ひめのり：辞書生成機能を備えたワードクラウド作成システム

澁谷 泰蔵 [1,2]　稲吉 希理 [1]　西田 健太郎 [1]　田中 朋 [1,2]
[1] 日本電気株式会社　[2] 産業技術総合研究所
{taizo, kiri.i, ken-nishida, tomotanaka}@nec.com

概要

専門用語辞書生成機能を持つワードクラウド作成システム「ひめのり」を開発した．システムはスマートフォン上でも動作可能な Web インターフェースを持ち，様々な分野の専門家による効率的な形態素解析辞書の生成を可能にする．1940 年代の応用物理学会誌のデータを用いた専門用語抽出実験から，システムによる作業効率の加速効果は手作業に比べ最大で 3 倍程度あることが分かった．今後ひめのりが得意なデータセットの特徴を明らかにするとともに，辞書以外の言語資源の整備機能も追加していく．

1　はじめに

膨大なテキストデータを概観する手法として，ワードクラウドが知られている．ワードクラウドは図 1 に示すように文書中に含まれる多数のトークン[1)]を並べた画像であり，重要なトークンほど大きく表示される．ワードクラウドの作成は，トークンリストの抽出とトークン重要度の決定という 2 つの作業から成り，素朴には形態素解析器による文書のトークン化とその頻度集計で実現される．高品質なワードクラウド生成には，トークンの分割単位や重要度に分析者の視点を反映させる必要がある．例えば「化合 物 半 導体」という短単位のトークンで構成された複合語をどこまで接続するかや，「情報」や「システム」といったトークンをワードクラウドに含めるかといった判断は分析者や分野に依存する．

我々は，複合語リストの抽出を分かち書きテキストにおけるトークン接続（チャンキング）タスクと捉え，MeCab [1] 用の複合語辞書生成機能を備えたワードクラウド作成システムを開発している．このシステムはスマートフォンでも操作可能なシンプルなインターフェースを特徴とする Web アプリケーションであり，トークン同士を接続する糊という意味を込めて「ひめのり」[2)]と名付けた．ひめのりはテキスト中に隣接して現れるトークンペアの接続判断をユーザに迫り，複合語辞書を効率的に生成する．また，頻出する低重要度トークンを非表示語（ストップワード）リストに追加するための画面も備えている．本稿では，ひめのりを紹介するとともに，そのトークン接続タスクの加速効果を実験で明らかにする．また，形態素解析辞書以外の言語資源生成ツールとしての可能性についても議論する．

図 1　ワードクラウドの例．2008-2021 年の『自然言語処理』掲載論文のタイトルから作成．

2　関連研究

複合語リストの自動抽出に関しては，頻度 [2] やルール [3] をベースにしたものから BERT を利用したもの [4] まで様々な研究があり，これらの手法に分析者の視点を反映させるには，テキストマイニング向けのツール [5] や，種々のスパンアノテーションツール [6–10] 等を利用できる．もちろん長単位の分割に対応した形態素解析器 [11, 12] や辞書 [13] も重要な選択肢である．トークン重要度の決定に関しても様々な手法 [2, 14] が提案されている．これらを統合したワードクラウド作成システムは筆者らの知る限りでは存在しないが，例えばユーザーローカル社のサービス [15] は複合語リストとストップワードリストの入力に対応し，トークン重要度は TF-IDF [16] で決定している．

1)　何らかの手段で文章を短い単位に分割したときの一つ一つの言葉のまとまりのこと．

2)　姫糊（ひめのり）は米を原料とした糊のこと．

3　システムの詳細

図 2　ひめのりの全体像.

