博士論文

音声対話インターフェースにおける自然言語処理の大規模語彙獲得手法

公立はこだて未来大学大学院
システム情報科学研究科
システム情報科学専攻

米持 幸寿

Doctoral Thesis

Methodologies of Integrating Huge Vocabulary on Natural Language Processing of Voice Interactive Interface

by

Yukihisa Yonemochi

Graduate School of Systems Information Science
Future University Hakodate

January 2022

概要

本論文は，2019 年から現在まで公立はこだて未来大学大学院在学中に行ってきた音声対話インターフェースの語彙獲得に関する研究成果である．

音声アシスタントを具備するスマートフォンや家庭用スマートスピーカーの普及に観られるように，音声対話インターフェースが急激に一般的になりつつある．音声対話インターフェースとは音声を使ってアプリケーションプログラムを利用することができるようにする仕組みである．様々な技術の構成が考えられるが，本研究では音声認識によりテキストデータを作り言語処理する構成を対象としており，その言語処理が題材である．アプリケーションプログラムが読み取る必要のある語を入力テキストデータ中から見つけ出す「語の抽出処理」が一回の対話に対する最初の処理である．その処理ではあらかじめ語彙を保持しているが，その語彙に含まれない語を話しかけられた時の，語の抽出処理エラーを削減することを目的として，外部の語彙を活用する際の問題解決手法を提案する．

自然言語処理分野において，テキストデータから属性を考慮して語を抽出する処理を「固有表現抽出」と呼び，音声対話インターフェースに用いることができる．固有表現抽出は自然言語処理として長年研究されてきているが，従来研究ではあまり考慮されて来なかった音声対話インターフェースならではの特別な要件がある．幅広い語彙を獲得すること，常時継続稼働すること，新語を語彙に即時反映すること，応答性能を確保することである．

音声対話インターフェースでは，コマンドプロンプトや GUI などの従来インターフェースと比較すると入力内容の自由度が高く，幅広い語彙を獲得する必要がある．語の抽出に深層学習を使う場合，訓練データに含まれない語の抽出精度を向上させる一般的な方法として再学習か追加学習が考えられるが，システムの再起動が避けられない．従来研究では大量文書を処理することが多く，一旦処理を行ったあと抽出に失敗した箇所をリスト化し，語彙として追加して再度処理する方法がとられることがある．しかし，音声対話インターフェースの処理において，一つの入力テキストに

対してパラメーターを変更して複数回処理することは認められず，語彙は発生後即時に取り入れられることが求められる．ヒトが音声対話を行う際は，特別にレスポンスタイムに厳しい要求があるため，それを考慮する必要もある．

上述の要件を考慮しつつヒトが使う可能性のある幅広い語彙を獲得するため，1. 複数の言語資源を統合して使うことの有効性を示すこと，2. 深層学習において再学習なしに語彙を利用する手法を提案すること，3. レスポンスタイムを考慮しつつ外部検索を行う手法を提案すること，の三つが本論文の研究課題である．

複数の言語資源を統合して使うことの有効性を示す．音声対話インターフェースにおける語の抽出では「これは映画題名」「これは魚の名前」などと属性を意識する必要がある．これらの情報を保持する言語資源としてシソーラスやオントロジー辞書が必要になることがある．完全無欠な言語資源を作ることは困難であり，補完関係になることが予測されることから，これらの言語資源を同時に使うことでより多くの語彙が獲得できると仮説できる．世の中には多種多様な言語資源が存在するが，学術研究に利用しやすいものとして日本語 WordNet，DBPedia，Wikidata を取り上げる．両語彙セットを統合することで語彙が補完関係にあることを確認し，両語彙セットを同時に利用する価値を二つの検証で示す．一つ目は，対話コーパスの談話テキストを形態素解析によって分解し，隣り合う語の組み合わせが日本語 WordNet や DBPedia にどれくらい見つかるかを調べる．これにより，形態素解析だけでは語の抽出には不十分で，複合語を取り扱う必要があることを示す．二つ目は，日本語 WordNet および Wikidata の概念構造を利用して，食材，魚介類などの名称を双方で検索し，相互補完性があることを明らかにする．

深層学習固有表現抽出器で再学習なしに外部語彙を利用する手法として，入力に語彙特徴量を追加することを提案し，その有効性を示す．訓練データに含まれない語彙を追加する手法として追加学習や再学習が一般的であるが，大きな計算コストと再起動が必要である．固有表現抽出器には周辺語を利用して未知語を抽出する能力もあるが精度は低い．本研究では，シ

ステムに対して外部から語彙を与えることで問題の軽減を図る方法を提案する．最新の技術として Google 社が発表している言語処理モデル BERT を取り上げる．深層学習を利用する未知語としての語の抽出において，訓練データに含まれる語より長い語の抽出エラーが多いことに着目し，実際にその現象が発生することを確認する．現象が発生している状況に対して語彙情報を与えることで問題が軽減することを確認するために，入力に語彙情報としてバイナリ特徴量を追加し，効果を明らかにする．これらの状況が現実社会で起こり得ることを示すため，Wikidata から著作物名を収集し，その長さが年々増加していることを確認し，本提案で示すような対策が現実に必要な場合があることを示す．特に日本の日本語のマンガとアニメの題名は，米国の英語のものよりも優位に長いことを示す．

レスポンスタイムを考慮しつつ外部検索を行う手法を提案し，有効性を示す．商品データベースなどは日々更新されるため，その度に深層学習の抽出器を再訓練することは現実的でない．音声対話インターフェースを提供するシステムから外部にあるデータベースを即時検索して語彙を利用する場合を考える．入力文からデータベースにある語を見つけ出すには，非常に多くのクエリを実行する必要があり，レスポンスタイム要求に対して大きな障壁となる．即時検索する際に発生するクエリの回数を削減するために，日本語形態素文字種境界（JMCTB）法を提案する．音声対話インターフェースを対象とするため，話し言葉コーパスを使って実験を行い，効果があることを示す．

三つの研究成果を説明した後，それらを総合的に活用する音声対話インターフェースのシステムアーキテクチャーを提案する．外部の語彙データや語彙に使える外部システムをオンラインで結合し，音声対話インターフェースの要件を満たしつつ語彙を増強することができるようになることを説明する．提案するアーキテクチャーにより，あらかじめシステム語彙に含まれていない外部語彙を直接活用でき，新語にも即時対応が可能であることを示す．これにより，音声対話インターフェースが外部語彙を獲得する技術に大きく寄与するものであると考える．

Abstract

This paper is the result of my research on vocabulary acquisition for voice interactive interface, which I have been conducting while I was a graduate student at Future University Hakodate, Japan, from 2019 to present.

Voice interactive interfaces are becoming more and more common, as seen in the widespread use of smartphones equipped with voice assistants and home smart speakers. A voice interactive interface is a mechanism that allows users to use application programs using their voice. There are various possible configurations of the technology. In this research, the subject is a configuration that creates text data through speech recognition and performs language processing. The word extraction process, in which the application program finds the words it needs to read in the input text data, is the first process for a single interaction. This is the first process for each dialogue. This paper proposes a problem-solving method for using an external vocabulary to reduce errors in word extraction when a word that is not part of the pre-integrated vocabulary is spoken.

In the field of natural language processing, the process of extracting words from text data by considering their attributes is called named entity recognition (NER), and it can be utilized on voice interactive interface. NER has been studied for many years in natural language processing, but there are special requirements unique to voice interactive interface that have not been considered in previous research: a wide range of vocabulary should be supported, the system should run constantly, new words should be reflected in the vocabulary immediately, and response performance should be ensured. Compared to conventional interfaces such as command prompts and GUIs, voice interactive interface require a higher degree of freedom in input content and support for

a wider range of vocabulary. When deep learning is used for word extraction, retraining or additive training is considered as a common method to improve the extraction accuracy of words not included in the original training data, but system restart is inevitable. In conventional research, a large number of documents are often processed, and after processing, the parts that failed to be extracted are sometimes listed and added to the vocabulary for further processing. However, in the processing of a voice interactive interface, it is not allowed to process a single input text multiple times with different parameters, and vocabulary must be incorporated immediately after it appeared. Human speech interaction has special stringent requirements on its response time, which also need to be taken into account.

In order to acquire a wide range of vocabulary that humans may use while taking the above requirements into account, there are three research topics: 1) to show the effectiveness of integrating multiple language resources, 2) to propose a method to use vocabulary without retraining in deep learning, and 3) to propose a method to perform external search while considering response time. These are the three research topics of this paper.

The effectiveness of integrating multiple language resources is demonstrated. When extracting words in a voice interactive interface, it is necessary to be aware of attributes such as "this is a movie title" or "this is the name of a fish." A thesaurus or ontology dictionary may be needed as a language resource to hold this information. Since it is difficult to create a complete set of language resources, and since they are expected to be complementary, it can be hypothesized that a larger vocabulary can be acquired by using these resources simultaneously. There are a wide variety of language resources in the world, but Japanese WordNet, DBPedia, and Wikidata are utilized as those that are easy to use for academic research. By integrating both lexical sets, it is confirmed that the

lexicons are complementary, and shown the value of using both lexical sets at the same time in two verifications. The first is to use morphological analysis to decompose the discourse text in a dialogue corpus to see how many combinations of adjacent words can be found in WordNet and DBPedia. This shows that morphological analysis alone is not sufficient for word extraction and that compound words need to be handled. Second, using the conceptual structure of Japanese WordNet and Wikidata to search for names of meals and seafoods, etc. in both data set, and find that they are complementary.

As a method to use an external lexicon without retraining in a deep learning NER, this paper propose to add lexical features to the input and show its effectiveness. Additive training and retraining are common methods to add vocabulary that is not included in the training data, but they require large computational cost and restart. NER also have the ability to extract unknown words by using surrounding words, but the accuracy is low. This study proposes a method to mitigate the problem by providing an external vocabulary to the system. BERT, a language processing model released by Google, is utilized on this paper. In extracting words as unknowns using deep learning, focusing on the fact that there are many errors in extracting words that are longer than those in the training data, confirm that this phenomenon occurs in practice. In order to confirm that providing lexical information to the situation in which the phenomenon is occurring reduces the problem, binary features as lexical information is added to the input and clarify the effect. To show that these situations can occur in the real world, the titles work are collected from Wikidata and confirmed that their length is increasing year by year, indicating that the countermeasures shown in this proposal may be necessary in reality. In particular, it shows that Japanese manga and anime titles in Japanese are predominantly longer than their English counterparts in the US.

We propose a method for external search while considering response time, and show its effectiveness. Since product databases are updated on a daily basis, it is not practical to retrain the deep learning extractor for each update. Consider the case where a system that provides a voice interactive interface immediately searches an external database for vocabulary. Finding a word in the database from an input sentence requires executing a very large number of queries, which is a major barrier to the response time requirement. This paper proposes the Japanese Morphological Character Type Boundary (JMCTB) method to reduce the number of queries that occur during immediate retrieval. As the target is a voice interactive interface, experiments using a spoken corpus is conducted to show that it is effective.

After explaining the three research results, this paper proposes a system architecture for a voice interactive interface that comprehensively utilizes them. It illustrates how external lexical data and external systems that can be used for the lexicon can be combined online to augment the lexicon while meeting the requirements of a voice interactive interface. That the proposed architecture can directly utilize external vocabularies that are not included in the system vocabulary in advance, and can immediately respond to new words are shown. I believe that this will greatly contribute to the technology of acquiring external vocabulary for voice interactive interface.

目次

第 1 章　序論 3

1.1　音声対話インターフェースの市場動向 3
1.2　想定する音声対話インターフェース 5
1.3　音声対話インターフェースにおける問題点 8
1.4　音声対話における要件 8
1.5　研究目的と課題 10
1.6　本研究の方針 12
1.7　本論文の構成 13

第 2 章　関連技術 15

2.1　音声対話における入力テキストの解釈 15
2.2　自然言語処理技術の適用 19
2.3　深層学習による固有表現抽出 26

第 3 章　関連研究 29

3.1　大語彙言語資源 29
3.2　語彙増強アプローチ 31
3.3　各研究分野と本研究の位置づけ 34

第 4 章　複数大語彙資源利用価値の検証 36

4.1　はじめに 36
4.2　大語彙言語資源における複合語 36
4.3　日本語 WordNet と Wikidata 統合の価値検証 41
4.4　まとめ 46

第 5 章　深層学習 NER における長い未知語問題 47
5.1 はじめに 47
5.2 UnknownLoger 問題の検証概要 48
5.3 問題の定義 49
5.4 語彙特徴量の追加手法 51
5.5 検証 1 および 2 の実験内容 52
5.6 検証 1 および 2 の結果 56
5.7 検証 3 作品題名長さの分布 58
5.8 まとめ 60

第 6 章　日本語形態素文字種境界 (JMCTB) 法の提案 62
6.1 はじめに 62
6.2 固有名詞抽出手法の現状 63
6.3 課題に対する解決策 71
6.4 検索回数抑制手法 73
6.5 実験 78
6.6 考察 87
6.7 まとめ 89

第 7 章　統合アーキテクチャー 91
7.1 はじめに 91
7.2 アーキテクチャー 92
7.3 期待される効果 95
7.4 まとめ 97

第 8 章　結論 99
8.1 本論文のまとめ 99
8.2 今後の研究課題と展望 100

謝辞 102

業績一覧 103

参考文献 104

序論

第 1 章

本章では，本論文が研究対象とする音声対話インターフェースの市場動向と概要，語彙に関連する問題を説明する．問題の解決アプローチの背景となる音声対話特有の要件を説明し，要件を満たしつつ語彙に関連する問題を解決する手法の研究内容の概要，および本論文の構成について述べる．

1.1　音声対話インターフェースの市場動向

音声対話インターフェースとは，ヒトが機械に向かって話しかけ，機械がその声をマイクで拾い解釈してなんらかの処理を行い，スピーカーからヒトの音声を発することでインタラクション操作を行うしかけである．近年ではスマートフォンに搭載されている音声アシスタントや，家庭向けスマートスピーカーなどで知られている．

クラウド技術や深層学習技術を背景に，音声認識技術の精度が急激に向上し，音声アシスタントやスマートスピーカーなど音声対話型のインターフェースの普及がめざましい．**図 1.1** は，対話型 AI システム市場における市場予測を示したデータ [77] である．2018 年発表のこの資料によれば，2022 年の同市場規模は 132 億円になると予測されている．

