情報処理学会論文誌 Vol.62 No.2 594–606 (Feb. 2021)

日本語形態素文字種境界法によるデータベース検索量の削減

米持 幸寿[1,a)] 大場 みち子[1]

受付日 2020年4月17日, 採録日 2020年11月5日

概要：商品名などの固有名詞には，複数の語を含む長い複合語がある．それらは市場に出回っている自然言語処理用の基礎データに含まれていないことが多い．音声インタフェースのようなテキスト対話システムをビジネスシステムに適用する際，それら複合語を抽出するために業務データベースへの検索が多発することになる．人が話しかける対話型システムでは幅広い表現に対応するために大規模語彙を統合する必要もある．これに対応するため，シソーラスや Linked Open Data（LOD）といった外部データソースへのアクセス要求もある．そのようなシステムでは業務データベースや LOD エンドポイントへの検索回数が膨大になることを抑制する手法が必要となる．本研究では，業務データベース，シソーラス，LOD の語彙を統合検索できるアーキテクチャと，検索回数を抑制する方法として「日本語形態素文字種境界法：Japanese Morpheme Character Type Boudery（JMCTB）Method」を提案する．対話テキストコーパスを使う実験により，単純な N gram 形態素検索と比較して検索回数を 96%削減できることを示す．

キーワード：辞書/語彙意味，テキスト処理，ユーザインタフェースとインタラクティブシステム

Japanese Morpheme Character Type Boundery Method to Reduce Database Search

Yukihisa Yonemochi[1,a)] Michiko Oba[1]

Received: April 17, 2020, Accepted: November 5, 2020

Abstract: Proper nouns like product names include Multi Word Entity. They are often not registered in fundamental data for Natural Language Processing in the market. Searching Business Database is required to extract such words when Text Interactive System like voice interface is applied to a business system. Integrating Huge Lexcon is also required to respond to wide expression on interactive system which human talk to. Accessing external datasource includes thesaurus or Linked Open Data (LOD) for such requirement. Both of these requirements, a method to reduce the number of times of searching. This research proposes the architecture to integrate lexicons of Business Database, thesaurus, and LOD with a method Japanese Morpheme Character Type Boudery (JMCTB) to reduce the number of times of searching. The experiment using an Interactive Corpora shows 96% of number of searching can be reduced from simple N gram characters search method.

Keywords: dictionaries/lexical semantics, text processing, user interfaces and interactive systems

1. はじめに

IT システムとの最新のインタラクション方式として音声対話型インタフェースがある．音声認識・合成の性能の向上と，音声アシスタントやスマートスピーカの普及を背景に，音声対話型インタフェースやテキストチャットボットなどの，業務用テキスト対話型インタフェースのニーズが高まりつつある．業務用テキスト対話型システムにおいて，入力テキストから業務に関連する固有名詞の抽出は重要なタスクである．業務用対話型システムには，次のような要件がある．

- 常時稼働を続けること（無人チャットボットなど）
- 反応時間が短いこと（ユーザ入力から数秒）
- ユーザの幅広い語彙に対応できること

[1] 公立はこだて未来大学
Future University Hakodate, Hakodate, Hokkaido 041–8655, Japan
[a)] g3119008@fun.ac.jp

- 最新の語彙に対応できること

その際，業務データベース，類義語集であるシソーラス，および，インターネットに置かれプログラム処理しやすい形式のデータやプログラム呼び出し可能なサービスである Linked Open Data（LOD）を検索するニーズがあり，本研究はその検索システムのアーキテクチャおよび検索量削減の手法の提案を目的としている．固有名詞抽出時のデータベース検索回数の削減を目的に，形態素解析結果と日本語特有の文字種の境界を活用する「日本語形態素文字種境界法：Japanese Morpheme Character Type Boudery（JMCTB）Method」を提案し，文章集データセットである対話テキストコーパスを使った実験により有効性を示す．

2. 固有名詞抽出手法の現状

自然言語処理：Natural Language Processing（NLP）において，固有名詞抽出は研究対象としてさかんである．本章でよく知られた方法，過去の研究などを引用し説明する．その後，対話型システムに各方法を適用した場合の問題を例を挙げて説明し，解決すべき課題を 3〜5 章で述べる．

2.1 パターンマッチング

最も簡単な方法はパターンマッチングで，該当の語彙が入力テキスト中にあるかどうかを照合する方法である．この方法では前方一致，後方一致，単純マッチング，正規表現 [1] などいくつかの方法がある．現存する多くのプログラミング言語ライブラリで実装されており，プログラミングも容易である．具体的には Java 言語であれば，String クラスの startsWith()，endsWith()，indexOf() メソッドや，Pattern/Matcher クラスなどを使う．

2.2 N gram 文字列検索

N gram は，入力テキスト上で開始文字をずらしながら n 文字の文字列を生成し，その語が語彙に存在するかを確認する方法であり，NLP ではポピュラーな方法である [2]．パターンマッチングと比較すると，入力テキストから生成する文字列が語彙にあるか探す方法であり，逆方向の検索といえる．このため大規模語彙との相性がよい．語彙中で語を検索する際には語彙辞書データの単語にインデックスを作ることで処理コストを減らすことができる．

2.3 最大エントロピー法

N gram インデックスを使い，語の切れ目を発見するための手法として最大エントロピー法 [3] がある．N gram で切り取られた言葉をインデックス化し，切れ目となった回数を統計し確率を求めることで「語」を抽出する手法である．教師なし学習の一種といえる．入力となる文章量が多い場合に，「出現頻度が多く，語の境界と考えられる」箇所を発見するために有効な手法である．

2.4 日本語形態素解析

日本語形態素解析は文字列を語彙素に分解（分かち書き）し，品詞推定する処理の総称である．形態素解析器（プログラム）が，固有名詞を正しく分けられることもある．日本語形態素解析の歴史は古く，茶筌 [4]，MeCab [5]，JUMAN++ [6] など C 言語実装，Apache lucene-gosen [7]/Kuromoji [8]，Sudachi [9] などといった Java 実装がある．

日本語形態素解析は隠れマルコフモデルや条件付確率場など統計的確率法を使って形態素の分断を行う．このため，品詞並びの確率を示す形態素解析辞書データが必要である．よく知られている辞書データは，大規模コーパスから学習した確率データであり，機械学習の一種といえる．表記語，品詞，前後の品詞並び，出現確率などを大規模コーパスなどから学習したデータである．古くから IPA-DIC（IPA），NAIST-Jdic [10]，Unidic [11] などが存在する．これらは形態素解析プログラムとセットになっていることが多く，標準辞書，あるいはシステム辞書と呼ばれる．更新頻度が高くないことから，新出の語に弱いとされている．辞書を高頻度でメインテナンスするプロジェクト NEologd [12] も近年発表されており，Web で得られる語の多くが適用されるようになった．

いくつかの日本語形態素解析器でユーザ辞書が利用できる [13]．ユーザ辞書とは，システム辞書とは別に，形態素解析器を使うユーザが独自に作り追加できる辞書ファイルである．業務用語，大規模シソーラスや LOD の単語を，適切に辞書登録することで，形態素解析時に複合語である固有名詞を抽出できる．