ひめのりは Python と JavaScript で書かれた web アプリケーションであり，非同期処理を利用して軽快な動作を実現している．ひめのりの全体像を図 2 に示す．クライアント側には JavaScript ライブラリである React [17] を採用し，サーバー側は Python フレームワークである FastAPI [18] を利用している．テキストのトークン分割には MeCab を，データベースには SQLite [19] を，ワードクラウドの生成には Andreas Mueller 氏の Python ライブラリ WordCloud [20] と Google Noto Fonts [21] をそれぞれ採用している．辞書生成の基本的な流れは，分析するテキストのアップロード，MeCab によるトークン分割，トークンおよびトークンペアの出現頻度集計，トークンペアに対するユーザの判断で構成される．これらを組み合わせることでトークンペアのチャンキングを繰り返し，MeCab 向けのユーザ辞書を更新し，最終的に複合語辞書を得る．また，ユーザはストップワードリストを修正することでワードクラウドの見栄えを整えることができる．

ひめのりの機能の肝となるトークンペアに関する繰り返し処理（周回）というコンセプトについて図 3 を用いて説明する．最初の周回は，文章ごとに分割された入力テキストデータを MeCab によってトークンに分割し，単一トークンおよびトークンペアの出現頻度を集計することで始まる．この時，トークンの頻度情報からワードクラウドも作成する．次に，頻度順にソートされたトークンペアごとに，接続するかどうかの判断をユーザに促す．判断画面の例を図 4 に示す．画面には単一トークンとトークンペアの出現頻度，およびペアを含む文章一覧が表示され，ユーザはこれを参考にペアを辞書・仮辞書・非辞書のどれかに分類する．例えば辞書に分類されるのは「電子　顕微鏡」のような複合語として成立するペアである．仮辞書に分類されるのは

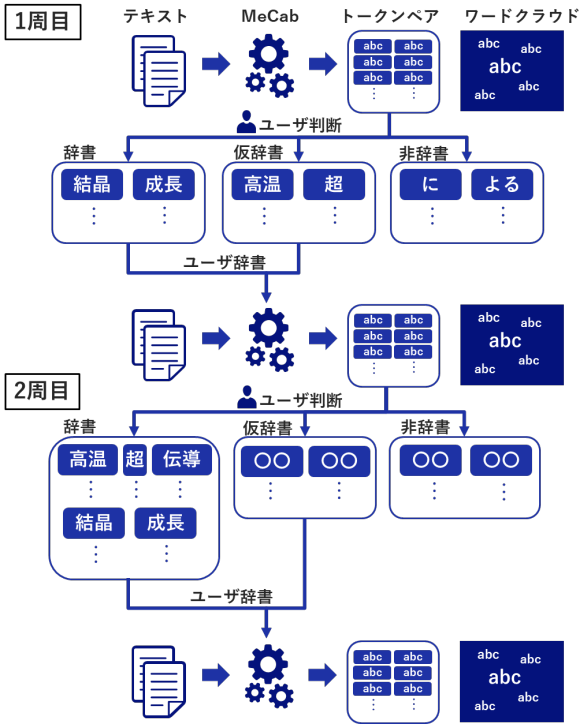

図 3　ひめのりによる複合語抽出の流れ.

図 4　ユーザ判断画面.

3 トークン以上からなる複合語の一部である「高温　超」（高温超伝導の一部）のような，そのままでは意味を持たないが，より長単位では意味を持ちそうだと判断されたペアである．仮辞書に登録されたトークンは，次周のトークンペア集計では繋がった状態で出てくるため，次周の判断によってはさらに長いトークンとなっていく．こういった意味を持たないトークンペアをユーザ辞書に登録せずに仮辞書に登録することで，ユーザ辞書の無意味な肥大化を防いでいる．非辞書に入るのは「に　よる」のような複合語として成立しないペアである．一度非辞書に登録されたトークンペアは，次周以降は頻度集計からは除かれる．分類は画面上の該当ボタンのクリックで実行できるが，キーボードの矢印キーでも

実行できる．必要なトークンペアについてのユーザ判断が終わったら[3]，辞書と仮辞書に含まれたトークンペアを見出し語とした MeCab のユーザ辞書を作成する．この際，品詞は全て名詞とし，生起コストは MeCab に分割されないようトークンの文字数と周回数に基づいて決定される．このユーザ辞書を利用してテキストを再び分かち書きし，トークンペアの頻度集計とワードクラウドの生成を行う．新しいユーザ辞書により生成されたワードクラウドは，図 5 に示す画面で直前の周回のワードクラウドと比較され，次周を実行するかの判断をユーザに促す．ここまでの一連の処理が 1 周目に該当する．2 周目