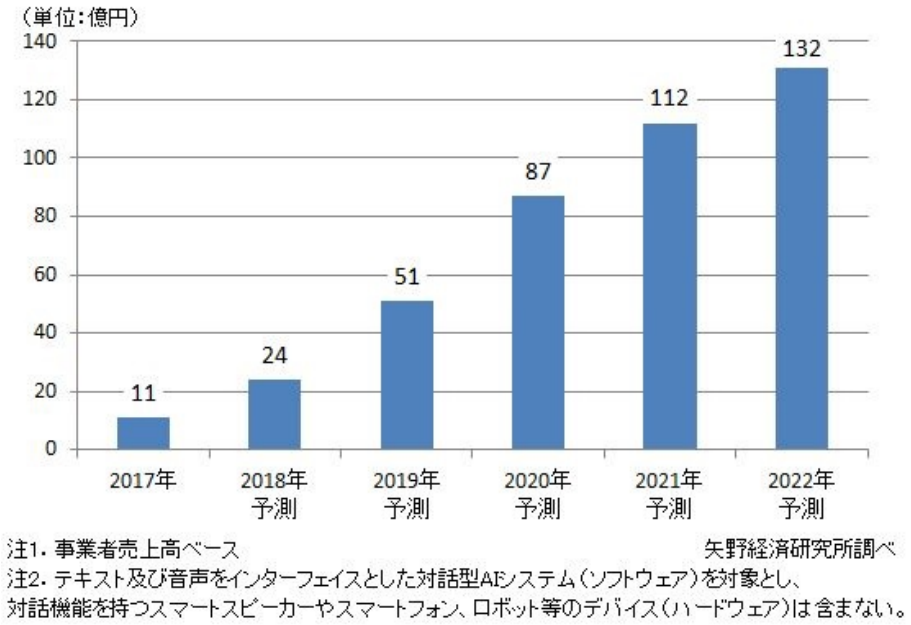

図 1.1　対話型 AI システム市場に関する調査（矢野経済研究所）

図 1.2 が示す音声認識システム市場概況 [61] によれば，2023 年度の音声認識システム市場は約 1,000 億円を超えるとされている．

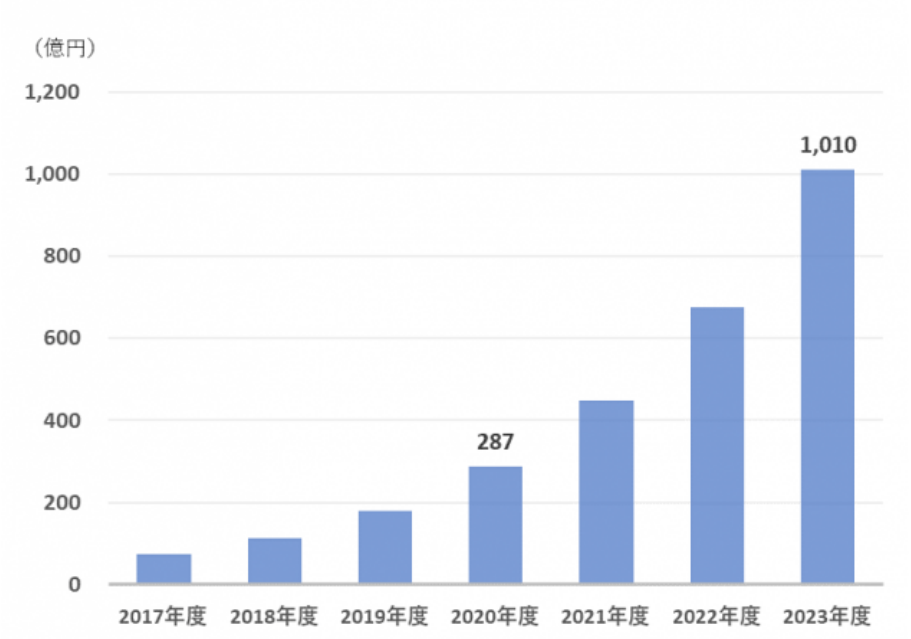

図 1.2　音声認識システム市場概況（MDB Digital Research）

家庭用照明機器もスマートスピーカー対応をうたうものが増え，**図 1.3** が示す予測 [76] によれば，2025 年の国内市場は 200 億円（12.5 倍）で，利便性の向上に加え，所有率の上昇が期待され市場拡大が予測されている．

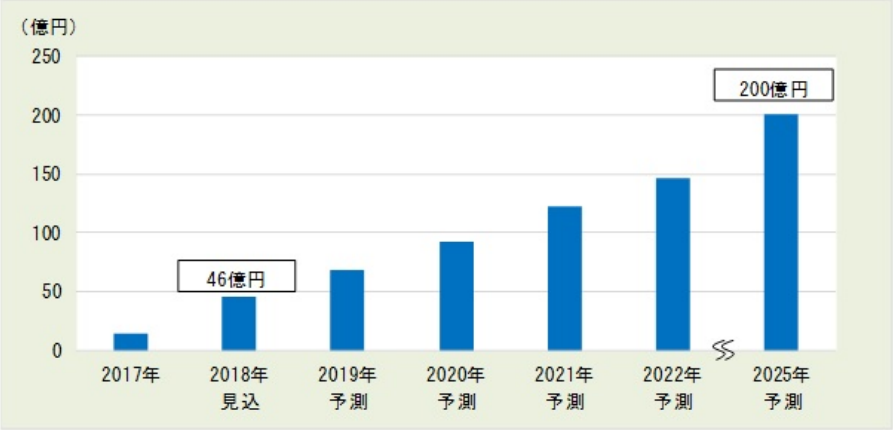

図 1.3　スマートスピーカーやスマートホーム対応照明の市場（富士経済研究所）

これらのデータから，市場のスマートスピーカーへの期待が大きい事を伺い知ることができ，音声インターフェースは市場にて活況と言え，それにまつわる技術の研究は価値が高いと考えることができる．

1.2　想定する音声対話インターフェース

本論文では，音声対話インターフェースの自然言語処理において，語彙不足による処理エラーの軽減を目的に，複数の語彙資源を統合して利用する手法を提案する．音声対話インターフェースを対象にしており，後方にアプリケーションプログラムが存在するものとする．音声対話インターフェースとは，音声すなわちヒトの声を入力してアプリケーションプログラムに引継ぎ，アプリケーションプログラムが出力したテキストを音声で出力する，取り次ぎを行う部分のことを指す．音声対話インターフェースには音響機器，音声認識ソフトウェア，自然言語処理，発話生成，音声合成など様々なコンポーネントが含まれるが，その中でも特に自然言語処理を対象とする．

本論文で想定する音声対話インターフェースを利用するシステムの問題は，チャットボット，ロボット，スマートスピーカー，スマートフォンなどに共通の問題である．音声対話を実現する方法には非常に多くの方式や選択肢があり，一般論として示すことは困難

である．たとえば，次のようないくつかの方式が考えられる．

1. 音声データをテキストに変換してから解釈し，処理プログラムを実行し，文章を得て音声で出力する．
2. 音声データをテキストに変換して解釈せずに入力し，深層学習などで直接出力テキストを得る
3. 音声データをテキストに変換せずそのまま入力し，深層学習などを使い直接音声データ出力を得る

本論文ではこれらの選択肢のうち，音声対話インターフェースは 1. の方式「音声データをテキストに変換してから解釈し，処理プログラムを実行し，文章を得て音声で出力する．」を使うものと定義する．**図 1.4** に，本論文で想定している音声対話インターフェースのアーキテクチャーを示す．「音声認識」「言語処理」「アプリケーション（処理)」「発

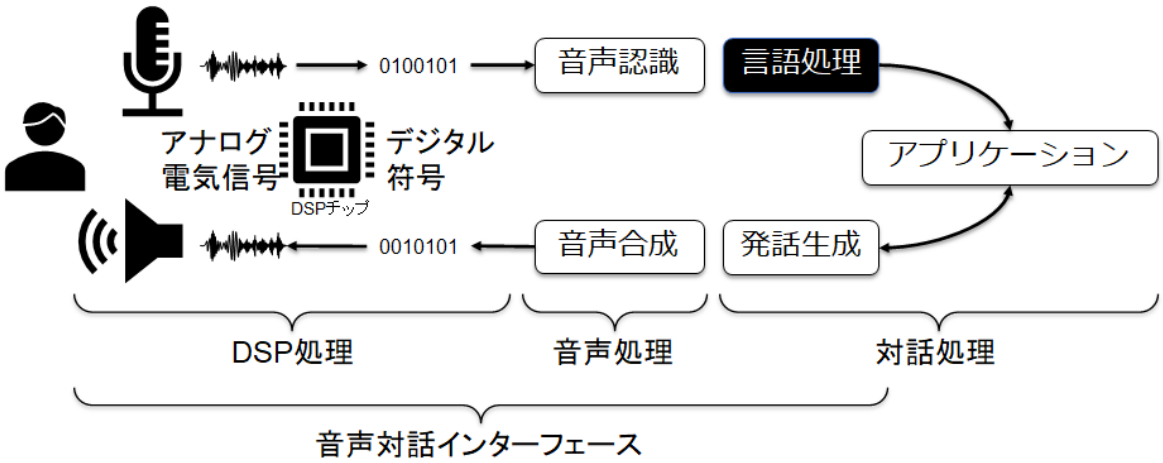

図 1.4　音声対話インターフェース・アーキテクチャー

話生成」「音声合成」の順で処理が進む．ユーザーであるヒト（図中，左）が言葉を機器に話しかけると，マイク，DSP 処理，音声認識などを経てテキストが作られる．

音声対話インターフェースの最初のステップは音声認識である．ヒトが機器に話しかけ，テキストに変換された後，テキスト処理や言語処理を行うため，前段の音声認識の精度が高いことが後段の処理に影響する．音声をマイクロフォンで電気信号にしたあと，Analog to Digital（AD）変換を行い，波形をサンプリングして符号化した Pulse Code Modulation(PCM）データとしたものを「音声信号データ」と呼ぶ．音声信号データをテキストデータに変換する処理を「音声認識」と呼ぶ．音声認識はスマートフォンの API

(Android API[12], iOS API[5]) やクラウド API (IBM[26], Google[22], Microsoft[37]) など様々なものが容易に利用可能で，有償で高機能な市販の音声認識製品も多数存在する．今日の音声認識技術は，深層学習を取り入れることで急激に性能が向上した．音声認識では大語彙を用いることで処理性能が向上することが広く知られている．大規模な事前にラベルを付与されたラベル付きコーパスを使った深層学習か，大規模語彙を投入した研究が多い．たとえば，咽喉マイクを用いた大語彙音声認識研究 [79] などがある．本論文の研究は，音声認識後のテキストの処理に関する研究であり，音声認識の精度に対しての語彙増強は対象外とする．

音声認識から出力されるテキストを言語処理することで呼び出すアプリケーションプログラムが決定され，アプリケーションが動作する．その後，発話文がテキストとして生成され，音声合成，DSP 処理を経てスピーカーから返答が発話される．入力テキストの言語処理（図中，黒塗り部）の語彙が本論文の主題であり，それ以外の箇所はテーマ外であるため詳細な説明は省略する．

図 1.5 はこの想定する音声対話インターフェースを利用するアプリケーションシステムの言語処理と語彙の例を示す．

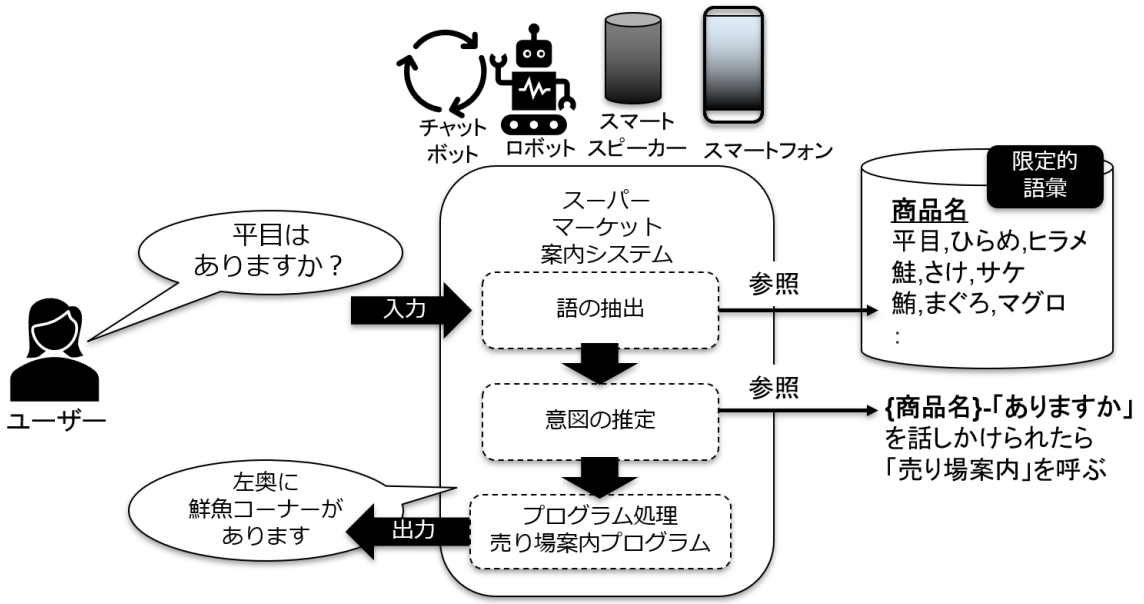

図 1.5　アプリケーションシステムの言語処理と語彙の例

ここでは例としてスーパーマーケットの案内システムを挙げ，言語処理とアプリケーション処理に注目して説明する．ユーザーである顧客が「平目はありますか」と尋ねたと

き，まず語の抽出を行う．語の抽出には，商品名のデータベースなどを語彙として使うことができる．平目が見つかると，次に意図の推定を行う．たとえば「<商品名>-ありますか」などといったパターンなどで実装することができる．語が抽出でき，意図が推定できると，処理プログラムに「平目」を引数として渡すことで，「左奥に鮮魚コーナーがあります」といった返答ができるものとする．