2.5 深層学習による固有表現抽出

近年 NLP 界でさかんな研究に，深層学習を使う固有表現抽出：Named Entity Recognition（NER）がある．固有表現：Named Entity（NE）は MUC [14] で規定された表現の種類だが，IREX [15] で「固有物名」が含まれており，これを固有名詞抽出手法と考えることができる．医療テキストを対象とした NER [16] のように特定業務分野に特化した研究や，MultiWOZ [17] のような研究用データセットを使った NER 研究 [18] などがある．学習データとして固有表現アノテーションつきコーパスを使い「固有表現の切れ目」を学習させる手法であり，Bi-LSTM [19] が幅広く使われている．

以上，固有名詞を抽出する手法を説明した．

1 章で説明した業務用対話システムの要件「常時稼働を続ける」，「反応時間が短い」，「ユーザの幅広い語彙に対応」，「最新の語彙に対応」を満たすことを想定した場合の適合性を検討する．「固有名詞をデータベース検索する」という本研究の目的において，これらの手法の問題と，解決すべき課題を 3 種類のカテゴリに分けて次章以降で述べる．

3. パターンマッチングの問題と課題

パターンマッチングはプログラミング言語の機能で簡単に処理できるが，語の抽出では以下の問題があり，解決すべき課題を定義する．

- 照合回数が多い

パターンマッチングでは語彙に存在するすべての語を照合してみる必要がある．たとえば，データベースに 10 万語が登録されていて「どれかの語が含まれているか」を確認するためには，10 万回照合が必要である．業務用語データベース，シソーラス，LOD といった大規模語彙の処理には向かない．

課題 1：大語彙での処理量の抑制

大語彙を取り扱うことが必須のため，大量の照合を行わない方策が必要である．

4. N gram 検索の問題と課題

N gram 検索は NLP では標準的な手法で，インデックス作成に広く使われているが，データベース検索において以下の問題があり，解決すべき課題を定義する．

4.1 検索回数が多い

N gram 文字列を作ると，入力文が長い場合には組合せ数が大きくなり，データベース検索回数が増える．n=2 以上の N gram 文字列であれば，n 文字のセンテンスに対して $\frac{n(n-1)}{2}$ 回のデータベース検索が発生する．具体的には，入力テキストが 100 文字であれば，4,950 回である．形態素解析辞書のようにメモリ内に展開されているインデックスを使った検索では大きな問題にならないが，1 文の処理ごとに社内データベースや Web サービスを呼ぶ回数としては負荷が大きい．

課題 2：検索回数を抑制する

本研究で対象にするシステムは音声対話システムやテキストチャットボットなど，人とリアルタイムに対話するシステムであることから高いレスポンス性能が要求される．入力テキストから業務に関連する固有名詞を抽出する際に，データベースや Web サービスを検索する回数を抑制する必要がある．

4.2 形態素を分断してしまう

N gram 文字は形態素の分かち書きと関係なく文字列を切り出すため，形態素の中間を切れ目として「一見言葉に見える」ものをデータベース検索してしまうことがあることが古くから指摘されている．例 1 は，「日数」を誤って抽出してしまう例である．

これ以降の例を示す箇所において，カンマ「,」は形態素区切り，太文字・アンダーライン付きの文字列 は抽出される箇所を示す．

例 1：形態素を分断する例

入力：「今日数えてみたら」

形態素解析結果：今 **日, 数** え, て, み, たら

誤った抽出箇所：日数

課題 3：形態素を分断しない

固有名詞を抽出する際，形態素を分断しないようにする．

4.3 長い語を分断してしまう

業務用語データベース，シソーラス，LOD には短い語も語彙として登録されており，より長い語を分断する短い語が抽出されてしまうことがある．例 2 は「インターナショナル」と「インターナショナルスクール」の両方がデータベースに発見できる場合に，下線部が示すように「インターナショナル」が誤って抽出されている例である．

例 2：短い箇所を誤って抽出する例

誤：**インターナショナル**, スクール

正：**インターナショナル, スクール**

課題 4：短い語で長い語を分断しない

長い語に含まれる短い語を抽出してしまうことにより長い語が抽出されない事象を防止する．

4.4 検索ノイズ

同一文字種で挟まれた文字列が，助詞などが並んだ箇所で誤って抽出されることがある．例 3 は「あまた」,「たより」,「やけど」,「ねんね」,「たわけ」が実際に間違って抽出されたケースを示している．これらの言葉はいずれもシソーラスデータである日本語 WordNet [20] で見つかる．特にひらがなの語尾で終わるケースが多数みられる．

例 3：ひらがなを分断する誤り例

誤：～ま, **あ, また**～

誤：～思っ, **た, より**～

誤：～思う, ん, **や, けど**～

誤：～そう, **ねん, ね**～

誤：～通わ, せ, て, **た, わけ**, じゃ, ない, し～

ただし，「たより」,「やけど」,「たわけ」が実際に文に存在するような場合，例 4 のように形態素解析器が適切に 1 つの形態素にわけてくれるため，これらは問題にはならない．

例 4：正しく形態素になるひらがな

正：～最近, あまり, **たより**, が来ない

正：～あー, **やけど**, したわっ

正：～この, **たわけ**, がっ

課題 5：言葉でないものを抽出することを防止する

文章の途中に出現する，語でないものを誤って抽出することを防止する．

4.5 よくある「いいまわし」の頻度が高い

N gram インデックスを最大エントロピー法で検索する

と「そう, なん, です, か」のような，よくある「いいまわし」が大量に発見されてしまい，固有名詞の候補が浮かび上がらない．最大エントロピー法は語彙との比較を行わないため，語彙に含まれる語の発見には使えない．

課題 6：「いいまわし」の誤抽出の抑制

大量の過去データから頻出パターンとして「いいまわし」が抽出されるのを防ぐ．

4.6 数字表現が検索ヒットする

「2 日」「3000 円」のような数字表現が LOD の中心的存在である Wikidata [21] で検索ヒットする．多くの場合，その日はなんの記念日か，などが書かれており業務用語に該当しないため，適切な抽出とはいえない．

課題 7：数字表現を検索しないようにする

業務要件によって差異はあるものの，数字表現は業務用語ではないとして検索対象外とする．

4.7 N gram と文字種による抽出

N gram インデックスに文字種を適用した研究もある [22]．文字種が変化する箇所にインデックスを生成する手法を提案しており，たとえば，「シート読み取り装置を開発した」という入力文に対して，文字種が変化しカタカナまたは漢字から始まる箇所にインデックスを作り 3 文字を取り出すと「シート」,「読み取」,「取り装」,「装置を」のようになる．しかし，検索システムのためのインデックスを作った例であり，本論文の「入力文中の固有名詞検索」という主旨とは少し違う．また，文字が切り替わる先頭部分のインデックスのみを活用しようとしているが，本論文の提案では，語の候補の終了箇所にも文字種境界を活用していることが違うが参考にはなる．

4.8 文字種による切り出し

文字種切り出しによるキーワード抽出の研究 [23] もある．理解しやすいように「むかしあした天気になあれ読んでたなぁ」を形態素解析と文字種切り出しで比較した例を**表 1** に示す．文字種だけを使う場合「あした天気になあれ」を取り出しにくいことが分かる．形態素解析と組み合わせていない点，助詞が含まれた語の抽出に弱いと推測される点において，本論文の提案とは異なる．

5. 機械学習の問題と課題

学習データを使った機械学習には以下の問題があり，解決すべき課題を定義する．辞書データと学習用コーパスから，プログラム（学習器）によって形態素解析辞書が作られる [24] ため，形態素解析についても言及する．形態素解析辞書は，登録語表記とその前後の品詞の組合せの出現率を集計したものであり確率モデルに相当するものと考えられる．

表 1 形態素解析と文字種切り出しの比較
Table 1 Comparison of morpheme analysis and character type separation.