図 5　周回の終了画面．

は，直前の周回で生成されたトークンペアへの判断をユーザに促すところから始まり，更新されたユーザ辞書でワードクラウドが生成されて終わる．図 3 では 2 周目で「高温超伝導」という複合語が辞書に登録されたことが分かる．この処理を繰り返すことでトークンペアは徐々に長くなり，最終的にはユーザが所望する複合語のリストを得ることができる．

ストップワードリストの編集には図 5 に似た別の画面が用意されている．ユーザはトークンリスト中のトークンをクリックすることで，動的にストップワードリストを変更し，ワードクラウドの変化を確認できる．

4　実験

システムの効果を評価するために，ひめのりを利用した場合としない場合それぞれについて 10 分間のチャンキング作業を設定し，2 分ごとに作業のス

3）　ユーザは必ずしも全てのトークンペアを処理する必要はない．例えば 10 回以上出現するペアまで処理する，といった使い方もできる．

表 1　BIO タグを用いた長単位チャンキングの例．

Token	化合	物	半導体	超	格子	の	無秩序	化	と	その	応用
正解	B	I	I	I	I	O	B	I	O	O	O
入力	B	I	I	B	I	O	O	O	O	O	O

コアを算出した．対象文書として J-Stage の API で取得した応用物理学会誌の 1940-1942 年の論文タイトル 409 件を利用した．このデータセットは古い専門用語が使われており，特に複合語辞書生成の重要性が大きいと判断したため選ばれた．チャンキング作業は MeCab と IPADIC 辞書が生成した分かち書きテキスト中のトークンを接続するものとした．正解データは，分野固有の専門用語を可能な限り長単位で接続することを基本方針とし，一人の材料科学者が用意した．スコア計算には CONLL2003 の固有表現抽出タスク [22] の BIO タグを参考に，適合率，再現率，F1 値を下記のように定義した：

$$\text{適合率 (Precision)} = \frac{\text{正解との一致タグ数}}{\text{ユーザの BI タグ数}},$$

$$\text{再現率 (Recall)} = \frac{\text{正解との一致タグ数}}{\text{正解の BI タグ数}},$$

$$\text{F1 値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}}.$$

タグ付けの例を表 1 に示す．この場合，ユーザ入力の適合率，再現率，F1 値はそれぞれ 0.8, 0.57, 0.67 となる．ひめのりを使用する場合（w/ Himenori）は 2 分ごとにひめのりの周回作業を終了し，各周回のユーザ辞書によって分かち書かれた結果を BIO タグに変換した[4]．ひめのりを使用しない場合（w/o Himenori）は，図 6 に示すようにアンダースコアを分かち書き文字としたデータを用意し，テキストエディタを用いた手作業による区切り文字の除去をタスクとした．この際も 2 分ごとに作業を保存し，結果を BIO タグに変換した．実験は 8 人の材料系研究従事者に対して行った．被験者はチャンキング作業を同じデータセットに対し w/ Himenori，w/o Himenori でそれぞれ 10 分間行った．作業を行う順番による影響を確認するため，被験者を 4 人ずつ 2 つの群に分けた．1 つの群は先にツールを利用し（Himenori 1st），もう一つの群は先にテキストエディタで作業を行った（Himenori 2nd）．

5　結果と考察

作業者群ごとの平均 F1 値の推移を図 7 に示す．作業の順番にかかわらず，全ての時間で w/ Himenori

4）　初期トークンが分かたれた文章はスコアの計算に含めなかったが，そのような文章は全体の 5 %程度だった．

```
1    高速度_現象_の_解析
2    獨逸_小型_受信_器_に_就_て
3    風向_風速_計
4    窒化_處理_に_依る_變形_に_就_て
```

