Python. In Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS), pages 44–51, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.nlposs-1.7

関連 URL

『短単位検索システム』 https://github.com/smtmto/jp-suw-search
『OpenCHJAnnotator』 https://github.com/smtmto/openchj-annotator
『中納言』 https://chunagon.ninjal.ac.jp/
『UniDic』 https://clrd.ninjal.ac.jp/unidic/
『源氏物語 形態論情報データ (OpenCHJ)』 https://github.com/togiso/OpenCHJ-Genji
『OpenCHJ-Aozora 形態論情報データ』 https://github.com/togiso/OpenCHJ-Aozora
『Web 茶まめ』 https://chamame.ninjal.ac.jp
『TagAnt』 https://laurenceanthony.net/software/tagant/