形態素解析	文字種切り出し
むかし	むかしあした
あした	天気
天気	になあれ
に	読
な	んでたなぁ
あれ	
読ん	
で	
た	
なぁ	

5.1 長い商品名

本研究が対象とする業務用語は，形態素数の多い複合語や，助詞などを含み文節のように見える長い固有名詞である．企業の特殊用語や，小説タイトル，漫画タイトル，映画タイトルなど商品名によく見られる．たとえば，漫画「あした天気になあれ」，小説「桐島，部活やめるってよ」，映画「燃えよ剣」，アニメ「鬼滅の刃」などが該当する．これらが文に含まれ，固有名詞が抽出される様子を例 5 に示す．

例 5：「文節のようなタイトル」を含む文

むかし, **あした, 天気, に, な, あれ**, 読ん, で, た, なぁ
なん, だ, っけ, **部活, 辞め, る, って, よ**, みたい, な, 映画
燃え, よ, 剣, 読ん, だ, こと, ない, 件
鬼滅, の, 刃, やばい

文節でできている固有名詞の抽出は，品詞並びの出現頻度の統計モデルを使う機械学習技術だけでは解きにくい問題である．形態素解析では品詞並び頻度の学習データから切れ目を推定することから，カンマによって例示されているように固有名詞を細かく分断してしまう．形態素解析辞書データには上記のような固有名詞は含まれておらず，標準では抽出できない．形態素解析器による品詞推定で一般名詞の抽出は可能だが，例示されているような複数の形態素を含む複合語の抽出ではうまく機能しないことが分かる．深層学習でも標準で配布されている辞書データの語彙に含まれない場合，やはり上記の固有名詞抽出に失敗する．

課題 8：長い複合語の固有名詞の抽出

長い複合語の固有名詞を抽出する方法が必要である．

5.2 頻繁に増える業務用語

業務用語は頻繁に増え，データベースに日々追加更新される．

課題 9：頻繁に増える業務用語に対応する

形態素解析辞書や深層学習プログラムの配布されている辞書データを更新せず，増加した語をリアルタイムに抽出できる手法が必要である．

5.3 バッチ処理による辞書更新

形態素解析辞書や深層学習を利用する自然言語処理の確率モデルは，独自のコーパスから作ることも可能である．いくつかの形態素解析器ではユーザ辞書を生成して追加することができる．ユーザ辞書を活用して固有名詞抽出を行うことも可能である．

日本語形態素解析が研究され始めた初期の 1990 年代は，現代よりコンピュータが非力で実メモリも少なかった．C 言語で研究開発が行われ，モノリシック（単一独立型）システムとして研究されてきた経緯もあり，辞書データのメモリ上での小型化，アクセス効率化などを考慮した結果，メモリ上の配列などの構造体をファイルに直接吐き出したものとして作られている．これが影響して，いったん構築された語彙辞書データに新語をシステム稼働中に追加することができない．語を追加するには辞書データのコンパイル，システムの再起動を余儀なくされる．

課題 10：バッチ処理によるユーザ辞書更新の回避

課題 9「頻繁に増える業務用語に対応する」に反するため，バッチ処理によるユーザ辞書を更新する方法を回避する．

5.4 新バージョンの配布

過去に作られた自然言語処理プログラムは，その開発者からプログラムコードや解析用辞書データなどが改新や保守されずに放置されることが多いが，ときが経った後に更新されて発表されることもある．形態素解析辞書そのものを日々更新する研究もある [12]．しかし，特定業務の用語が配布される辞書データに含まれることは期待できない．さらに，業務ごとに作成してしまうと，新しい版が配布されたときマージしなくてはならない問題が発生する．

課題 11：新バージョン配布を考慮する

既存の形態素解析器，形態素解析辞書，シソーラスなどに新語が追加され再配布されたとき，容易に置き換えられるように配慮する．

5.5 大語彙の要求

音声対話型システムなどでは業務語彙以外に大規模語彙を活用する必要もある．ここでいう「大語彙」とは，人が知りうる幅広い語彙のことである．業務語彙に存在しない類義語をユーザが発話入力することが考えられるからである．ユーザとなる人間は通常，業務語彙よりはるかに語彙量が多いことが背景にある．システムに登録されていない似たような言葉をユーザが発したときに，その語が固有名詞抽出されないために処理が不可能となる問題が発生する．

たとえば，商品として取り扱ってない「魚の名前」を顧客が発する可能性が考えられる．商品リストに「まぐろ」「鮭」があるが，「アジ」は掲載されていないとする．そのとき「アジはいくらですか」という入力に対して，システムは「アジ」を「魚である」と抽出できない．これにより「申し訳ありません，わかりません」などとシステムは処理を打ち切ることになる．抽出できた場合「アジは取り扱っておりません」と，より適切な回答を生成できる．

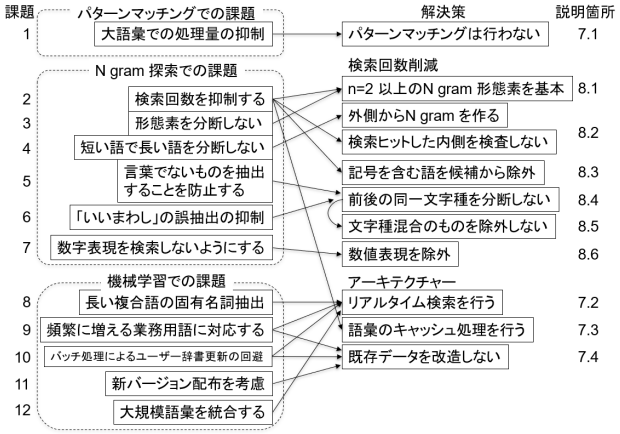

図 1 課題と解決策
Fig. 1 Challenges and solutions.

課題 12：大規模語彙を統合する

大規模語彙データを用いて，同じ意味を表す様々な言葉「類義語」，ある言葉の意味的に抽象化した言葉「上位語」，その逆の「下位語」などを適切に処理する必要がある．シソーラスデータや LOD を活用することが有効であり，それらを語彙として統合する必要がある．最も語彙が幅広く登録され，オープンに利用可能である LOD として Wikidata が知られている．同様の背景から，自然言語処理時の語彙の統合も提案されている [25]．

6. 課題に対する解決策

3〜5 章にて，パターンマッチング，N gram 文字列による検索，機械学習などを利用した場合の問題点と，それぞれの解決すべき課題を説明した．それらの解決すべき課題と，本研究で提案するアーキテクチャおよび検索回数削減手法で用いられる解決策について，その関連を**図 1** に示す．

課題は，パターンマッチングでの課題，N gram 検索での課題，機械学習での課題にグループ化できる．それらに対する解決策は提案アーキテクチャの考え方と，検索数削減手法にグループ化できる．それらを 7 章および 8 章でそれぞれ説明する．

7. 提案アーキテクチャの考え方

入力テキスト中に含まれる語が複数の大規模語彙セットのどこに見つかるか，を調べるため，**図 2** が示す，語の候補を抽出し検索を行うアーキテクチャを設計する．テキストを入力するとローカルの辞書ファイル，リモートの外部データベース，Web サービスなどに問合せを行うことで語彙を総合して，固有名詞を抽出する．本章では，アーキテクチャを設計するうえでの指針を説明する．

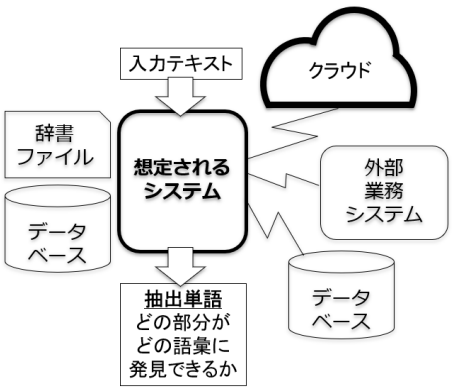

図 2 想定されるシステム
Fig. 2 Assumed system architecture.