図 6　テキストエディタを用いた手動チャンキングタスク．トークンを区切るアンダースコアを手動で削除する．

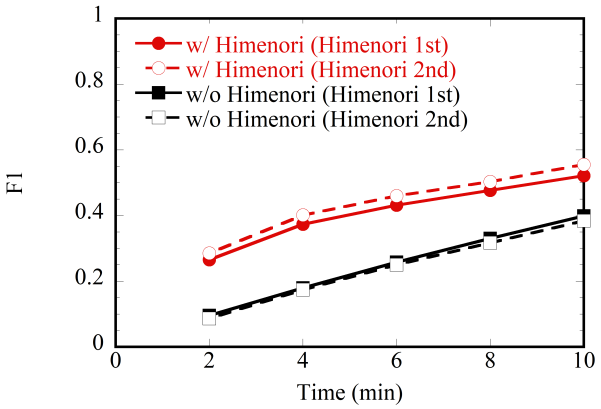

図 7　F1 値の推移．各点は 4 人の作業者の平均値を示す．

の F1 値が w/o Himenori の F1 値を上回っていることが分かる．またそれぞれの群において，後に作業した結果の F1 値が高く，先の作業でデータセットの特徴を掴んだことで，後の作業の処理速度が向上したと思われる．しかしながらその影響は極わずかであり，専門用語抽出タスクはひめのりによって加速されたと言える．また，w/o Himenori のカーブが線形なのに対し，w/ Himenori は w/o Himenori よりも高い値をとっているが 4 分を過ぎた頃には緩やかな傾きになる．次に，図 8 に F1 値の内訳である

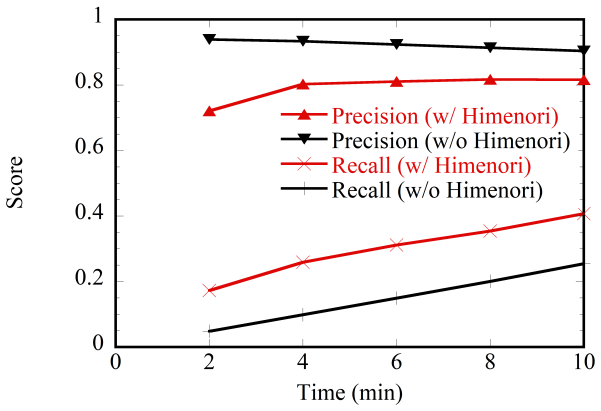

図 8　適合率（Precision）と再現率（Recall）の推移．各点は 8 人の作業者の平均値を示す．

適合率と再現率の平均値の推移を示す．作業の順序による差は極わずかであったため，2 つの作業者群をまとめた 8 人の結果の平均値を用いた．適合率は w/o Himenori は時間によらず高い値を維持するのに対し，w/ Himenori は最初に低い値をとり 4 分以降で一定値となる．この違いが生まれるのは，w/o Himenori では同じ判断を下せる限り時間経過によって適合率が変化しないのに対し，w/ Himenori では始めの周回で仮辞書に追加されたトークンペアに短単位で BIO タグが貼られ適合率が低く出るからである．これら仮辞書に登録されたトークンペアは，その後の周回でより長単位の BIO タグが貼られるため，最終的には適合率は上昇する．再現率は，w/o Himenori がゼロ点から線形に上昇するのに対し，w/ Himenori はゼロ点から 2 分までの傾きが大きく，4 分を過ぎた時点で w/o Himenori の傾きと一致する．w/o Himenori の線形性は単純作業の繰り返しとして説明できるが，w/ Himenori の傾きの変化は頻出するトークンペアの一括処理による効果が 4 分頃までででなくなることを示唆している．今回設定した実験時間の範囲内では，w/ Himenori の 2 分と w/o Himenori の 6 分の F1 値がほぼ同一であり加速効果は最大で 3 倍程度とも言えるが，より長時間の作業では F1 値が逆転し得る．これはひめのりに有利な処理時間範囲があることを示唆しており，データセットの特徴とも相関があると考えられる．今後，頻出するトークンペアの出現頻度の分布などから，ひめのりの得意なデータセットの特徴を示していきたい．