1.3　音声対話インターフェースにおける問題点

音声対話インターフェースにおける語彙不足による処理エラーの例を説明する．前節で説明したような音声対話インターフェースを持つ案内システムに「うしのしたはありますか？」と尋ねる場合を考える．案内システムは，「店舗奥の右側，精肉売り場にあります」と回答するかもしれない．「うし」という部分を「牛」，あるいは「うしのした」を「牛たん」として処理した結果であろう．しかし，この回答は必ずしも正しい選択ではない可能性がある．「うしのした」という言葉には，ある地方では「舌平目」の意味があり，植物にも「ウシノシタ草」というのがある．それらの語彙を知っていれば「舌平目は店舗奥左側，鮮魚売場に，ウシノシタ草は取り扱いがございません」といった案内のほうが適切かもしれない．

このように，音声対話の処理が意図しない動作をする場合のほとんどは，ヒトから機械に話しかけたとき，その意図の読みとりに失敗していることが起因している．意図読み取りには語の抽出が行われるが，その原因の大半は語彙の問題 [27] である．ヒトの話しかけの意図の読み取りエラーのうち，語彙不足に由来するエラーに対応するのが本研究のスコープである．語彙不足に対応するために語彙を増強する事を「語彙獲得」という．本研究では，音声対話インターフェースを具備するコンピューターシステムにおいて，語彙獲得をすることで語彙不足問題の軽減を目的とする．

1.4　音声対話における要件

音声対話インターフェースではインタラクションの質が異なるため，音声対話インタフェース特有の要件がいくつかあり，それらから研究課題を明らかにする．

1.4.1　幅広い語彙の獲得

古くからコンピューターシステムにはコマンドプロンプトという対話型機能がある．通常，ヒトから始まり，コンピューターが応答するタイプの対話である．対話型インターフェースであるため音声対話インターフェースと似た仕組と考えられるが，実際は異なる．コマンドプロンプトではシェルコマンドといったコンピューターの人工言語を使ったインタラクションであり機械が主導的に対話を進められる．コマンドプロンプトを使うヒトが主にコンピューターエンジニアであることが背景として考えられる．ヒトが話す言葉，すなわち自然言語と比較すれば表現方法は限定的で，内容の解析がより限定的である．コマンドはつづりや順序が厳密であり柔軟性はとても限定的である．つづりや順序が異なればエラーとなり，間違えたヒトに責任がある．

その後登場したグラフィカルユーザーインターフェース（GUI），ウィンドウシステム，Web システムでは，ヒトの入力の多くは「システムが提示したものの選択」，すなわち「ヒトによる応答」に近いものである．コンピューターシステムの都合でインタラクションを進めることができる．選択できるものは限定的で，システムが想定する以上の入力が行われることは通常ない．

コマンドプロンプト，GUI，ウィンドウシステム，Web インターフェースなどと比較して，音声対話やテキストチャットではヒトの言葉を使うため，ヒト手動で対話が行われる．ヒトの言葉は語彙がずっと広く文法も柔軟であり，システムが期待している以上のことをヒトが話しかけることを想定しなくてはならず，入力内容を自然言語として解釈する必要がある．

1.4.2　常時継続稼働

無人の応答システムなどの場合，音声対話システムは常時継続稼働を継続することが求められる．語を追加するなどの目的で，辞書ファイルを更新して再読み込みしたり，深層学習モデルを再学習して再起動したりすることは本来好ましくない．

1.4.3　新語の即時反映

業務システムの商品データベースなどは常に更新されており，最新の語を取り扱うための方法が必要な場合がある．「1,000 万語を学習した AI」という謳い文句を見ることがあるが，学習データ作成のたった 1 分後に発生した題名などの名称は学習されておらず，処理できない．

今日，語の抽出に使われる技術として深層学習技術を利用することが最新研究とされている．深層学習で語彙を増やすためには再学習や追加学習をすることが一般的であるが，新語は即時反映されない．再起動を回避して常時継続稼働するシステムに新語彙や外部語彙を反映させる方法が必要である．

1.4.4　応答性能

応答タイミングを考慮した音声対話システムの研究とその評価 [71] によれば，応答は 0.7 秒程度がよいとされている．対話ロボットの反応時間は，およそ 1.0〜2.0 秒の間になんらかの回答をヒトの要求に対して発話すべき，という研究もある [70]．これは，音声対話インターフェースがバックエンドのデータベースやインターネットに置かれている Linked Open Data(LOD) を検索するような場合に制約となる．話速はユーザーの速度に併せることが心地よい対話となる [64]．このことから，話速の早い話者のためには早く返答する必要がある．システムの作り方として考えれば，遅く反応したい場合は応答をディレイすればよいため問題とはならず，早く反応しなくてはならない要件は制約と考えることができる．

1.5　研究目的と課題

ここまで説明したように，音声対話インターフェースには語彙不足によるエラーが発生する問題があり，語彙不足を軽減する手法が必要である．そして，大量文書処理タスクにはない，音声対話インターフェース特有の要件がある．音声対話インターフェースの要件を満たしつつ，語彙不足軽減を最終目的に語彙獲得を行う手法を提案することが本論文の目的である．目的を整理すると音声対話インターフェースが以下のことを達成することである．

1. ヒトが幅広く知っている大規模な語彙を獲得する
2. 常時継続稼働し再起動を避け大規模な語彙を獲得する
3. 常時継続稼働し再起動を避け新語を獲得する
4. 適切な時間で応答する

音声対話インターフェースは，従来のコマンドプロンプトや GUI システムと異なり，自然言語の解釈が必要であることを述べた．これらより，本論文では音声インターフェースの入力テキストを，要件を満足できるように自然言語処理するための方法を検討する．

図 1.6 に示すようにして，要件から研究課題を明らかにする．

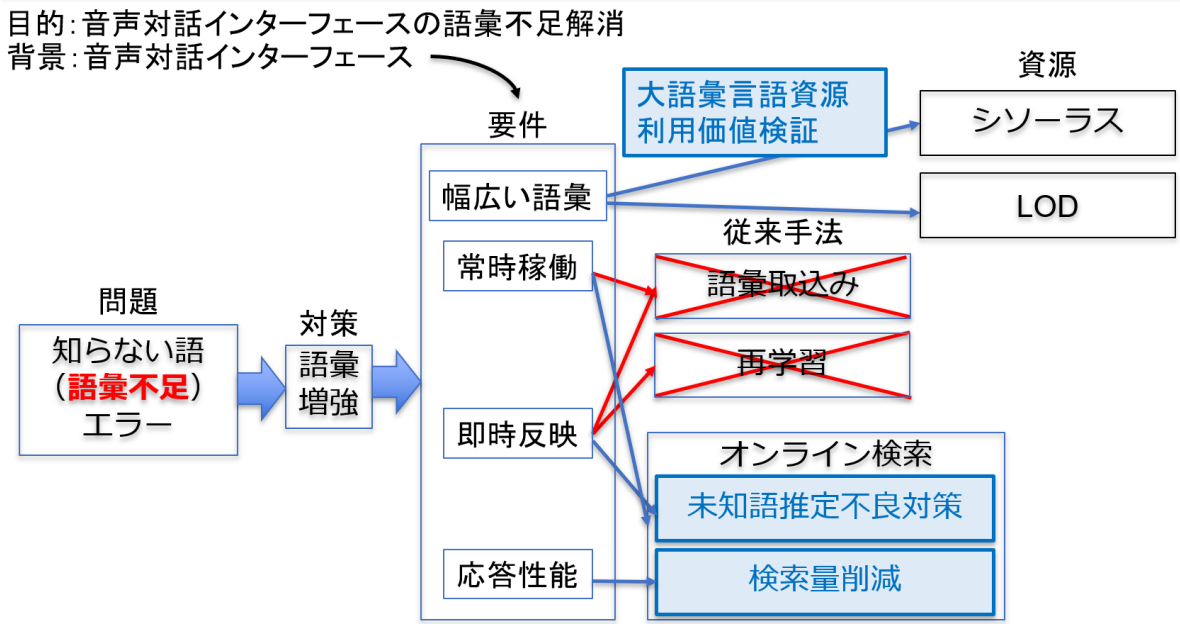

図 1.6　要件から課題の導出

本論文では，音声対話インターフェースの語彙不足を軽減することを目的とする．システムには「知らない語」があるため，語彙を獲得する方法を検討する．前節で述べたように，音声対話インターフェースには「幅広い語彙の獲得」「常時継続稼働」「新語を即時反映」「応答性能」といった要件がある．常時継続稼働，新語を即時反映の要件から，語彙取り込み，再学習といった従来手法だけでは不足であると考えられる．

幅広い語彙の獲得のため，シソーラスや LOD などの大規模言語資源を活用することを考え，その活用の価値を検証することを課題とする．

常時継続稼働，新語の即時反映のためには外部のデータベースや LOD エンドポイントなどをオンライン検索可能としたい．形態素解析や固有表現抽出では，辞書増強や再学習なしに新語を即時反映させられず，語彙が不足している場合に問題が起こる．未知語推定という能力もあるが，語彙の不足による不具合もある．この未知語推定の不具合に対して対策を検討することを課題とする．

オンライン統合する場合，語の候補を生成して検索することになるが，大量になると応答性能に影響が出る．そこで，検索量削減を課題とする．

まとめると，次の 3 つが研究課題である．

1. 大規模言語資源の利用価値を検証する
2. 未知語推定不良に対策する
3. 検索量を削減する

本論文の研究は，音声対話インターフェース特有の要件を満たしつつ語彙を獲得するために，外部にあるシソーラス，LOD，業務データベースなどをオンライン検索することを念頭においている点で独自性がある．

1.6 本研究の方針

前節で説明した課題に対する研究方針について本節で説明する．

1.6.1 複数大語彙資源利用の価値の検証

大規模言語資源の利用価値を検証する，という課題に対して，価値の検証を行う．

音声対話インターフェースにおいて語彙が原因となる処理エラーは，システム構築時に想定した語彙セットよりヒトが持ち話す語彙の方が広いことに起因する．その問題への対策として大規模データセットの語彙をシステムに取り入れることが考えられる．

本論文では日本語 WordNet および Wikidata の語彙をシステムに同時に取り入れて語彙獲得することの価値を検証する．この検証では二つのことを確認する．

- 大語彙資源には多くの複合語が見つかることを示し，複合語の発見に有用であること
- 日本語 WordNet と Wikidata は規模が違うものの両方を同時に取り入れることで，より広い語彙を利用できること

1.6.2 深層学習による固有表現抽出における長い未知語問題

固有表現抽出において，長い未知語の抽出では問題が発生する，という仮説を立てた．本研究では深層学習を用いて検証を行う．この検証では三つのことを確認する．

- 実際に問題が発生すること
- 深層学習の入力特徴量に語彙情報を注入すると効果があること
- 実社会において，この問題を検討する必要性があること

1.6.3　日本語形態素文字種境界法の提案

語彙をオンライン検索する上で，企業の構内システムや LOD エンドポイントなどネットワーク越しにアクセスする可能性のあるシステムを考慮すると，検索の負荷を減らす必要がある．テキストの中に，検索にヒットする箇所があるかどうか調べるには，単純に考えれば隣り合う文字をつなげた候補を作り，すべて検索してみることが考えられる．しかしそれはとても負荷が高いため，処理レスポンスに多大な影響がある．そこで，検索回数を削減する手法として「日本語形態素文字種境界（JMCTB）法」を提案し，その効果を検証する．

1.7　本論文の構成

第 2 章では，音声対話インターフェースの入力テキストにおける語の抽出に利用可能な関連技術として自然言語処理の手法を説明する．自然言語処理の分野では様々な形で語彙が利用されており，それぞれの技術において語彙はどのように管理されているかについても説明する．

第 3 章では，従来研究において自然言語処理に語彙増強がどのように取り組まれているかを説明する．従来の手法では，どのような点において音声対話他インターフェースの要件を満たさないかを説明する．また本研究で利用する大規模言語資源について説明する．

第 4 章では，複数大語彙統合の価値の検証について述べる．自然言語処理において語彙を獲得するには様々な方法がある．その中でも，一般語彙を獲得するには自然言語処理研究の中で作られてきた言語資源を利用することができる．シソーラスデータベースの日本語 WordNet や，Web 百科事典 WikiPedia のデータをベースにした DBPedia や Wikidata は，大語彙辞書データとして自然言語処理の中でも広く使われている．自然言語処理を活用した統計処理において形態素解析を使う統計がよく行われているが，大語彙辞書データを併用することの意義を検証した結果を示す．Wikidata のほうが非常に大規模なため，Wikidata のみを使う研究が多いが，双方を同時に使ったほうが語彙を増強できることを検証した結果を示す．

第 5 章では，深層学習 NER における長い未知語問題について述べる．テキストから特定の分類の言葉を発見する技術は，自然言語処理研究分野において固有表現抽出（Named Entity Recognition: NER）と呼ばれる．音声対話インターフェースを含むテキストインターフェースの自然言語処理においても NER 技術は役立つ．NER の最新手法は，時系列データに強い深層学習を使うものである．本論文では，深層学習 NER の学習データに

含まれていた語より長い未学習の語の抽出に問題があることを指摘する．その現象が実際に発生する様子と語彙情報を入力特徴量に追加することで回復することを検証した結果を示す．検証には日本語および英語の著作物題名を利用し，日本語のマンガとアニメーションは英語のものより長いこと，年々長くなっていることを検証した結果を示す．これらの検証によって，深層学習を利用する NER システムの入力に，外部から語彙を注入することの必要性と効果があることを示す．

第 6 章では，日本語形態素文字種境界法の提案について述べる．大規模語彙をシステムに統合する際，外部にあるデータベースを即時検索する要求がある．データベース上に見つかる語をテキスト中に見つけられるか検査する処理では，検索回数が膨大になるという問題がある．テキストインタフェースでは応答時間に厳しい制限があるため，処理時間が大きくなることを回避したい．日本語では語の最低単位となる形態素と，含まれる文字種の変化する箇所を参考に，名称などの切れ目を推測できる．検索候補を絞ることで検索回数削減を実現する，日本語形態素文字種境界（JMCTB）法を提案する．話し言葉コーパスと Wikidata を利用して実験を行い，効果を測定した結果を示す．

第 7 章では，本論文で説明している 3 つの研究成果を総合的に使うということはどういうことか，総合的に使うことでどのような効果が得られるかを説明する．複数の語彙資源を直接的に利用し，深層学習 NER の入力に特徴量として語彙情報を追加し，その語彙情報の検索を JMCTB によって検索削減する，というアーキテクチャーを示す．このアーキテクチャーにより，深層学習の欠点の一つである再学習や追加学習を回避でき，新語が即時反映され，応答時間への制約を守ることができることを説明する．