7.1 パターンマッチングは行わない

業務データベース，シソーラスデータ，LOD などを総合することにより大語彙となることが予想される．課題 1「大語彙での処理量の抑制」を考慮し，パターンマッチングは不向きであることが明らかであり，パターンマッチングは行わないこととする．

7.2 リアルタイム検索を行う

本研究で対象とするシステムでは，バックエンドデータを検索して「どのデータベースに登録されている語であるか」を確実にとらえる必要がある．課題 6「いいまわしの誤抽出の抑制」があり，語彙検索の技術ではない最大エントロピー法は適用できない．課題 8「長い複合語の固有名詞の抽出」のために語の並びを学習する方法だけでは不十分である．課題 9「頻繁に増える業務用語に対応する」ために，ふだんから日々更新のかかっているデータベースを直接参照する必要がある．課題 11「新バージョン配布を考慮」といった課題に対応するためには言語リソースを独自に作って置き換えることも不適切と考える．これらを総合すると，オンラインによるリアルタイム検索を行う必要があると考える．また，課題 12「大規模語彙を統合する」に対応するため，業務データベースだけでなく，シソーラスや LOD も同時に検索対象にする必要がある．本研究の実験では WordNet と Wikidata を対象とする．

7.3 語彙のキャッシュ処理を行う

業務データベース，シソーラス，LOD などは外部リソースとして検索される語彙システムのため検索回数を抑制する必要がある．見つかった語のキャッシュデータを作ることで検索回数を減らすことができる．「見つからない語のキャッシュ」も検討したが，入力文章のほとんどがそこにリストされてしまうため肥大化するので意味がないとし，「見つからない語のキャッシュ」の考えは棄却する．

表 2 N gram 文字と N gram 形態素の比較
Table 2 Comparison of N gram character and N gram morpheme.

N gram 文字	N gram 形態素
あ	あした
あし	あした天気
あした	あした天気に
あした天	あした天気にな
あした天気	あした天気になあれ
あした天気に	天気
あした天気にな	天気に
：	：

7.4 既存データを改造しない

課題 9「頻繁に増える業務用語に対応する」，課題 11「新バージョン配布を考慮」に対応するため，形態素解析のシステム辞書，ユーザ辞書，シソーラスデータベースなどを作り直すことは行わない．既存のものをそのまま使い，新バージョンが発表されたときはそのまま置き換えられるようにする．

8. 検索回数抑制手法

アーキテクチャとして，オンラインでリアルタイムに業務データベースや LOD を検索することは前章で述べた．これに対し，誤った抽出を回避したり，検索回数を削減したりする方法が必要である．

形態素解析結果に文字種を考慮した複合語の候補抽出方法「日本語形態素文字種境界法：Japanese Morpheme Character Type Boudery (JMCTB) Method」を提案する．

8.1 n=2 以上の N gram 形態素を基本とする

検索候補の組み立てを，文字単位でなく形態素解析結果の形態素単位で行う．たとえば，「あした天気になあれ」という言葉を形態素解析で分かち書きすると「あした・天気・に・な・あれ」と形態素列に分割される．この形態素を基準に N gram を作る．比較のため，N gram 文字と N gram 形態素で抽出する違いを**表 2** に示す．表示例では冒頭部分のみを示している．

1 形態素のものは，そのままでデータベース検索できるため，対象外とする．N gram 文字列（形態素ではなく）によるデータベース検索では，形態素の中央からまたいで別の言葉を発見し抽出してしまうことがある．課題 3「形態素を分断しない」への対応として，形態素解析後のリストを使い N gram 形態素を生成し，データベース検索を行うことによりこれを防止する．

8.2 外側から N gram を作る

課題 4「短い語で長い語を分断しない」ようにするために，隣り合う形態素の組合せから N gram の組合せを作る

表 3 N gram を内側からと外側から行う比較
Table 3 N gram from inner or outer.

N gram を内側から作る（15 回）	外側から作り発見語の内側を検索しない（9 回）
俺, 宇宙	
俺, 宇宙, 戦艦	
俺, 宇宙, 戦艦, ヤマト	
俺, 宇宙, 戦艦, ヤマト, 観	俺, 宇宙, 戦艦, ヤマト, 観, た
俺, 宇宙, 戦艦, ヤマト, 観, た	俺, 宇宙, 戦艦, ヤマト, 観
宇宙, 戦艦	俺, 宇宙, 戦艦, ヤマト
宇宙, 戦艦, ヤマト	俺, 宇宙, 戦艦
宇宙, 戦艦, ヤマト, 観	俺, 宇宙
宇宙, 戦艦, ヤマト, 観, た	宇宙, 戦艦, ヤマト, 観, た
戦艦, ヤマト	宇宙, 戦艦, ヤマト, 観
戦艦, ヤマト, 観	宇宙, 戦艦, ヤマト
戦艦, ヤマト, 観, た	観, た
ヤマト, 観	
ヤマト, 観, た	
観, た	

際に外側から検索する．

入力テキストの長さを length，N gram 形態素の最初の形態素のインデックスを start，最後の形態素のインデックス+1 を end とする．たとえば，形態素 10 個の文字列を考える．インデックスが 0 から始まる場合，テキスト全体は start=0，end=10 で表すことができる．2 番目から 4 番目の 3 個を取り出す場合，start=1，end=4 で表すことができる．このような切り出しインデックスは，Java の subString 関数などによく使われている．外側（start=0, end=length）から検索を始め，start を後方へ，end を前方へずらすことで外側から N gram を作ることができる．

語彙検索にヒットした場合は次のフラグメント（start=end，end=length）へ移動することで，すでに検索ヒットした形態素列をの内側を処理しないようにする．

たとえば，「俺宇宙戦艦ヤマト観た」という入力文を形態素解析によって分かち書きすると「俺, 宇宙, 戦艦, ヤマト, 観, た」と分割される．N gram を生成し Wikidata を検索すると，「宇宙戦艦」「宇宙戦艦ヤマト」が見つかる．ここで説明しているように外側から N gram を生成して検索し「宇宙戦艦ヤマト」が見つかった時点で内部の検索を止めることで「宇宙戦艦」および「戦艦」「ヤマト」から始まる N gram を検索しないことになる．その様子を**表 3** に示す．下線のある箇所は，Wikidata で見つかる語である．その次の候補は「初めて, 観た」である．この例では検索回数を 6 回減らすことができ，「宇宙戦艦」が「宇宙戦艦ヤマト」を分断して抽出されることを防いでいる．

長い語の分断を防止でき，内側を検索しないことでデータベースや LOD の検索回数の抑制の効果もある．

8.3 記号を含む語を候補から除外

記号を含む文字列は語彙にある可能性が少ないと考え，データベース検索候補から除外する．！”＃＄％＆’（）-＾￥＝～｜@「；：」,．・￥‘{＋＊} ＜＞？＿といった文字列を含む語が業務データベース，WordNet，Wikidata 上の語彙に含まれることはほとんどない，と考えられるからである．