この実験は，LOC や PER 等の分類はしていないものの，ひめのりの固有表現アノテーションツールとしての可能性にも注目して設定された．w/o Himenori の作業はスパンアノテーションと同種の作業であり，ひめのりによる作業の加速効果が認められた．今後は辞書生成だけでなく固有表現アノテーションツールとしての機能も追加し，他のツール [6–10] との比較も行いたい．

6　おわりに

辞書生成機能を備えたワードクラウド作成器「ひめのり」を開発し，複合語抽出の時間対効果を実験的に明らかにした．ひめのりを使うことで，様々な分野の専門家が自然言語処理の細部を意識することなく効率的に形態素解析辞書を生成できる．今後はひめのりと相性の良いデータセットの特徴を明らかにするとともに，固有表現アノテーションツールとしての可能性も探る．

謝辞

本論文をまとめるにあたり，日本電気株式会社の石川開氏には，実験の設定や評価について有益な助言をいただきました．

参考文献

[1] 工藤拓. MeCab: Yet another part-of-speech and morphological analyzer. https://taku910.github.io/mecab/.

[2] 中川裕志, 湯本紘彰, 森辰則. 出現頻度と連接頻度に基づく専門用語抽出. 自然言語処理, Vol. 10, No. 1, pp. 251–258, 2003.

[3] 橋本泰一, 藤井敦. 特許文書のための形態素解析辞書の構築. 言語処理学会 第 18 回年次大会, 2012.

[4] 五井野琢也, 濱上知樹. Bert を用いた医療文書からの固有表現抽出. 計測自動制御学会 第 48 回知能システムシンポジウム, 2021.

[5] 高間康史, 阿部美里. テキストデータマイニング統合環境を利用した看護記録からの専門用語辞書作成支援ツールの提案. 第 27 回人工知能学会全国大会, 2013.

[6] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for NLP-assisted text annotation. In **Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics**, pp. 102–107, 2012.

[7] Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. YEDDA: A lightweight collaborative text span annotation tool. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics**, pp. 31–36, 2018.

[8] 山村崇, 嶋田和孝, 吉川和, 岩倉友哉. Tatara: 支援機能を持ったアノテーションツールの構築. 言語処理学会 第 25 回年次大会, 2019.

[9] Prodigy · An annotation tool for AI, Machine Learning & NLP. https://prodi.gy/.

[10] doccano: Open source annotation tool for machine learning practitioners. https://github.com/doccano/doccano.

[11] 小澤俊介, 内元清貴, 伝康晴. Bccwj に基づく長単位解析ツール comainu. 言語処理学会 第 20 回年次大会, 2014.

[12] Sudachi: A Japanese Tokenizer for Business. https://github.com/WorksApplications/Sudachi.

[13] 佐藤敏紀, 橋本泰一, 奥村学ほか. 単語分かち書き用辞書生成システム NEologd の運用-文書分類を例にして. 情報処理学会研究報告 自然言語処理 (NL), Vol. 2016, No. 15, pp. 1–14, 2016.

[14] Hakan Altınçay and Zafer Erenel. Analytical evaluation of term weighting schemes for text categorization. **Pattern Recognition Letters**, Vol. 31, No. 11, pp. 1310–1323, 2010.

[15] AI テキストマイニング by ユーザーローカル. https://textmining.userlocal.jp/.

[16] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. **Journal of documentation**, Vol. 28, No. 1, pp. 11–21, 1972.

[17] React A JavaScript library for building user interfaces. https://reactjs.org/.

[18] FastAPI framework, high performance, easy to learn, fast to code, ready for production. https://fastapi.tiangolo.com/.

[19] https://www.sqlite.org/.

[20] Andreas Mueller. WordCloud for Python. https://amueller.github.io/word_cloud/.

[21] Noto Sans Japanese - Google Fonts. https://fonts.google.com/noto/specimen/Noto+Sans+JP.

[22] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In **Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003**, pp. 142–147, 2003.