第 8 章で結論を述べる．検証結果を用いてテキストインターフェースに大規模語彙を統合することの価値と効果をまとめる．実際にテキストインターフェースに大規模語彙を統合して運用するには，さらなる課題が存在し，それらを今後の展望として示す．

関連技術

第 2 章

本章では，語の抽出に利用できる本論文に関連する技術を説明する．

2.1　音声対話における入力テキストの解釈

本論文において基礎となる技術は，テキスト中に語を発見し，その語の位置づけを正しく推定することである．本節では，従来研究においてどのような手法があるかを説明する．

2.1.1　テキスト処理・言語処理

音声認識が終わると「入力テキスト」が手に入る．本論文では，以下の二つの視点で入力テキストの解釈処理を行うこととする．

1. 語の候補の抽出
2. 周辺語を利用した意図の推定

語の抽出と，周辺語を利用した意図の推定は，順番に処理しても同時に処理してもよい．抽出された語の分類を「語の属性」や「語義」と呼ぶ．入力テキストの語義と周辺語を組み合わせることで，入力テキストの意図が推定される．他の箇所と同様，意図の推定に関しては本論文のテーマ外であるため，説明を省略する．

2.1.2 ラベル付け

本論文の研究の中心にあるのが「語の抽出」処理である．テキストから語を抽出するためによく使われている手法は，テキスト中の「どの範囲」に「なに」を発見したかのデータを生成する，というものである．前出の例文「ヒラメはありますか」の中に，見つかった「ヒラメ」に対して生成されるデータは，以下のようなものである．

- 開始：0
- 終了：3
- 表記：ヒラメ
- 分類：魚

このようなデータを「タグ（tag)」「ラベル（label)」「アノテーション（annotation)」などという．それらを生成する処理を「タグする，タグ付けする（tag, tagging)」「ラベル付けする（label, labeling)」「アノテーションする（annotate, annotation)」などという．生成する処理プログラムを「タガー（tagger)」「ラベラー（lableler)」「アノテーター（annotator)」などという．本論文ではラベル，ラベル付けに統一して使う．

2.1.3 スロットフィリング

文章から語を抽出し，意図を推定することは音声対話インターフェースの手法の一つである．アプリケーション機能はプログラムメソッド（関数）と考えると，入力テキストから引数にあたる部分を取り出す必要がある．入力文に引数が埋め込まれているはず，と考え，その箇所を「スロット」と呼ぶ．スロットが一つないし複数あり，穴埋め問題のように考えるので「スロットフィリング」という．スロットフィリングができ，周辺語を活用することで話しかけられた内容の意図が推定できれば，アプリケーションプログラムが呼び出せる．**表 2.1** に例を示す．

表 2.1　スロットフィリングの例

	スロット		スロット	
期待	<service>	で	<meal>	を食べた
入力	銀のさら	で	寿司	を食べた

この例では，システムは service と meal という二つのスロットを埋めたい．スロットを埋めるために，ヒトが話しかけてくる文として「<service> で<meal> を食べた」という

文型をシステムは期待している．そこに「銀のさらで寿司を食べた」と話しかけたときには，service=銀のさら，meal=寿司，が取り出せれば処理が成功したと言える．

2.1.4　語彙

語彙とは，表記と意味または位置づけがセットになったものである．ここで，表記とは言葉を文字列で表したもので，IT システムではバイト列である．「言葉の意味」を，文章の中で「どれを意味するか」という意味合いで「語義」と呼ぶ．**図 2.1** は，語義を抽象化構造で管理する例を示している．

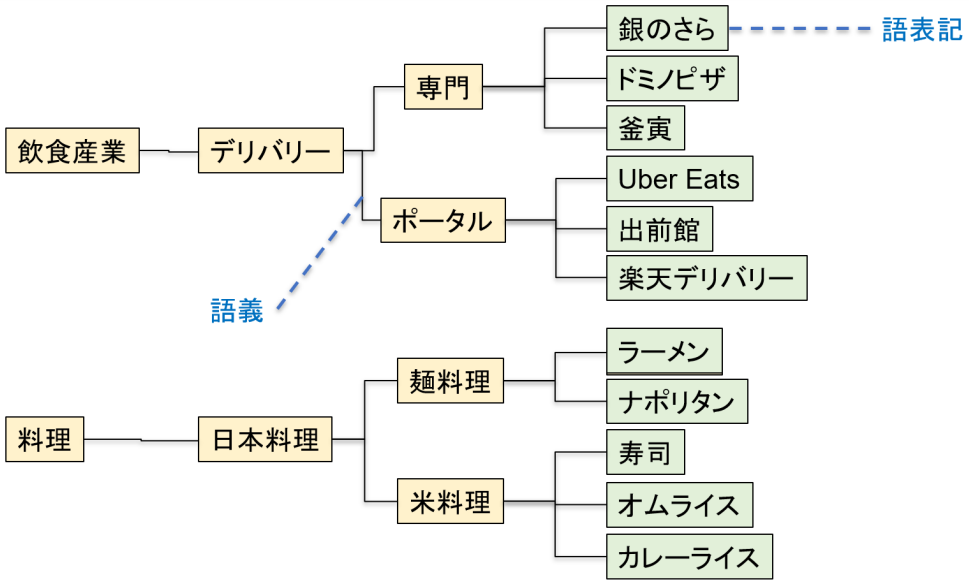

図 2.1　語彙の管理

上位概念を「概念化」「抽象化」などと言い，下位概念を「具体化」「具象化」などと言う．「銀のさら」は語表記と呼び，上位概念は飲食産業のデリバリーの専門業である．

一つの表記に対して複数の語義があり，ひとつの語義に対して複数の語表現があることが処理を困難かつ複雑にしている．逆に，一つの語義に対して複数の語表記もある．

2.1.5　複数の語義をもつ語表記

語義が一つの場合，ラベル付けをするのが楽である．文字列を比較して同一であれば，そこにラベル付けされるべきである可能性が非常に高い．

しかし現実には，多くの語は複数の語義を持っており，ラベル付けの困難さの原因となっている．例えば，図中の「ラーメン」に着目する．ここでは「麺料理」の一種であると管理されている．しかし，同じ列に「壁式」「ラーメン」「ブレース」「筋交い」といった言葉が並ぶと，これらは「建物構造」が上位概念となる．これは異義語の例である．「ラーメンを食べる」であれば，ラーメンは麺料理であるし，「ラーメンを採用したビル」であれば，ラーメンは建物構造である．このように「ラーメン」という文字列がテキスト中に存在することを確認しただけでは，正しい語義を選択できない．「ラーメンを食べる」「ラーメンを採用したビル」の二つの文において，正しい語義を選択するための重要な情報は周辺語にあることが多い．

2.1.6 言い回しに使われる可能性のある名称

語表記の中には，一般名詞や接続詞などを組み合わせたものもあり，場合によってはその対象物でない箇所に出現する可能性のあるものもある．**表 2.2** に，映画題名または Blu-ray 題名の「君の名は」を例に，文章の一部が間違ってラベル付けされる例を示す．

表 2.2 語の抽出の例

入力文	ラベル付けされるべき
君の名はを観に行った	映画題名
君の名は持ってる	Blu-ray 題名
山田**君の名は**なんですか？	されるべきでない

太字は語表記として合致する箇所，下線部はラベル付けされるべき箇所である．入力文「**君の名は**を観に行った」に対して「君の名は」が辞書に見つかる．この場合はタグ付けされるべきであり，二つの候補の中から選択しなければならない．どちらでも適切ではあるが，劇場映画のほうが少し確率が高い．

入力文「**君の名は**持ってる」だとすると少し事情が変化する．この場合は，Blu-ray 商品のほうが確率が高い．

入力文「山田**君の名は**なんですか？」に対して，「君の名は」が辞書に見つかるが，これは名前を訪ねている文であり，タグ付けされるべきでない．この入力に回答する場合「太郎です」というような回答がふさわしい．このように，辞書に見つかる文字列でもラベル付けされるべきでない場合があることに注意されたい．

2.2 自然言語処理技術の適用

自然言語とは，自然発生した言語のことをいい，より分かりやすく言えば普段私たちヒトが話す言葉のことである．英語，スペイン語，ドイツ語，フランス語，日本語，韓国語，中国語といったものである．それに対して，定義言語やプログラミング言語のような人工的に作られたものを人工言語と呼ぶ．XML，JSON，C 言語，Java，JavaScript，Python といったものである．

自然言語処理:Natural Language Processing(NLP) は，コンピューターすなわち計算機を使って自然言語の処理を行うことをいう．一方，コンピューターとは関係なく自然言語の研究を行う分野もあり，自然言語学，あるいは単に言語学という．言語を論理的にとらえる論理言語学や，それをさらに論理式で理解しようという計算言語学という分野もある．しかし，これら言語そのものの研究は NLP とは違うアプローチのものである．ここでは，コンピューター技術を利用した NLP について主に説明する．NLP には非常に多くの研究があるが，ここでは本論文の展開に必要なものを選択して説明する．

NLP 研究の多くが，ある程度大きなボリュームの自然言語で書かれたテキスト文書をさまざまな手法で整理することを題材としてきた．音声対話インターフェースにおける NLP の役割は，入力されたテキスト一文の即時処理である点が特徴的である．

2.2.1 文字列比較

語を抽出するもっとも簡単な方法は文字列比較で，該当の語表記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，単純一致などいくつかの方法がある．文字列比較は，入力文の全部または一部があらかじめ準備されている文字列と同じかどうか比較する方法である．プログラミング言語には，文字列比較の関数があらかじめ準備されていることが多いため，特殊なライブラリなどが必要ない．自然言語的な手法ではないため，単純なバイト列比較方法といえる．比較的単純なプログラムで実装できるため，初歩的な方法といえる．たとえば，Java 言語であれば，String クラス [42] の equals，indexOf，contains，startsWith，endsWith，などの関数が使える．

次のプログラムコードは，Java において，入力文 input 中に「りんご」が含まれているか調べる例であり，結果の boolean 引数 result には true が返る．

```
String input = ” りんごのパイが食べたいな”;
boolean result = input.contains(” りんご”); // true
```

この例では「りんご」が語彙である．

2.2.2 正規表現によるパターンマッチング

正規表現 [50] は柔軟な表現で文字列比較をおこなうパターンマッチングの技術である．たとえば，Java であれば Pattern/Matcher クラス [41] を使う．次のプログラムコードは，Java において，入力文が「〜が食べたい」のような表現が含まれているか調べる例である．

```
Pattern pattern = Pattern.compile(”ˆ.+ が食べたい (よ | な)?$”); // パターン
System.out.println(pattern.matcher(” りんごが食べたい”).find()); // true
System.out.println(pattern.matcher(” りんごが食べたいよ”).find()); // true
System.out.println(pattern.matcher(” りんごが食べたいな”).find()); // ture
System.out.println(pattern.matcher(” りんごが食べたいぜ”).find()); // false
```

最初の行で渡している文字列は正規表現パターンと呼ばれ，これによって柔軟な文字列検査ができる．先頭のキャレットは「検査対象文字列の先頭である」，ピリオドはなんらかの文字列，プラス文字は「一つ以上含む」，カッコ内の縦棒で仕切られたものは「このうちどれか」，ドルマークは「検査対象文字列の最後尾」，などと言った意味である．このプログラムは，結果として，true，true，true，false を出力する．最後の文字列は「ぜ」がパターンに合致しないからである．

2.2.3 N gram 文字列検索

語の抽出を行うにあたり，前出の単純比較のパターンパッチングや正規表現では，調べたい語の回数だけ検査を行う必要があり問題となる．もし，語彙が 40 万語ある場合，入力テキストを 40 万回検査しなくてはならず，現実的でない．

語彙を使ってインデックスを作り，入力テキストのフラグメント，すなわち文字列の断片を作って，語彙インデックスに存在するか検査する方法が考えられる．N gram は，入

力テキスト上で開始文字をずらしながら N 文字の文字列断片を生成する手法で，その語が語彙に存在するかを確認することができ，NLP ではポピュラーな方法である [56]．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．例えば，Java であればハッシュインデックスを作る HashSet や HashMap クラスが利用できる．

表 2.3 に，果物の名前を検査するプログラムコードと，出力結果の一部を示す．

表 2.3　N gram 検査の例

プログラムコード	出力

HashSet <String> set = new HashSet<String>();
set.add(”りんご”);
set.add(”みかん”);
set.add(”ぶどう”);
String input = ”りんごが食べたい”;
for(int start=0; start<input.length()-1; start++) {
　for(int end=input.length(); end>start; end–) {
　　String fragment = input.substring(start,end);
　　boolean found=set.contains(fragment);
　　System.out.println(
　　　fragment + (found ? ” found” : ””));
　}
}
	りんごが食べたい
	りんごが食べた
	りんごが食べ
	りんごが食
	りんごが
	りんご found
	りん
	り
	んごが食べたい
	んごが食べた
	んごが食べ
	んごが食
	んごが
	:

HashSet クラスは，Java で簡単にハッシュインデックスを作ることができるクラスで，contains メソッドで対象の文字列がインデックス上にあるかを true/false で返す．例では，「リンゴ」「みかん」「ぶどう」が語彙として候補にあり，N gram で断片を作ると「りんご」が合致して見つかることがわかる．

出力を見ると表示されている範囲の中だけで見ても 1 行だけが該当し，それ以外のすべてが比較対象として無駄な検索をしていることがわかる．第 5 章で提案する日本語形態素

文字種境界（JMCTB）法提案のきっかけになる事象である．

2.2.4 大量文書から語の候補の抽出

語の抽出を行うタスクとして，大量文書から N gram インデックスを集めて集計する手法がある．語として認定できそうな連続した文字列は，N gram インデックスを作ったとき複数回出現する．とくに回数が多いものは，ひとまとまりの言葉の可能性が高い．

大量の文書があったとき，全体の N gram インデックスを作り，文書ごとの出現回数を集計する．特定の文書を観察し, 他の文書よりも特定のインデックスの出現量が多いとき, その語は「特徴的に出現している」という．インデックスの出現の特徴ごとに分類することで, 文書をグループ化することができる．この手法を TF-IDF (term frequency–inverse document frequency) [43] といい，そのような処理を文書分類という．