ただし，この機能はコーパスによる実験のときのみ有効で，音声対話型インタフェースの実装ではあまり問題にならないことが予測できる．「桐島，部活やめるってよ」のような映画タイトルに記号が含まれている例があるが，本研究が主な対象としているうち，音声対話型システムにおいて，このようなカンマを含んだ文字列が音声認識によって入力される可能性はほとんどない．

8.4 前後の同一文字種を分断しない

4.4 節で述べた「あまた」「たより」「やけど」「ねんね」「たわけ」などの文字列は，前後に同じ文字種が並んでいるときに誤って抽出されていることが 9 章で用いる実験データの調査で判明した．ここでいう文字種とは，日本語で普通に混在する，ひらがな，カタカナ，漢字（漢数字），英字（英数字），記号といったものである．そして日本語の言葉は文字種が切り替わる箇所に言葉の切れ目が存在する可能性が高いという常識的な考え方が適用できる．

課題 5「言葉でないものを抽出することを防止する」に対して，N gram 形態素列を取り出したとき，前後の文字種を確認し，同一文字種を分断する場合は検索対象にしないことにする．後述する実験システムは Java で実装されており，データは UTF-8 を採用しているため，文字コードの範囲で判定が可能である．

今回，対策としてひらがなだけを対象としている．カタカナ，漢字，英字，数字，いずれにおいても同一文字種が並ぶ中間を抜き出したことによる誤った検索結果が示されているものが見つからないためである．

9 章で示す実験データにて対策前に実際に間違って抽出された文字列を例 6 に示す．

例 6：ひらがなを分断する誤り例

誤：じゅう, **に, ぶん**,’, に, あっ, て,,
誤：同級生, と, **か, も, い**, た, し
誤：女性, は, し, **たい, よう**, に, 生きる
誤：で, も, 作り, **から, し**, たら

にぶん（二分），かもい（鴨居），たいよう（太陽），からし（辛子）など，ひらがなでも言葉として認識できるものがひらがな部に多く見つかる．

8.5 文字種混合のものを除外しない

前述の「前後の文字種を分断しない」解決策を施すと新たな問題が発生する．前後に並ぶ文字が同一だったために

候補から除外した場合においても，例 7 のような語が除外されてしまう．

例 7：混合文字種の例

正：大, 好き, な, メッセージ, は, お, 気に, 入り, に, 入れて, おく

この文において「お気に入り」を複合語として抽出したい．これはメールクライアントソフトウェアなどにおけるフォルダのことである．前出のルールに従えば，「は・お気に入り・に」のため「は・お」「り・に」とひらがな続きのため除外されてしまう．「お気に入り」には「気」「入」といった漢字が中に混じっており，9 章で用いる実験データを確認したところこのようなケースが多くある．複数の文字種で構成されているような場合は複合語の可能性が高く，除外しないこととする．混合文字種として除外から復活する語の例を例 8 に示す．

例 8：混合文字種として除外から復活する語

教育, ママ
ハングル, 文字
お, 台場
いい, カモ
ピーポ, 君
E, メール
t, 検定
日本, バレーボール, 協会
T, シャツ
セリエ, A
FC, 東京

8.6 数字表現を除外する

課題 7「数字表現を検索しないようにする」への対応として，数字表現を発見するフィルタを作成した．これにヒットするものは検索対象から除外する．

8.7 アルゴリズム

8.1 節から 8.6 節の解決策を施した JMCTB アルゴリズムを以下に示す．

```
procedure JMCTB(input)
  形態素列 = 形態素解析処理 (input)
  For start = 0 to 形態素列.length-1
    For end = 形態素列.length to start+1
      flagment = 形態素列.substring(start, end)
      flagment と前後の文字種を確認
      if 記号を含む continue
      if 同一文字種を分断 AND !混合文字種 continue
      if 数字表現 continue
      検索 (flagment)
      if found start = end
```

JMCTB の各対策の順序には意味がある．最初に記号を取り除く．記号はコーパスや書き言葉に含まれるものであるが，意味のある語の途中に記号が出現することはまれであるためである．次に「前後の文字種が同一種を除外→混合文字種のものを除外しない」はセットで考えるべきであり，単独で適用できるものではない．数字表現は最終的に残ったものから探して除外することで効率的に見つけ出し除外できる．これより前の時点で「数字表現なのか？」をチェックしても，そうでないものがほとんどであるため無駄なチェックとなる．

9. 実験

8 章で述べた方法を実際に実装し，効果を確認する．

本手法は，入力文から検索候補を作りデータベース検索をする際に，ありとあらゆる組合せを検索するのではなく，語として正しくかつ検索ヒットする可能性の高い組合せのみを選択する方法といえる．本手法を組み込んだ結果，より多くの語が検索ヒットすることはない．検索ヒットしたものが語として発見されたものとして抽出されることになり，その正確性は次の 3 つのことが要求される．

- 語でない箇所が検索ヒットしてしまうことを防ぐ
- そもそも検索ヒットしない検索候補文字列を除外する
- 検索候補の除外処理が応答時間に影響を与えない

これらを確認するため，3 種の実験を実施した．

9.1 対象データ

実験には入力テキストが必要である．音声対話型システムをターゲットとしている研究のため，書き言葉より話し言葉が適していると考え，話し言葉コーパスを利用する．大語彙データとして，シソーラスと LOD を利用する．

- 入力テキスト：BTSJ による日本語話し言葉コーパス（トランスクリプト・音声）2018 年版 [26]
  句点で句切ったテキストを使用．原文 112,235 センテンスを抽出し，実験インプットとする．
- 形態素解析辞書：system_full.dic（Sudachi の一部）
- シソーラス：日本語 WordNet 1.1
- LOD：Wikidata

対象になる語彙は，WordNet では 93,834 語，Wikidata では 2020 年 4 月 1 日現在のダンプデータから確認したところ日本語ラベルの数は 2,900,747 語である．

業務システムにデータベースなどがあれば SQL などを使って単語の有無を確認することもできるが，今回の実験システムでは業務データベースを統合していない．

9.2 実装

実験に使用したシステムの構成は以下のとおりである．

- 使用言語：Java
- 日本語形態素解析器：Sudachi 0.3.3-SNAPSHOT
  Tokenizer.SplitMode.C を使用

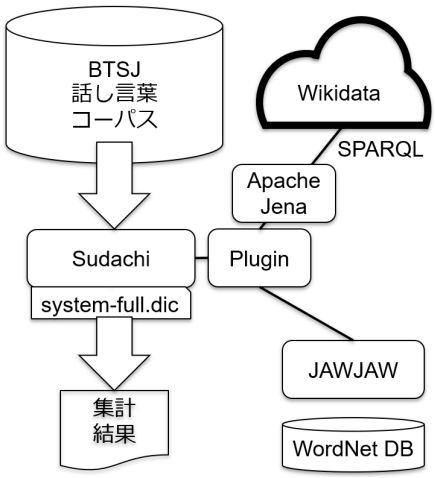

図 3 実験システムの構成図
Fig. 3 Experiment system architecture.

- シソーラスアクセス：JAWJAW 1.0.2 [27]
- LOD アクセス：Apache Jena 3.14.0 [28]

実システムであれば処理時に多くのシステムを検索することが求められるが，本研究の実験においては形態素解析器 1 つに対して，シソーラスとして日本語 WordNet，LOD として Wikidata を結合する．入力テキストを形態素解析し，その形態素解析から複数の形態素の組合せを WordNet や Wikidata からどれくらい効率良く発見できるか，を検証する．

図 3 は今回の実験システムの構成図である．Sudachi は Java で実装されているため直接呼び出しが可能である．また，Sudachi にはプラグイン機構があり，今回，PathRewritePlugin として JMCTB 法アルゴリズムを実装した．Sudachi にはいくつかの種類のプラグインがあるが，PathRewritePlugin は最終的な形態素出力の結果を編集するためのプラグインである．形態素解析器の出力そのものを修正することができるため，JMCTB 法の実装に最も向いている．検索回数，抽出数などはプラグイン内で数え，集計結果として出力する．Sudachi のプラグインの場合には Sudachi を使う既存システムにも恩恵を与えることができる．

他の形態素解析器でも，形態素解析後の出力結果を JMCTB 法を使って編集することで同様の結果を得ることができる．ただし，その場合は既存のソフトウェアにも修正を加える必要があるだろう．

日本語 WordNet には API ライブラリ JAWJAW を使ってアクセスしている．

Wikidata は，SPARQL エンドポイントであり，Apache Jena を使い SPARQL を使って検索できる．ただし，実験において，ベースラインとするためにすべての形態素組合せを Wikidata に問合せすることは現実的ではない．特に，全 N gram 形態素の実験の場合問合せ数が多すぎて実験プログラムが長時間終了しない．そこで，あらかじめ候補となる語のセットを最大エントロピー法と手動で作り，Wikidata に発見できる語のリストをローカル保存して実験に利用している．リストには Wikidata で発見できる 352 語を含んでおり，実験のすべての集計に共通で使用している．

Wikidata では，登録されているレコードの代表的な呼び名を Label，別名を AltLabel という識別子で管理しており，それらを対象とする．

Wikidata へのアクセスにはコード 1 の SPARQL クエリを使い，件数がゼロでないものを「語彙が見つかる」とする．このクエリは，対象名称がラベルになっていて，なにかのインスタンス（固有名詞と考えられるもの）と，別名に指定されているケースである．この検索方法は，より厳しい条件にすることも，より緩い条件にすることもできる．

コード 1
```
SELECT DISTINCT ?item ?instance_of
WHERE{
  {?item ?p "対象名称"@ja;
         wdt:P31 ?instance_of}
  UNION
  {?item skos:altLabel "対象名称"@ja.}
}
ORDER BY ?item
```

日本語 WordNet，Wikidata，業務データベースはいずれも収録語彙をメモリ内で管理するためのキャッシュインデックスを装備することで将来の問合せを減少させることに役立つ．キャッシュインデックスはメモリ上のインデックスを使うことで軽量に実装できる．実験システムでもキャッシュインデックス機構を備えているが，今回の実験ではキャッシュの効果は「検索回数削減」という本論文の主旨とはずれるためその効果を測定していない．

複合語候補は Wikidata キャッシュ→ WordNet キャッシュ→ WordNet（API）→ Wikidata（SPARQL API）の順で語彙確認を行う．Wikidata のほうが長い語が多く見つかり，WordNet より優先して検索すべきだが，できるかぎりメモリ内での処理で終わらせ，SPARQL ネットアクセスを最終手段とさせるための工夫である．

9.3 実験方法と手順

3 種類の実験を実施した．それぞれの実験方法と手順を以下に示す．

実験 1 同一文字種を分断する検索ノイズの問題（4.4 節で言及）が起きている箇所を計測．

N gram の形態素から，4.4 節で言及している「前後の文字と同一文字種で並んでいる」ものを一覧し，手作業で仕訳け作業を行った．具体的には「語として誤り」「一般用語」「WordNet や Wikidata を検索するのに適した固有

名詞など」である．語が同一文字で構成されているか，混合文字であるか，でも仕訳けしている．この実験によって8.4節の対策が有効であることを示す．

実験 2 Wikidataのみ，WordNetのみ，Wikidataを検索したあとWordNetを検索する，の3種類において検索回数および抽出量を計測．

各ステップを追加するごとに検索回数のみが減り，抽出数が減らないことで各対策が効果的であることを確認する．処理プログラムは，入力データを形態素解析処理したあとN gramの形態素を外側から組み合わせてWordNetおよびWikidataを検索するように作られている．解決策8.2節はあらかじめ含まれている．解決策8.1，8.3，8.4，8.5，8.6節を1つずつ追加し，そのつどの検索回数，抽出数すなわちWordNetおよびWikidataにヒットする数を計数している．入力ステートメント数，検索回数，抽出数はプラグイン内でカウントできる．

実験 3 各ステップの処理時間を計算量の指標として計測する．

各対策がインタラクションシステムの応答時間に影響がないことを確認するために計算量を計測する．ステップは実験2と同等とするが，ファイル読み込みのオーバヘッドの確認と形態素解析の計算量の確認も行うために「ファイル読み込みのみ」と「形態素解析のみ（プラグインしない）」を同時に計測している．CPU処理量のみを計測するため，キャッシュも含めWordNet，Wikidataの検索は行わない．実験データのテキストを全件読み込ませ，時間を計測する．繰り返すごとにばらつきがあるため，10回計測して平均を求めている．実験環境は以下の構成のマシンである．

- CPU：Intel Core i7-4702MQ 2.20 GHz
- メモリ：SODIM DDR3 1,600 MHz 16 GB

9.4 実験結果

9.3節で説明した3つの実験の結果を示す．

9.4.1 実験 1

表 4 に結果を示す．前後の同一文字種を分断する組み合わせのうち，ひらがな単一文字種と複合文字種について，誤り，一般語，複合語の数を表示している．

9.4.2 実験 2

表 5 に実験結果を示す．8章の解決策を順番に適用することにより，どのように効果があるか確認した．列「検索回数」はWikidataやWordNetを検索した回数であり「候補となった」形態素列の数である．列「抽出」は複数の形態素を統合した，つまり分かち書き結果を修正した個数であり，抽出された個数である．WordNetやWikidataに見つかった語の数ともいえる．列「削減%」は，N gram形態素すべてを検索したときの検索回数からの削減率である．最上段は1形態素を含めるすべての形態素組合せを取り出して検索をしたときの数字でありベースラインとなる．残りの5段がJMCTB法で行われている工夫を追加したときの検索回数，形態素の修正回数を示している．検索回数が減り，抽出数が減らないことが効率化を意味する．工夫を追加したとき抽出数の減少が少ないことが精度の高さを示す．つまり，検索回数が少なく，抽出数が多いことが要求された性能を発揮できていることを示す．

9.4.3 実験 3

表 6 に実験結果を，**図 4** に各ステップ追加後の処理時間のグラフを示す．「すべてのN gram形態素を外側から検索」の時点で，すべての形態素の文字種がどれであるかの確認が行われているため，その計算量も同時に増加する．

9.5 考察

表4の結果から，問題が起きているものは同一文字種で隣り合う文字種が同じものであり，誤りを効果的に抑止できることが確認できる．混合文字種の一般語76と複合語21は同一文字種を分断するが，混合文字種であるためにWordNetやLODで発見できた語（8.5節で言及）である．この実験結果から，「前後の同一文字種を分断する語」であ

表 4 同一文字種を分断する検索ノイズ
Table 4 Problem on character type boundary.

分類	ひらがな単一種	混合文字種
誤り	**25**	3
一般語	77	**76**
複合語	1	**21**

表 5 検索回数と抽出数
Table 5 Number of search and extraction.