大量の文書の N gram インデックスでは，短い断片とそれを含む長い断片のインデックスも作られることがある．断片の切れ目となる箇所の出現回数を集計することで，その箇所が切れ目として有効かどうかを評価することができる．この手法を最大エントロピー法 [2] という．

TF-IDF や最大エントロピー法は，あくまでも大量文書から語の候補を導き出す手法である．音声対話インターフェースでは入力テキスト 1 文を即時処理する必要があり，目的が異なり，制約も違う．

2.2.5 日本語形態素解析

日本語形態素解析 [73] は日本語のテキストバイト列を論理的な最小単位に分解（分かち書き）し，品詞推定する処理を併せた処理である．英語などのラテン語系言語では単語がスペースで区切られているが，日本語にはない．そのため，日本語特有の特殊な処理によって区切る処理を行う必要がある．

日本語においてもっとも短い語は，1 文字である．たとえば，海，空などである．文字単体だけでなく，複数の文字や，おくり仮名などを結合して論理的な単位となる場合のほうが多い．形容詞，副詞，動詞などの場合，送り仮名などがつく場合があり，それらは数文字で構成され,「軽い」「早く」「作る」などがある．一文字の語や，送り仮名などを加えた語は，日本語を構成する最小構成とされており「形態素（morpheme）」と呼ばれる．日本語で書かれたテキストを形態素に分解し，個々の品詞を推定する処理を「形態素解析」という．形態素解析の歴史は長く，以下に著名な形態素解析器（プログラム）の名称を列

挙する．

- 茶筌 [36]
- MeCab[31]
- JUMAN++[51]
- GoSen[4]
- Apache lucene-gosen[4]
- Kuromoji[7]
- Juman[51]
- Janome[52]
- Sudachi[49]

図 2.2 に，「銀のさらで寿司を食べた」というテキストを日本語形態素解析で形態素解析した例を示す．

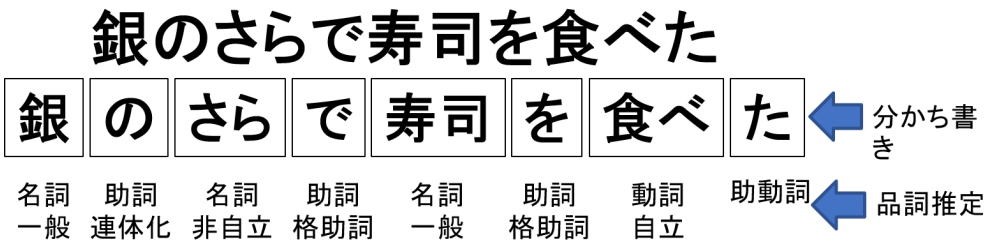

図 2.2　形態素解析の例

このテキストは「銀, の, さら, で, 寿司, を, 食べ, た」のように分割される．このように分割することを「分かち書き」という．形態素を英語で Morpheme あるいは Token，分かち書きを英語で Tokenize というため，日本語形態素解析器は英語で Japanese Tokenizer あるいは Morphological Analyzer と呼ばれている．先に挙げた形態素解析器では，品詞辞書を使い解析を行うため，分かち書きと同時に品詞がわかる．名詞・一般，助詞・連帯化，名詞・非自立，といったものが品詞である．各形態素に品詞を紐づける処理を「品詞推定」という．

文書に含まれる語を抽出して統計するタスクにおいて，形態素解析は入門的によく使われる．しかし，形態素だけを集計すると，この例のように「銀のさら」が解体されてしまい，処理が困難になることがわかる．

英語やドイツ語などのラテン語ではテキストの単語がスペースで区切られているた

め，分かち書きの必要がない．ただし，品詞推定は同様に行うことができ，必要であればApache OpenNLP[3] などを利用することができる．品詞を英語で Part Of Speech(POS) といい，タグ付けをするという意味で OpenNLP のようなプログラムを POS Tgager と呼ぶ．

2.2.6 形態素解析辞書

形態素解析や POS Tagger の処理の仕組みは，文字列比較と並び順の統計確率による最適解の選択である [6]．形態素になる可能性のある表記の辞書を用意しておき，N gram と辞書上の語の表記との文字列比較によりテキスト上の文字列と辞書エントリーを紐づける．語の切れ目と選択される品詞に複数の可能性が考えられるが，有向グラフ等の考え方で確率計算することで正しいものを選択する．

複数の候補からの選択に，旧来は隠れマルコフモデル（HMM）[16]，条件付き確率場：Conditional Random Field（CRF）[57] を使って単語の並び順確率を推定しているものが多い [74]．たとえば，MeCab，Apache lucene-gosen，Sudachi などがそうである．今日では，RNN を使った形態素解析の研究もある [62]．

確率場を使う形態素解析では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．CRF などの入力データには単語表記と隣接する単語の品詞の可能性を示すパラメーターが必要であり，これを何らかの形で用意すればよい．これを「パラメーター推定辞書」と呼ぶ．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，機械学習の一種と言える．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，収録されている文章から単語並びの可能性が統計され，辞書データとして生成されてきた．古くから IPAdic[28]，NAIST-Jdic[38]，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[45] 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．

2.2.7 固有表現抽出

今日 NLP 界で盛んな研究に，固有表現抽出:Named Entity Recognition(NER) がある．テキスト中に出現する特定の分類に属するものを固有表現（Named Entity: NE）と言い，テキスト上に固有表現があることをラベル付けするタスクを NER と言う．NE

は MUC[9] で規定された分類で，組織名 (ORGANIZATION)，人名 (PERSON)，地名 (LOCATION)，日付表現 (DATE)，時間表現 (TIME)，金額表現 (MONEY)，割合表現 (PERCENT) が既定されている．IREX[47] でさらに「固有物名」が追加されており，これを固有名詞抽出手法と考えることができる．これらの既定は学術コンテストなどで性能評価をするめに共通化されているものであり，個々の用途に併せて分類を定義して使う．たとえば，医療テキストを対象とした NER[67] のように特定業務分野に特化した研究や，MultiWOZ[19] のような研究用データセットを使った NER 研究 [25] などがある．

固有表現抽出は，一つまたは複数の語の並びに対して，その用法としての意味を結び付けようとするので，日本語においては形態素解析の上位にある処理と考えることができる．たとえば，前出の「銀のさらで寿司を食べた」に対して「団体名（ORG）」や，「食品（FOOD）」のようなラベル付けを行うと**表 2.4** のようになる．

表 2.4　固有表現抽出の例

入力	銀のさら	で	寿司	を食べた
ラベル	ORG		FOOD	

この例は，**表 2.1** のスロットフィリングと同じ処理であることがわかる．すなわち，固有表現抽出は音声対話インターフェースにおけるスロットフィリング処理に適していると言える．[75]

NER には以下の三つの処理が必要である．

1. 範囲を特定する
2. 語義を選択する
3. ラベル付けするか否か決定する

正しい範囲に，正しい語義を選択し，正しい箇所にラベルをつけ，間違った箇所につけないことが求められる．処理に参考となる情報はテキストに含まれる文字列であり，語表記と一致する．周辺語を使って語義を選択したり，ラベル付けするかしないかを決定することになる．

ステップ 1 と 2 によって語の範囲が特定される．これをラベル生成と呼ぶ．ステップ 3 によって，あいまいさが解消（disambiguation）される．ラベルを推定するために最も重要なのは語表記であるが，それだけでは情報が不足する．たとえば「銀のさらに乗せて寿司を出した」という文であれば，「銀のさら」は組織名ではなく [色]-の-[一般名詞] という意味である．このように語義を推定するために，語表記に加えて周辺語がラベル付けに重

要な位置づけを持っている．

固有表現抽出のラベル付け処理は，あらかじめ定義されたラベルを選択して出力するため，分類タスクの一種である．

旧来からある手法として，語の上位概念辞書データを活用した軽量の属性抽出方法 [69] や，Hidden Malkov Model を利用した固有表現抽出 [58]，CRF を用いた研究 [32] などがある．

2.3　深層学習による固有表現抽出

近年の研究では，固有表現抽出に深層学習が広く使われるようになった．深層学習で NER を行う場合，学習データとして固有表現ラベルつきコーパスを使い「固有表現の切れ目と該当する分類」を学習させる．本節で具体的な処理方法について説明する．

2.3.1　BIO ラベル

表 2.5 は，「The Bridge on the River Kwai」という小説または映画の題名であり得る箇所を含むテキストの例である．

表 2.5　NER の入力と出力

index	0	1	2	3	4	5	6	7	8	9
input	**I**	**saw**	**The**	**Bridge**	**on**	**the**	**River**	**Kwai**	**yesterday**	**.**
Label as Movie	O	O	B-MOVIE	I-MOVIE	I-MOVIE	I-MOVIE	I-MOVIE	I-MOVIE	O	O

B-NOVEL と B-MOVIE は，それぞれ小説と映画の題名の先頭 (Begin) を示している．同様に，I-NOVEL と I-MOVIE は，題名の範囲 (Inside) を示している．さらに，O は範囲外 (Outside) を指す．これは BIO スタイルのラベルと呼ばれる，NER タスクでよく使用されているラベルの方式である．NER 処理では，まず正しい範囲を取得する必要があり，この例ではインデックス番号で 2〜7 の範囲を取得している．NER は次に小説と映画を選択する曖昧性の解消を行う．この例では，「B-MOVIE」と「I-MOVIE」が正しい選択である．

この文章は，一見すると「I saw the bridge（私は橋を見た）」と言っているように見えるが，実際には「I saw the movie（私は映画を観た）」と言っている．題名の知識があれば，ヒトは脳内でこの意味を修正することができる．ヒトは「saw」という動詞を使って，題名が映画であることを直感的に認識する．

2.3.2 深層学習の適用

NER システムでは，統計的手法を適用して範囲を認識し，正しいタイプを選択する．近年，いくつかの深層学習が高い NER スコアを獲得している．NLP-progress[44] では，いくつかのデータセットで 90% 以上の NER 精度が出ていることを報告している．

NLPprogress の CoNLL データセットに対する NER のランキングでは，CNN[33]，RNN[24]，CRF，LSTM[23]，Bi-LSTM[46]，BERT[13] が高得点を獲得している．これらは時系列データに適した深層学習である．近年，OpenAI[40] が発表した GPT-3[21] や EleutherAI[18] が発表した GPT-J[17] なども高い性能を発揮すると注目を集めている．これらも NER に適用できる [34][14]．どの手法でも NER において同じ特徴量を使用する．同時に，タグを示すラベルデータを用意し，トークンの並びを説明変数とし，ラベルの並びを目的変数とする．

図 2.3 は，深層学習 NER が入力テキストを処理する例を示している．

input	feature tensor						label	label tensor			
I	0	1	0	1	0	0	O	1	0	0	0
saw	0	0	1	1	1	0	O	1	0	0	0
the	1	0	1	0	1	0	B_MOVIE	0	1	0	0
bridge	0	1	1	1	0	0	I_MOVIE	0	0	1	0
on	1	1	1	0	1	0	I_MOVIE	0	0	1	0
the	0	1	0	1	1	1	I_MOVIE	0	0	1	0
river	1	1	1	1	0	1	I_MOVIE	0	0	1	0
kwai	0	1	0	1	1	0	I_MOVIE	0	0	1	0
yesterday	1	0	1	1	1	0	O	1	0	0	0
.	0	0	0	0	1	1	O	1	0	0	0

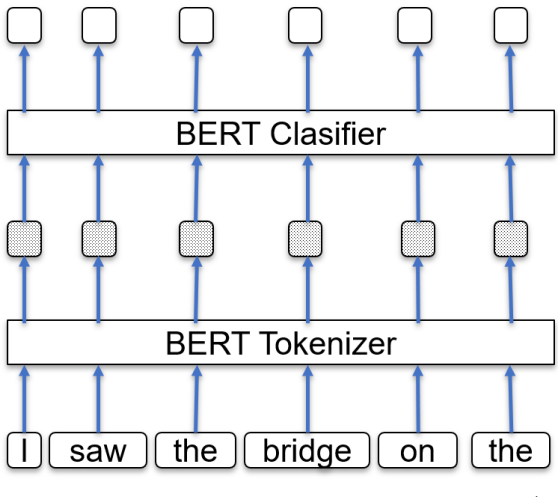

図 2.3　題名を深層学習で固有表現抽出する

深層学習 NER は，次のように処理される．

トークン化

入力テキストを単語や形態素の列（図中 input）としてトークン化する．ラテン語系の言語はスペース文字を使ってトークン化することができ，日本語は，日本語形態素解析器を使ってトークン化することができる．

コード化

テキストがトークン化されたあと，トークンの並びはテンソルと呼ばれるベクトルデータ（図中 feature tensor）に変換される．トークンの分散表現を入力特徴量として利用することで，精度の高いパフォーマンスを実現できる．ラベル（図中 label）も同様にテンソル（図中 label tensor）に変換されるが，ワン・ホット・ベクターなどよりシンプルな方法でよい．

学習

深層学習モデルは，形態素をトークンとしたテンソルを入力とし，ラベルテンソルを出力するように学習される．テンソル（ベクトル）に変換されてから用いられるので，もとの言語（日本語か英語か，など）に関わらず動作する．

この図では，入力テキストのトークンを分散表現テンソルに変換するために BERT Tokenizer を，ラベル推定に BERT Classifier を使うことを示している．

2.3.3 未知語推定

固有表現抽出では，事前に準備された訓練データに含まれる語彙が使われるが，その語彙に含まれないものもある程度抽出できることがある．例えば次のようなデータで学習したとしよう．

- 昨日**川崎市役所**に行った
- **東京都庁**に行くのは何日ぶりだろう
- **横浜市役所**の前にひとだかり

そこに次のようなデータが入力されたとする．

市電で行けば**函館市役所**はすぐだよ

この場合に「函館市役所」は学習データにないとしても，正しく抽出できることがある．これを「未知語抽出」または「未知語推定」という．ただし，必ず正しく処理できるわけではなく，うまくいかないケースも多く，一つの研究分野となっている．

関連研究

第 3 章

本章では，スロットフィリングを行うにあたり語彙増強に利用できる大語彙言語資源を説明し，語彙の増強のために行われてきた従来研究の手法を説明する．

3.1 大語彙言語資源

「うしのした」のように標準でない一般用語や「君の名は」のような作品題名において，その語の知識を音声対話インターフェースに取り入れるために語彙資源を利用することが考えられる．自然言語処理研究において，大量の語をまとめあげたデータを大規模言語資源という．本節では，本論文で検証などに利用している大規模言語資源である日本語 WordNet と Wikidata について説明する．

3.1.1 日本語 WordNet

WordNet は，世界的に有名なシソーラスデータベースとその派生である．シソーラスとは類語辞典，対語辞典などと呼ばれるもので，元来は紙で，人が読むものである．WordNet はシソーラスデータベースであり，1995 年にプリンストン大学の心理学者ジョージ・ミラーらの手によって最初の Princeton WordNet[3] が作られた．その後，2009 年から 2010 年にかけて情報通信研究機構（NICT）によって WordNet に日本語を追加する形で派生である日本語版が作られた．現存する最新の日本語 WordNet はバージョン 1.1 で，ベースになっている Princeton WordNet のバージョンは 3.0 である．本研究では最新の日本語 WordNet 1.1[29] を使う．

表 3.1 に登録語彙数を示す．

表 3.1　日本語 WordNet 1.1 の語数等

種類	個数
概念 (synset 数)	57,238
表記（word の数）	93,834
語義 (synset と単語のペア)	158,058
定義文	135,692
例文	48,276

3.1.2　Wikidata

Wikidata[20] は，ボランティアが作る Web 百科事典プロジェクト WikiPedia[60] のコンテンツをもとに作られたナリッジデータベースである．セマンティックウェブ [55] やオントロジーなどで使われるトリプル構造となっている．トリプル構造とは，あるアイテムから，別のアイテムに述語でつながっていることを示す．Wikidata は，W3C で規定されている RDF[53] フォーマットとして，同じく W3C で規定されている検索言語 SPARQL[54] で検索できるエンドポイントとして公開されている．WikiPedia に掲載されているすべてが載っているわけではないが，非常に膨大である．データは日々追加更新されているが，2021 年 4 月 8 日時点でのダンプデータから数えた日本語ラベルの個数は 2,525,494 個である．Wikidata はインターネットに公開されオンラインでアクセスするため，固定のバージョンの概念はないため，実験等実施時のものを使う．

以前 DBPedia という類似した SPARQL エンドポイントが存在したが，現在は運用が終わっている．第 4 章の検証で DBPedia を使っているが，Wikidata と同様に考えてよい．

3.1.3　共通のデータ構造

日本語 WordNet および Wikidata のデータ構造を模擬的な図に表すと **図 3.1** のようになる．

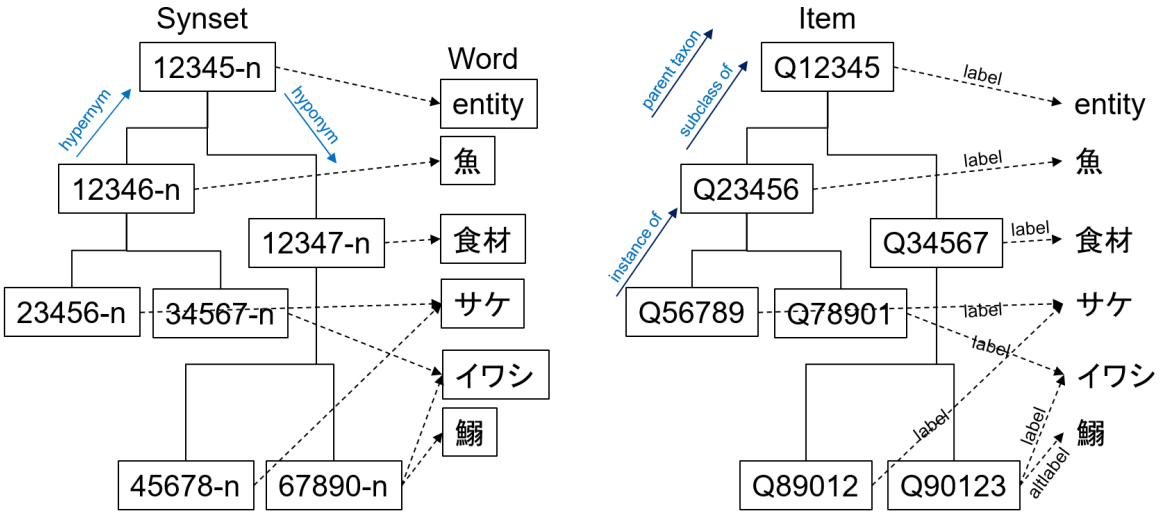

図 3.1　日本語 WordNet と Wikidata のデータ構造

左が日本語 WordNet，右が Wikidata である．

日本語 WordNet では概念を Synset というデータで表し，そこには Synset-id という識別子がある．Hypernym が上位概念，Hyponym が下位概念というリンクで概念の上下関係を示す．それぞれの概念 Synset には Word という語表記がゼロ，または多対多で紐づけられている．

一方，Wikidata は，Item という単位で概念を表し，Qnnnn（n は数字）形式の識別子がある．一つの Label とゼロまたは複数の AltLabel という属性によって表記が結び付けられている．概念の上下関係は強く意識されていないが，ペアレント・タクソン，サブクラスオブ，インスタンスオブの属性を追跡すると，日本語 WordNet と同様の構造が追跡できる．

第 2.1.4 項で説明したような語彙システムの概念と酷似しており，このデータを語彙データとして活用できることがわかる．

3.2　語彙増強アプローチ

事前学習された形態素解析と深層学習 NER を組み合わせた場合，形態素解析器のシステム辞書作成時や深層学習 NER の事前学習モデル作成時に使われたラベル付きコーパスに含まれていない語は未知語となる．そこで，新語を追加する方法を検討する．本節では現存する形態素解析器や深層学習 NER に語彙を追加する標準的な方法と，音声対話イン

ターフェースの要件における問題点を説明する．

3.2.1 システム語彙に追加

形態素解析器（プログラム）は一般名詞のラベル付けが基本の処理だが，固有名詞にラベル付けすることもできる．固有名詞にラベル付けするには辞書データを修正する必要がある．システムに統合される言語処理プログラムがもっとも性能よく働く方法は，言語処理プログラムが持つ処理モデルの語彙を完全に作り直すことである．

単独の辞書を活用して語彙を増強した質問応答システム [65]，WordNet と日本語語彙体系を取り入れた対話システム研究 [78]，Wikdiata の語彙を取り入れようという取り組み [10][48]，などがある．

どれも「語彙を複製してシステム語彙とする」という研究であり，「新語の即時反映」というニーズに応えられない．形態素解析のシステム辞書や，深層学習 NER の事前学習済み認識モデルなどは，学習に使われたラベル付きコーパスが完全な形で公開されているわけではないうえ，学習にかかる計算コストがとてつもなく高い．日々の語彙追加に選択できる方法とはいえない．

3.2.2 ユーザー辞書に追加

いくつかの日本語形態素解析器でユーザー辞書が利用できる [72]．ユーザー辞書とは，システム辞書とは別に，形態素解析器を使うユーザー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，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．大語彙はシステム辞書に存在しているので，大語彙をシステム上に残したまま新語を追加できる．

形態素解析のユーザー辞書は大規模コーパスなどからパラメーター推定するわけではなく，前後品詞や確率を手作業設定するため，適切なパラメーターにならないこともある．品詞，という考え方から抜け出せないため，第 2.1.4 項で説明したような語彙システムに結合することが難しい．ユーザー辞書ファイルはコンパイル処理によって作成し，初期化時にメモリーに読み込まれるため「新語の即時反映」というニーズに応えられない．

3.2.3 再学習・追加学習

第 2.3 節で説明したように，深層学習 NER は形態素並びに対してラベルを生成する．このため，形態素解析辞書に複合語を追加しても深層学習 NER がその追加した語に正し

いラベルを生成することはない．なぜなら，形態素として出力された新語に対してテンソルを生成する際「未知語」となり，その未知語に対してラベル生成するかどうかの処理となるため成功率が低い．すなわち深層学習において語彙を追加するには学習を実行しなおす必要がある．本研究において対象となるのは大きな語彙を対象とする場合のため，学習に長時間を要することが想像できる．たとえば，100,000 語の BERT の pre-training に，NVIDIA V100x32 枚で 9 日もかかる [39]，というデータもある．常時継続稼働，新語の即時反映という要件を満たせない上，日常の運用において負荷が大きすぎて実運用に耐えない．

BERT などの一部の深層学習 NER では追加学習ができる．形態素解析でいうユーザー辞書生成のようなものと考えてよい．これも学習処理，識別器の初期化が必要であるため「新語の即時反映」というニーズに応えられない．

3.2.4 大語彙言語資源の利用

IT システムが広範囲な語彙を獲得する言語資源としてシソーラスや Linked Open Data(LOD) の活用が考えられる．たとえば，固有表現抽出にシソーラス WordNet と Wikidata を統合する研究がある [2]．しかし，どのように補完関係にあるか示されていない．本研究では日本語 WordNet と Wikidata には登録語彙は補完関係にあると仮説する．補完関係に関する先行研究は調査した限りでは見当たらず，本研究で検証する．

3.2.5 大語彙資源の強化

英語版 WordNet に存在する単語に対し，特定分野の単語であることのタグ付けを付加する研究 [17]，同様に WordNet に存在する単語の選択的関連性情報をメタデータ的に付加する研究 [18]，WordNet と WikiPedia のデータを使って大きなオントロジーを構築する研究 [19] などがすでに行われている．これらの研究は，形態素解析辞書，シソーラス，オントロジーといったひとつの形態のデータの質を向上させるための活動である．本論文での研究は音声対話インターフェースの自然言語処理において語彙を統合して利用することが目的であり，目的が異なる．

3.3　各研究分野と本研究の位置づけ

第 2 章と本章で説明したように，音声対話インターフェースの入力テキストから語彙を活用して語を抽出するためには，多くの技術を組み合わせる必要がある．それらをとりまく従来研究との関係を説明する．

序論で触れたように，本研究は音声対話インターフェースの言語処理部における語彙に関する研究である．音声対話インターフェースの研究は「対話システム（Spoken Dialogue System）」分野の一分野として長年研究されている．

対話システムのアーキテクチャーには様々なものがあるが，本研究が対象とするものは音声認識結果のテキストを自然言語処理するものを対象としている．このため自然言語処理（NLP）研究の活用が前提となる．自然言語処理のうち，ラベル生成の技術が語の抽出に欠かせない．そこで形態素解析や固有表現抽出（NER）といった研究成果を活用する．

語彙に関する研究のため，語彙の資源が必要である．計算機と関係なく，自然言語そのものを研究する分野として自然言語研究がある．自然言語研究と自然言語処理研究は平行してお互いに影響しあいながら研究が進められてきた．その中で様々なデータが研究成果として編纂されており，言語資源と呼ばれている．とくに対話システムでは語義を抽象化して管理する辞書が利用しやすい．そのような言語資源の研究としてセマンティクスまたはセマンティック Web，あるいはオントロジーといった研究分野がある．本研究では，セマンティック Web やオントロジーで研究されているデータ日本語 WordNet や Wikidata といった研究成果を語彙獲得のための言語資源として利用する．

図 3.2 に，自然言語研究，自然言語処理研究，セマンティック研究，オントロジー研究など各研究分野との関連を示す．

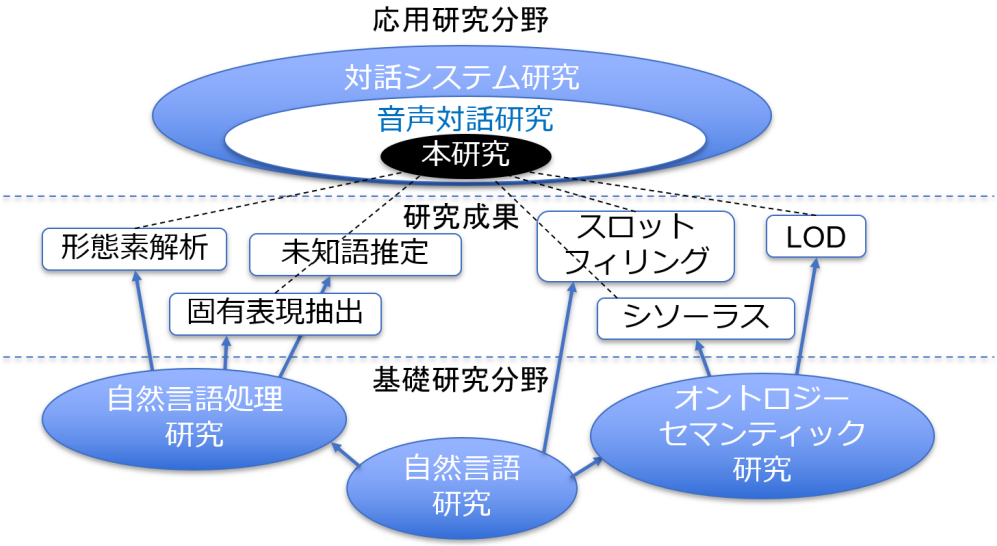

図 3.2　各研究分野と本研究の位置づけ

本論文の研究は対話システム研究の一分野である音声対話研究である．自然言語処理研究の形態素解析や固有表現抽出などの従来研究成果を使って語の抽出をする際，オントロジーやセマンティックといった研究によって作られた研究成果の言語資源を使って語彙獲得する手法の提案を行う．

複数大語彙資源利用価値の検証

第 4 章

本章では二つの視点において大語彙言語資源を語彙増強のために利用する価値について検証した結果を説明する．

4.1 はじめに

まず，大語彙言語資源における，複合語の掲載量の検証について述べる．日本語における NLP の一般的な処理として形態素解析の利用が広く紹介されている．2.2.6 節で説明したように形態素解析では形態素解析辞書を使う．本論文では形態素解析では固有名詞の抽出に不十分であることを指摘しているため，実情を確認するために形態素解析結果と大語彙に含まれる語との間にどの程度の差異があるかを調査する．

次に，大語彙言語資源を複数同時使用することの価値検証について説明する．第 3.1 節で説明したように，日本語 WordNet と Wikidata には，非常に多くの語が登録されておりスロットフィリングの語彙として有用である．本論文では，日本語 WordNet より Wikidata の語彙数が大きく上回っているとしても，同時に使うことで補完的に語彙を増強できると仮定し，相互補完性を確認する．