	Wikidata			WordNet			Wikidata → WordNet		
適用した解決策	検索回数	削減%	抽出	検索回数	削減%	抽出	検索回数	削減%	抽出
すべてのN gram形態素を外から検索	17,948,311	-	226	17,845,685	-	294	17,834,293	-	495
8.1節 1形態素を除外	16,723,965	7	226	16,633,131	7	294	16,623,145	7	495
8.3節 記号を含む語を除外	1,308,405	93	218	1,302,935	93	198	1,301,675	93	391
8.4節 前後の文字種が同一のものを除外	286,026	98	176	284,767	98	134	284,722	98	288
8.5節 混合文字種のものを除外しない	699,495	96	**181**	695,759	96	**161**	694,737	96	**320**
8.6節 数字表現を除外	695,857	96	181	692,181	96	160	691,159	96	319

表 6 各ステップ追加後の処理時間
Table 6 Time of steps.

	処理時間	1 行あたり
ファイル読み込み	38	0
N gram 文字切り出し	1,418	13
Sudachi 処理のみ	5,497	50
すべての N gram 形態素を外から検索	10,688	102
8.1 節 1 形態素を除外	9,974	93
8.3 節 記号を含む語を除外	9,275	87
8.4 節 前後の文字種が同一のものを除外	9,392	91
8.5 節 混合文字種のものを除外しない	9,367	90
8.6 節 数字表現を除外	18,346	183
	ミリ秒	マイクロ秒

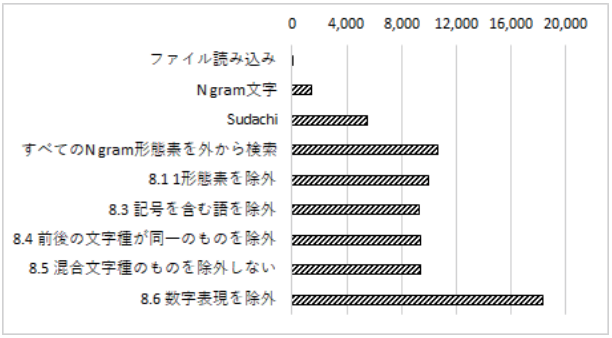

図 4 各ステップ追加後の処理時間
Fig. 4 Time of steps.

り「混合文字種でない」ものを検索候補とすることが，ノイズ除去に有効であることが分かる．

表 5 の結果では，検索回数が実験ごとに違う値になっている．これは，長い語検索で見つかったとき，より短い内側の語の検索をしないためであり，検索回数が少ないことは長い語を発見できていることを示している．

8.1 節で提案している「1 形態素を除外」して 2 gram の形態素から検索をすることで 1,800 万回弱から 1,700 万回弱程度まで検索回数を減らすことができる．

8.3 節で提案している「記号を含む語を除外」することで，検索回数をさらに 1,500 万回程度削減でき，130 万回程度まで激減させることができる．このコーパスでのデータでは 93%が削減できている．

8.4 節で提案している「前後の文字種が同一のものを除外」することにより形態素を分断してしまうような誤抽出が抑制できるようになる．エラー数も減り，検索回数も 28 万回程度まで激減し効果的にも見える．ここでの単独の削減率は 80%である．

しかし，抽出される箇所が激減してしまい正しいものも多く処理できないことが分かる．このため 8.5 節で提案している「混合文字種のものを除外しない」ことにより修正できる箇所の 1〜2 割程度を復活することが見て取れる．これにより「隣り合う文字が同一のものを除外し，文字種混合のものを復活させる」という 2 つの工夫によって，検索候補数を激減させ，精度の高い候補抽出ができており，この 2 つの工夫が JMCTB 法の性能に大きく貢献することが分かる．（表中，太文字下線の数値）

8.6 節で提案している「数字表現を除外」し，修正される箇所は変化がないため精度にはほとんど影響を与えることなく検索回数のみを減らすことができていることが分かる．ただし，対策前と対策後の発見された語のセットの差分の中には，厳密には正しい語だが検索対象から外れてしまい検索漏れとなってしまった語もデータから見つかる．たとえば，「できるかな」「どんだけ」「ひとつだけ」「いろいろあったけど」「えびせん」「ときめいて」「はじめて」「いいじゃない」「おにぎり」などがあげられる．本来は正しい言葉であり対象になってもよい．ひらがな中のひらがなであり見分けが難しいこと，形態素解析が細かく分断しすぎていること，などが理由としてあげられる．ある程度量が限られるため短期的な対策として，こういった語を形態素解析のユーザ辞書にあらかじめ入れておくことが考えられる．

今回抽出精度については触れていない．抽出された語の一覧を目で検査したところ，8.4 節で説明したようなひらがなで構成される誤抽出以外には不適切な抽出は発見できないからである．これは「語として認定できる」かつ「WordNet や Wikidata に見つかる」ことを示しており，正しい検索といえる．

ベースラインとなるすべての N gram 形態素を検索対象とする場合に対して，すべての解決策を適用することで 96%の検索回数を削減できた．

入力センテンス数が 112,235 行，データベース検索候補数が 691,159 回だったことから，1 センテンスあたり 6 回程度のデータベースや Web システムへの検索で済んでいる．Wikidata への検索回数の妥当性を検証するために，20 文字の入力文を前提に Wikidata への問合せにかかる時間を調べた．だたし，Wikdiata エンドポイントへのネットワーク距離，回線性能，込み具合，検索語が結合されている具合などによって速度は変動するため，参考程度としたい．20 文字のテキストを 2 gram 以上の N gram 文字にすると 190 個の文字列が作られる．190 個の文字列を SPARQL にて Wikidata 検索をかけると，10 回実施した場合の平均で 66 秒かかった．実験 2 の結果を参考に 6 回の検索を行うと，10 回実施した場合の平均で 1.4 秒であった．音声対話型インタフェースやテキスト対話のチャットシステムなどを作る場合に LOD 検索時間が 1.4 秒程度であれば，他の処理と合わせて数秒での返答を実現できる可能性があり，妥当な数値といえる．

表 6 および図 4 により各ステップごとの計算量が比較できる．Sudachi の形態素解析のみに比べて，N gram 形態素組み立ておよび文字種確認を行うと倍程度時間がかかる．2 gram 以上にし，記号を含む語を除外することで

処理量が少しずつ減るが，前後の文字種が同一のものを除外し，混合文字種のものを除外しない場合にはあまり処理量は変わらない．これは，WordNet や Wikidata の検索を行っていないため「見つかった語より内側を検索しない」という効果がないからだと考えられる．数字表現を除外する処理には多くの時間が費やされる．今回の実験システムでは 25 種類の数値表現を正規表現によってパターンマッチングすることで確認し除外しているため多くの計算量を使っていると考えられ，今後の課題としたい．この実験では全データ 112,235 件を処理する時間全体を計測して平均値である．1 行あたりの平均処理時間はすべての対策を施した場合の最大で 138 マイクロ秒であり，ヒトとのインタラクションの応答時間に大きな影響を及ぼす時間ではないと考えられる．