4.2 大語彙言語資源における複合語

本節では複合語が大語彙言語資源でどの程度発見できるか検証した結果を説明する．

4.2.1 大語彙言語資源における複合語

本研究において複合語とは，並び合う複数の形態素を意味する．スロットフィリングを行う際，語の範囲を推定することが処理の重要なステップである．スロットの値として検討する際，一つのトークン，すなわち形態素であれば，高い確率でラベル付けが成功する．次の例では，一つ目の例ではラベル付けされる対象は形態素ひとつで，二つ目の例では形態素が四つである．

> 昨日，**火花**，を，読ん，だ
> 昨日，**君，の，名，は**，を，観，た

一つ目の例において，「火花」はトークンが一つであり，周辺語との境界がはっきりしているので未知語推定によって「読み物（小説を含む）ではないのか」という推定が行われる可能性がある．二つ目の例は，未知語推定が失敗する典型的な例で，語彙知識が情報として付加されないと処理がうまくいかない．このような例から，複合語の知識を語彙資源から処理に注入する必要がある．文章からどの程度複合語が大語彙言語資源に発見できるかを検証する．

4.2.2 検証環境

本研究では，学術研究において利用しやすい点，著名である点，実験を実施した 2018 年時点で保守などが行き届いている点などを考慮して**表 4.1** に示すオープンソースソフトウェアおよびデータを利用した．ただし，DBPedia は現在運用が終わっている．

表 4.1　複合語の検証環境

形態素解析器	lucene-gosen，Sudachi
日本語シソーラス	日本語 WordNet
オントロジー	DBPedia

研究が進展するにあたり他の実装や他のデータを取り入れる可能があることは否定しない．とくに，商品化されてライセンス販売されているソフトウエアはより精度や緻密度が高く研究の価値が高い可能性がある．本研究では，上記のソフトウェアおよびデータを統合するにあたり, 主に Java を活用する．

4.2.3　対象とする入力データ

実験システムがどの程度語彙を網羅的に処理できるかを調べるために入力データが必要である．完全網羅性のあるデータは入手できないため，本研究では 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) 2011 年版 [19] を利用して，出現単語の率を比較する．後段（シソーラスやオントロジー ）にはあるのに，前段処理（形態素解析）で失敗するケースの比率も集計する．

4.2.4　調査方法

対象データの分布を調査するため，専用のプログラムを作成した．その実験システムのアーキテクチャーを**図 4.1** に示す．

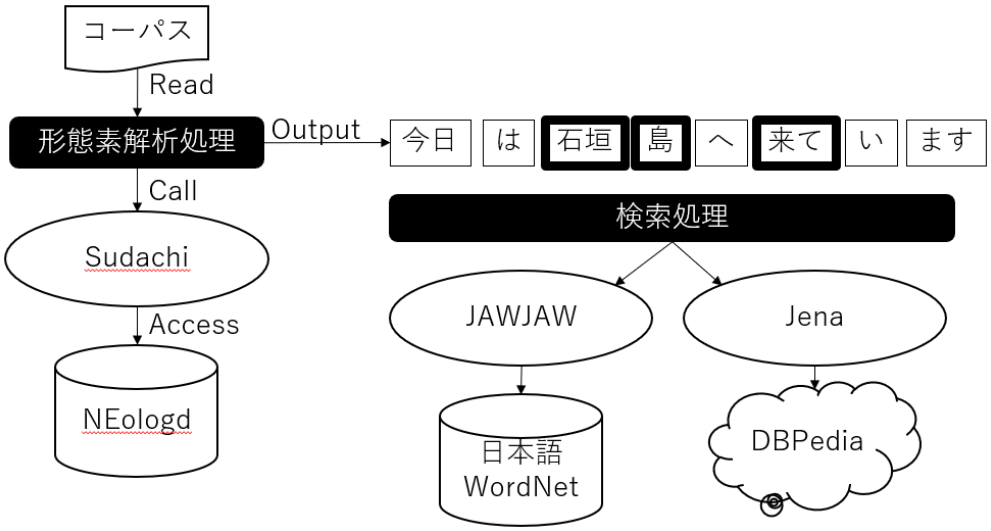

図 4.1　実験システムのアーキテクチャー

実験システムでは入力データとしてコーパスを利用し，形態素解析器として lucene-gosen と Sudachi，シソーラスデータとして日本語 WordNet，オントロジーデータとして DBPedia を使う．処理の流れは，コーパスから文を取り出し，形態素解析を行い，形態素解析出力のうち名詞と動詞をシソーラスとオントロジーで検索して集計する，というものである．実験システムでは，次のように処理が行われる．

1. コーパスデータを読み取る．Apache POI[20] を利用し，BTSJ データの EXCEL ファイルから談話部分の言葉を行データとして順次読み取る．

2. 行データを lucene-gosen と Sudachi へ送り，形態素解析を行う．
3. 形態素解析結果を順次検索し，名詞または動詞と解析されたものを検索処理へ送る．このとき，同一語，とくに動詞の場合は活用形ではなく正規化された語を使いインデックス化して整理して蓄積する．たとえば，[行き][ます] とか [行か][ない] の場合の動詞は [行く] を収集する．
4. 日本語 WordNet の Java API である JAWJAW[21] を通して日本語 WordNet を，Apache Jena を通して DBPedia を検索し，「存在するか」を検査する．
5. 検索対象になった語の数，日本語 WordNet に存在した語の数，DBPedia に存在した語の数をカウントする．

4.2.5 前処理

実験において，BTSJ コーパスから lucene-gosen と Sudachi を用いて名詞および動詞を抽出し，日本語 WordNet および DBPedia での登録率を集計するにあたり調整した点を説明する．

(1) **調整** 1: **ノイズの除去**

抽出された語には語彙となりえないものが含まれている．以下のものは除去して集計している．

- 非漢字の一文字：E，ん
- 数字のみ（数詞）：15，20
- 記号だけのもの：(^o^)

(2) **調整** 2: **除去フィルター**

同様に，日本語 WordNet や DBPedia 上に見つからないデータには，表記を見てみると検索価値のないものがある．これは，目検で確認して除去するフィルターを用いることとした．例としては「いっか」「ちょっかい」などである．

4.2.6 調査結果

日本語 WordNet では表記とともに POS.n，POS.v パラメーターで名詞・動詞の指定をしつつ，DBPedia では表記のみで検索を行い，該当文字列が見つかるかを検査した．個数と比率を**表 4.2** に示す．こちらのデータでは，WordNet および DBPedia で見つかる数が多いほど，性能に寄与する．

表 4.2　NLP 抽出された名詞と動詞の発見数

	品詞	All	日本語 WordNet	DBPedia
GoSen	noun	10,006	5,638(56%)	6,898(69%)
	verb	1,983	1,078(54%)	159(8%)
Sudachi	noun	10,475	5,714(54%)	7,308(70%)
	verb	1,461	1,108(75%)	110(8%)

本研究においてさらに主眼においていることは，日本語 WordNet や DBPedia には見つかるが形態素解析処理でうまく抽出できない語の問題を解決することである．形態素解析により名詞として抜き出された語が 2〜3 個並んでいるものを複合名詞である可能性があるものとして，同様に日本語 WordNet と DBPedia で探してみた結果を**表 4.3** に示す．

表 4.3　2〜3 個並んだ名詞の結合語の発見数

		All	日本語 WordNet	DBPedia
GoSen	triword	108,541	837	2,642
Sudachi	triword	11,764	269	1,323

こちらのデータでは，日本語 WordNet および DBPedia で見つかる数が少ないほど性能に寄与すると考えられる．NLP が正しく固有名詞を抽出できていれば 2〜3 個並べたものが固有名詞としてシソーラスやオントロジーから見つからないはずだからである．

4.2.7　語彙調査の考察

表 4.2 の調査結果において，lucene-gosen および Sudachi を使い BTSJ コーパスから名詞が 1 万語強抽出され，そのうち半数以上の 6,000 個弱が日本語 WordNet から，7,000 語強が DBPedia から見つかる．検索対象となっている語に，こういったシソーラスやオントロジー に登録され得ないと推測できる語も含まれている可能性はあるが，60%〜70% の名詞がこれらシソーラス，オントロジーから情報が得られない状況にある．名詞のみが連続で並ぶことが lucene-gosen で 10 万例以上，Sudachi で 1 万例強あり，そのうち日本語 WordNet から 200 以上，DBPedia においては 1000 以上が見つかる．日本語 WordNet 上で 0.8%，DBPedia で 2.6% が見つかることとなり，日本語 WordNet や DBPedia が有用であることがわかる．**表 4.2** の名詞数 1 万強に対すれば，10% 近くの語がより大きな複合語であり，文の意味を解釈する上で重要な情報が形態素解析時に抽出で

きていないことがわかる．

4.3 日本語 WordNet と Wikidata 統合の価値検証

本節ではシソーラス日本語 WordNet と Wikidata を同時に利用した場合の相互補完性を検証した結果を説明する．完全な言語資源は現存せず，一つの言語資源を取り込んだシステムでは語彙が不足していると考える．複数の言語資源を統合することが語彙増強に有用と仮説を立て，シソーラスの日本語 WordNet および Wikidata の語彙集合を比較することで補完関係にあることを検証する．

4.3.1 検証手順

日本語 WordNet 1.1 は sqlite のデータベースファイルとして配布されており，いくつかの API ライブラリが存在する．今回は JAWJAW[7] を使い，Java によってアクセスする．Wikidata は SPARQL エンドポイントのため HTTP で Web 経由でアクセスできる．Apache Jena[8] を使い Java から呼び出すことで利用する．

本研究では分野を絞って比較することとする．全レコードを比較対象とすることもできるが，数量が膨大なうえ，一つの語表現が他分野にわたって異義語として登録されているものを処理することは処理が複雑になりすぎることがその理由である．対象は「料理」「食材」「魚介類」の名詞とする．魚介類は生物分類から追跡するが，ひとつの概念から追跡することが困難なため，いくつかのグループにわける．

語の検索には上位・下位概念関係を使う．まず対象となる分類の語の共通概念を探す．探索にはツリー構造に表示できる可視化ツールを使う．**図 4.2** は，甲殻類の上位概念を探索する例である．

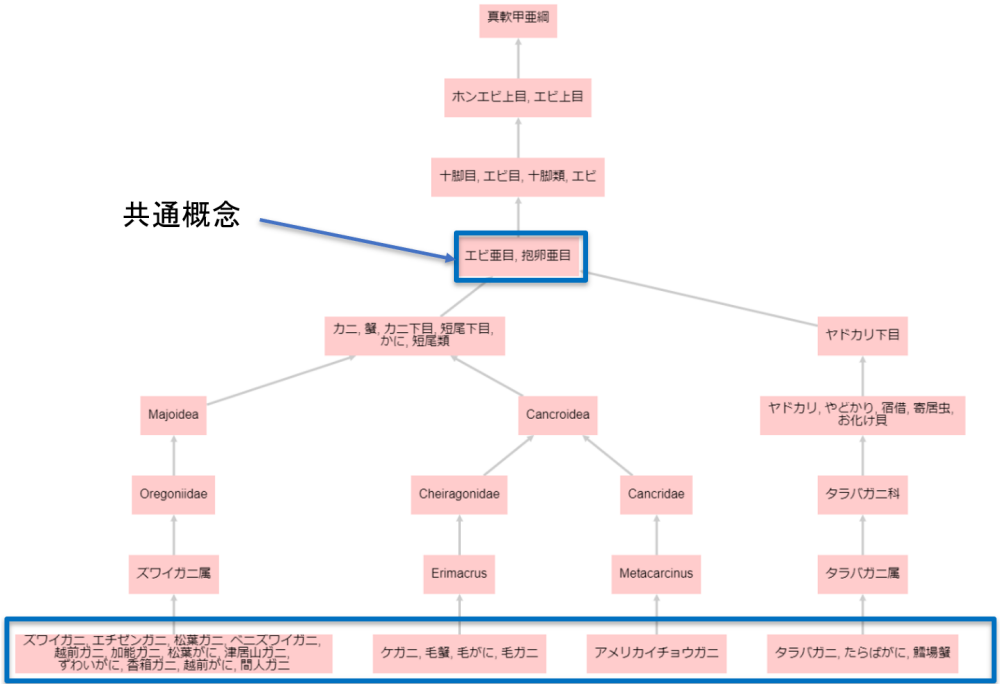

図 4.2　甲殻類の上位概念の探索

この例ではズワイガニ，毛ガニ，アメリカイチョウガニ，タラバガニなどを使い，上位概念「エビ亜目」が見つかる．その他エビの種類などを考慮すると，さらに上位の概念を選択することとなる．同様の手順で，魚類，頭足類（イカ，タコ等），二枚貝（ホタテ等），腹足綱（サザエ, タニシ等），棘皮動物（ナマコ等），甲殻類（エビ, カニ等），料理（カレーライス等），食べ物，食材の共通上位概念を探索する．**表 4.4** に日本語 WordNet, Wikidata それぞれで利用する Synset および Item の識別子を示す．

表 4.4　語の検索に利用する識別子

	日本語 WordNet	Wikidata
魚類	01473806-n	Q127282
頭足類（イカ, タコ等）	01971094-n	Q128257
二枚貝（ホタテ等）	01955933-n	Q25368
腹足綱（サザエ, タニシ等）	01942177-n	Q4867740
棘皮動物（ナマコ等）	02316707-n	Q44631
甲殻類（エビ, カニ等）	01974773-n	Q25364
料理（カレーライス等）	07557434-n	Q746549
食べ物，食材	07555863-n	Q25403900

これら日本語 WordNet の Synset-id および Wikidata の Item-id を開始として下位概念のラベルを収集し，付き合わせをして補完関係を調査する．日本語 WordNet および Wikidata では英語のラベルも入手できるが，今回は日本語ラベルのみを対象として集計する．

4.3.2　結果と考察

集計結果を**表 4.5** に示す．

表 4.5　日本語 WordNet と Wikidata の語彙数

	日本語 WordNet				Wikidata		
	Synset	N	oN	C	Item	D	oD
料理	316	185	87	98	2422	4120	4022
食べ物	1121	926	680	246	1527	2867	2621
魚類	633	220	116	104	1612	2256	2152
甲殻類	66	36	15	21	421	642	621
腹足綱	34	20	10	10	333	458	448
二枚貝	35	16	7	9	201	341	332
頭足類	7	3	0	3	144	198	195
棘皮動物	14	11	3	8	77	107	99