10. まとめ

音声対話型インタフェース，テキストチャットボットなど，リアルタイムにテキストを処理するシステムにおける，データベースや Web サービスを利用した語を抽出するアーキテクチャおよび検索回数削減手法を提案した．検索回数削減手法として，形態素解析のあと，長い N gram 形態素列から優先的に検索し，2 gram 以上の形態素列を，同一文字種を分断しないように取り出し，混合文字種のものを除外せず，数字表現を除去してデータベース検索を行うという手法「日本語形態素文字種境界法：Japanese Morpheme Character Type Boundery（JMCTB）Method」を提案した．実験システムを実装し効果を確認したところ，全形態素を N gram 検索するときに比べて検索回数 96%を削減でき，4%程度の検索回数にて，全形態素検索をしたときとほぼ同じ精度を発揮できることが確認できた．同実験によって，形態素を分断する問題や，ひらがなを分断するなどの間違った抽出を行う問題も防止できることが確認できた．

11. 今後の展望

今回の提案にて「数字表現を候補から除外」することを提案しているが，固有表現として定義されている「時間表現」「金額表現」「割合表現」にも同様のことがいえる．追加的に除外する方向で実装を進めている．

今後，テキストから JMCTB 法によって「WordNet に掲載されている」「Wikidata に掲載されている」「業務データに掲載されている」ことが抽出できることにより，語の位置づけをより詳細に処理するために，WordNet，Wikidata，業務データベースを活用できると考えている．

参考文献

[1] Thompson, K.: Programming techniques: Regular expression search algorithm, *Comm. ACM*, Vol.11, No.6, pp.419–422 (1968).

[2] Wada, K.：N-gram の作り方，入手先 ⟨https://qiita.com/kazmaw/items/4df328cba6429ec210fb⟩.

[3] Berger, A.L., Pietra, S.A.D. and Pietra, V.J.D.: A Maximum Entropy Approach to Natural Language Modeling, *Computational Linguistics*, Vol.22, No.1 (1996).

[4] Matsumoto, Y. et al.: Japanese morphological analysis system ChaSen version 2.0 manual, NAIST Techinical Report (1999).

[5] Kudo, T.: MeCab: Yet Another Part-of-Speech and Morphological Analyzer (2005) (online), available from ⟨http://mecab.sourceforge.net/⟩.

[6] 森田　一，黒橋禎夫：RNN 言語モデルを用いた日本語形態素解析の実用化，第 78 回全国大会講演論文集，Vol.2016, No.1, pp.13–14 (2016)（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/170000163288/⟩.

[7] Apache: lucene-gosen, available from ⟨https://github.com/lucene-gosen/lucene-gosen⟩.

[8] atilika: Kuromoji, available from ⟨https://www.atilika.org/⟩.

[9] Takaoka, K., Hisamoto, S., Kawahara, N., Sakamoto, M., Uchida, Y. and Matsumoto, Y.: Sudachi: A Japanese Tokenizer for Business, *Proc. 11th International Conference on Language Resources and Evaluation* (*LREC 2018*), Calzolari, N. (chair), Choukri, K., Cieri, C., Declerck, T., Goggi, S., Hasida, K., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. and Tokunaga, T. (Eds.), European Language Resources Association (ELRA) (2018).

[10] NAIST: Japanese Dictionary, available from ⟨https://osdn.jp/projects/naist-jdic/⟩.

[11] Den, Y., Yamada, A., Uchimoto, K., Koiso, H. and Ogiso, T.: The development of a multi-purpose electronic dictionary for morphological analyzers, Report from the Electronic Dictionary Group at the "Japanese Corpus" General Meeting, pp.21–26 (2006).

[12] Sato, T.: Neologism dictionary based on the language resources on the web for MeCab (2015), available from ⟨https://github.com/neologd/mecab-ipadic-neologd⟩.

[13] 工藤　拓：単語の追加方法，入手先 ⟨https://taku910.github.io/mecab/dic.html⟩.

[14] Chinchor, N. and Robinson, P.: MUC-7 named entity task definition, *Proc. 7th Conference on Message Understanding*, Vol.29, pp.1–21 (1997).

[15] Sekine, S. and Isahara, H.: IREX: IR & IE Evaluation Project in Japanese., *Proc. 13th International Conference on Language Resources and Evaluation* (*LREC*), pp.1977–1980, Citeseer (2000).

[16] 矢野　憲，伊藤　薫，若宮翔子，荒牧英治：深層学習による医療テキストからの固有表現抽出器の開発とその性能評価，人工知能学会全国大会論文集第 31 回全国大会，一般社団法人人工知能学会，pp.2J2OS16a4–2J2OS16a4 (2017).

[17] Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S. and Hakkani-Tur, D.: Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines, arXiv preprint arXiv:1907.01669 (2019).

[18] Hu, A., Dou, Z., Nie, J.-Y. and Wen, J.-R.: Leveraging Multi-token Entities in Document-level Named Entity Recognition, *Proc. Association for the Advancement of Artificial Intelligence* (*AAAI*), pp.7961–7968 (2020).

[19] Schuster, M. and Paliwal, K.K.: Bidirectional recurrent neural networks, *IEEE Trans. Signal Processing*, Vol.45, No.11, pp.2673–2681 (1997).

[20] Isahara, H., Bond, F., Uchimoto, K., Utiyama, M. and Kanzaki, K.: Development of the japanese wordnet

(2008).
[21] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J. and Vrandečić, D.: Introducing Wikidata to the linked data web, *Proc. International Semantic Web Conference* (*ISWC*), pp.50–65, Springer (2014).
[22] 川口久光ほか：N gram 型大規模全文検索方式の開発：文字種適応型 N gram インデクス方式，全国大会講演論文集データベースとメディア，pp.237–238 (1996).
[23] 下畑光夫，杉尾俊之ほか：文字種切り出しと複合語分解によるキーワード抽出，情報処理学会研究報告自然言語処理（NL），Vol.1997, No.69 (1997-NL-120), pp.83–88 (1997).
[24] 小木曽智信：近代語テキストの形態素解析，国立国語研究所近代語コーパス設計のための文献言語研究成果報告書，pp.83–92 (2012).
[25] 米持幸寿，大場みち子：多段自然言語処理における NLP，シソーラス，オントロジー辞書データ統合の提案，人工知能学会研究会資料，Vol.47, No.1, pp.1–7 (2019).
[26] 宇佐美まゆみ：BTSJ 日本語自然会話コーパス（トランスクリプト・音声）2018 年版，国立国語研究所，機関拠点型基幹研究プロジェクト「日本語学習者のコミュニケーションの多角的解明」，サブ・プロジェクト「日本語学習者の日本語使用の解明」（リーダー：宇佐美まゆみ）(2018).
[27] 嶋　英樹：Java Wrapper for Japanese WordNet，入手先 ⟨http://www.cs.cmu.edu/~hideki/software/jawjaw/index.html⟩.
[28] Jena, A.: Semantic web framework for Java (2007).

米持 幸寿

1987 年群馬工業高等専門学校機械工学科卒業，日本アイ・ビー・エム入社，ソフトウェアエンジニアとして製品サポート，セリング，マーケティング，基礎研究などに従事．2012 年から国立はこだて未来大学講師．2015 年ホンダ・リサーチ・インスティチュート・ジャパン入社，リエンジニアリング・研究戦略室長．2020 年 Pandrbox 創業．人工知能学会，言語処理学会各会員．

大場 みち子　（正会員）

1982 年日立製作所入社．知識工学応用の研究，ミドルウェアの開発・マーケティングに従事．2001 年大阪大学大学院工学研究科博士後期課程修了．博士（工学）．2010 年公立はこだて未来大学教授．知的行動の記録と分析，ドキュメンテーション，ソフトウェア工学等の研究に従事．学術会議連携会員，電気学会，ソフトウェア科学会等の各会員．