上から発見語数の多い順に並べてある．左右に日本語 WordNet と Wikidata の語数を

示す．Synset は日本語 WordNet で見つかる概念の数で，N はそれに紐づく日本語のラベルの数，oN はそのうち日本語 WordNet にのみ見つかる語の数を示す．同様に，Item は Wikidate で見つかる概念の数，D はそれに紐づくラベルの数，oD はそのうち Wikidata にのみ見つかる語の数を示す．C は双方に見つかる語の個数を示す．式に表すと以下のようになる．

$N =$ 日本語 WordNet に登録されている語彙集合
$D =$Wikidata に登録されている語彙集合
$C = N \cap D$
$oN = N \cap \overline{D}$
$oD = \overline{N} \cap D$

獲得できる語彙量 v は次の式で表すことができる．

$$v = N \cup D = oN + C + oD$$

これらの傾向を確認するため oN，C，oD を可視化したグラフを**図 4.3** に示す．

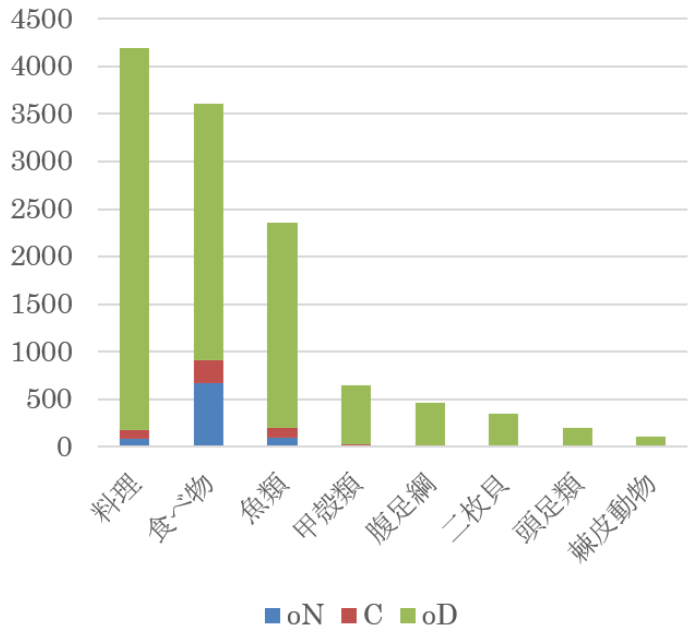

図 4.3　語彙獲得数

魚類全般に対して甲殻類，腹足類，二枚貝，頭足類，棘皮動物は登録量が少ないが，そもそも生物の数として少ないと推察できる．

上位 3 項目の比率を**表 4.6** と，それを可視化したグラフを**図 4.4** に示す．

表 4.6　日本語 WordNet と Wikidata の語彙比率

	oN	C	oD
料理	2%	2%	96%
食べ物	19%	7%	74%
魚類	5%	4%	91%

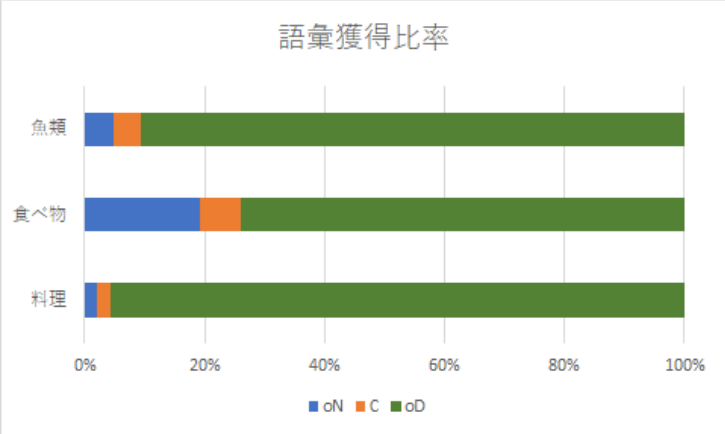

図 4.4　語彙獲得比率

すべての項目で日本語 WordNet が少ないながらも語彙を補完できることがわかる．料理の名称は圧倒的に Wikidata が多いが，食材の名称では日本語 WordNet からの語彙補完量が多い．Wikidata で「これは食材である」という関係プロパティの登録量が少ないことを表している．

4.4　まとめ

本章では，音声対話インターフェースが幅広い語彙を獲得するために利用できるシソーラスの日本語 WordNet，および，オンライン百科事典 WikiPedia のデータをナリッジベース化した DBPedia・Wikidata の活用価値の検証を行った．話し言葉コーパスのテキストから名詞を 2 個か 3 個ならべた語において，日本語 WordNet 上で 0.8%，DBPedia で 2.6% が発見されることがわかり，複合語の語彙知識を得るために有用であることを示した．日本語 WordNet と Wikidata から見つけることのできる料理，食材，魚介類の語彙の比較では，Wikidata の語彙が圧倒的に多いながらも，日本語 WordNet のみに見つかる語もあり，相互補完性が確認できた．

これら二つの検証によって，日本語 WordNet と Wikidata を語彙獲得のために利用する価値と，双方同時に使うことの価値を示した．

深層学習 NER における長い未知語問題

第 5 章

本章では，深層学習 NER の未知語推定において，学習時に含まれていたものより長いものに対して性能が悪いことを指摘し，語彙特徴量を使用することが有用であることを検証した結果を説明する．

5.1 はじめに

固有表現抽出の研究において深層学習を利用したものが最新の研究として多く報告されている．本章では深層学習を使う固有表現抽出を深層学習 NER と呼ぶ．実社会において新語が次々と現れる中，クリエイターたちは複数の語を使って「聞いたことがない」題名を作品につけようと努力する．そういった題名は異なる品詞を混ぜて徐々に長くなる傾向にあると考えられる．そういった語は，過去の経験，すなわち「学習データから得られる特徴量」からの推定は難しく，深層学習だけでは正しく処理できないことが推測される．そのような場合「その名称を知識として知っている」，すなわち語彙が重要となる．

深層学習 NER において，認識の際に，未知の名前や長い名前がエラーの原因になることが多いという問題に着目する．ここでいう「未知」とは学習データに含まれていない固有名を指し，「長い」とは，既知の名前よりも語数が多い固有名を指す．このような未知の長い名前は，小説，マンガ，アニメ，映画などの題名によく見られる．本論文ではこのような名前を長い未知語として UnknownLonger と呼ぶ．以下のような場合，実際のシステムでは，UnknownLonger の一部が正しく抽出されないことが想定される．

1. 既存の題名リストを含むテキストで深層学習モデルを訓練する．
2. 入力テキストから既存の題名，および既存の題名と似たような題名を高い精度で抽出することができる．

3. 新たに，より長い題名が発表された場合，正しく抽出されない．

本章では，深層学習 NER において，UnknownLonger の問題が実在することと，改善するために語彙特徴量を付与することの有効性を示し，実社会で対策が必要な場合が実際にあることを示す．

5.2 UnknownLoger 問題の検証概要

本節では，深層学習 NER において，UnknownLonger の問題が実際に起こること，語彙特徴量で改善できること，実社会で考慮が必要なこと、といった仮説と検証方法について説明する．

5.2.1 三つの観点

実験の設計にあたっては，典型的な NER モデルが実際に UnkonwLonger 問題を持つことを確認する必要がある．そのため，用意したデータセットでうまく動作する典型的な NER モデルを用意し，意図的に UnknownLonger 問題を発生させる．精度測定の際のテストデータの分割方法に着目することで，意図的に問題を再現する．NER モデルの一般的な性能測定では，テストデータをランダムに分割する．その場合，UnknownLonger の問題は隠蔽されてしまう．実験では，長い題名を含む文のみをテストデータとして分割することで，UnknownLonger 問題を再現している．問題が再現するはずであり，それを観察し，次に，語彙特徴量を追加することで，改善効果があることを確認する．一連の実験を通して，高い精度を持つ NER モデルが UnknownLonger 上では精度が低下すること，また語彙特徴量によって精度が大きく向上することを確認することを目的とする．

NER の最新の手法の多くは，最も高い精度を持つ深層学習を利用する．ほとんどの場合，入力特徴量として単語の並びだけを基準にしている．それらの手法では，未知の長い複合語を認識することに問題がある．小説，漫画，アニメ，映画などの作品題名には，長い題名が存在する．この現象を以下の 3 つの観点から検証した．

観点 1

標準的な深層学習 NER がどのように長い題名の問題を抱えているかを検証．

観点 2

語彙特徴量を加えることで性能が向上すると仮定しその効果を検証．

観点 3

長い題名が，現実の既存の作品題名の中にどのように分布しているか確認．その検証結果を報告するとともに，対策を検討する必要性を提言．

本研究の目的は，深層学習 NER を対話型インターフェースで使用する場合，特徴量に語彙情報を付加することが必要であることを主張することである．深層学習 NER テキストから UnknownLonger 名称を抽出するために，語彙特徴量を加えることの有効性を検証する．

5.2.2 検証手順の概要

本手法の有効性は，漫画，小説，アニメ，映画の題名を集めたオリジナルデータセットを用いた実験で測定する．比較のために，日本語の題名と英語の題名を使用した．なぜなら，長い題名は日本語でより頻繁に発生すると考えられるからである．このデータセットとして，実際の題名を持ついくつかの音声テキストパターンを用いて作成されたものを使う．テキストパターンは手動で作成し，題名は Wikidata から収集した．

本研究では，次の 3 ステップでこの問題を検証する．

検証 1: 問題の実在

深層学習モデルを準備し正常に動作すること (テスト 1)

データの選別を変更し，意図的に問題を起こすこと (テスト 2)

検証 2: 語彙特徴量の追加の効果

語彙特徴量を追加するようにジョブを修正し問題が改善すること (テスト 3)

検証 3: 分布の調査

時系列および種類ごとに，作品題名の分布を可視化，国籍やタイプごとに長さの比較

詳しい検証方法を続く節で説明する．

5.3 問題の定義

5.2 節で述べたように，標準的な NER 手法では，単語や単語列の表記のみを入力特徴量として使用する．そのため，UnknownLonger の題名にラベル付けエラーが発生することがある．

訓練データで訓練された NER の分類モデルを考える．下線部は題名で，アプリケー

ションによっては引数として抽出する必要がある．

- I want to watch **Star Wars** next week.
- When **Harry Potter** be released?

題名の単語数はともに 2 つで，その単語はすべて名詞である．

この学習データで学習したモデルでは，「Star Trek」が学習データになくても，次のようなテキストから「Star Trek」を正しく抽出できる可能性がある．

- I want to watch **Star Trek** next week.

これは未知語抽出である．しかし，題名が長い場合に，学習済みモデルは題名の抽出に失敗する．次に長い題名の例を示す．

- I saw **The Bridge on the River Kwai** yesterday.

題名には 6 つの単語があり，単語は定冠詞・名詞・前置詞・定冠詞・名詞・名詞である．これは上述の例よりも複雑で長い．これは分類器を混乱させる可能性があり，このような状況が問題となる．マンガ（コミックブック），アニメ（カートゥーン），小説，映画などの著作物は日々追加されている．しかし，計算コストが高いため，分類モデルを毎日訓練することはできない．

深層学習の一般的な精度評価の方法は，データセットを訓練用とテスト用に分ける際にランダムに 7 対 3 等に分けて評価する．ほとんどの題名が訓練データに含まれているため高い評価を得られるが，UnknownLonger 問題を隠してしまっている．

対象となる題名の単語数で任意に分割されたデータセットで検証を行う．本研究では，短い題名のみを訓練に使用し，残りの長い題名をテストに使用することで意図的に問題を再現する．

NER タスクにおけるこの問題の存在は，後の実験結果で検証する．

5.4　語彙特徴量の追加手法

深層学習 NER の未知語抽出の精度を改善する目的で，識別器の入力に対して，語彙情報をバイナリフラグとして追加することを提案する．この手法は，すでに gazetteer の NER を改善するために提案されている [35]．入力された特徴量に，論理的なバイナリフラグの特徴量を追加する．このフラグは，ある単語がデータベース等，語彙に存在するかどうかを示すものである．学習データにおいては，ラベルからフラグを生成することができる．テストデータにおいても，ラベルからフラグを生成することができる．本番入力データでは，データベース内の単語列を検索してフラグを追加することができる．

図 5.1 に，特徴量にどのようにフラグが付加されるかを示す．

input	feature tensor						b	label	label tensor			
I	0	1	0	1	0	0	0	O	1	0	0	0
saw	0	0	1	1	1	0	0	O	1	0	0	0
the	1	0	1	0	1	0	1	B_MOVIE	0	1	0	0
bridge	0	1	1	1	0	0	1	I_MOVIE	0	0	1	0
on	1	1	1	0	1	0	1	I_MOVIE	0	0	1	0
the	0	1	0	1	1	1	1	I_MOVIE	0	0	1	0
river	1	1	1	1	0	1	1	I_MOVIE	0	0	1	0
kwai	0	1	0	1	1	0	1	I_MOVIE	0	0	1	0
yesterday	1	0	1	1	1	0	0	O	1	0	0	0
.	0	0	0	0	1	1	0	O	1	0	0	0

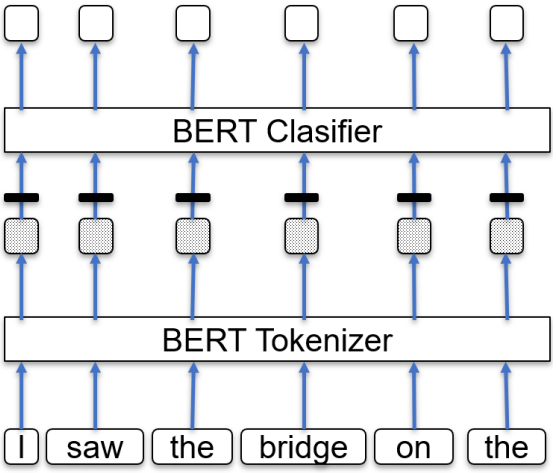

図 5.1　題名に語彙特徴量を付加する

この例で「The Bridge on the River Kwai」が映画の題名である．カラム”b” が入力に語彙情報を付加するバイナリ特徴量フラグを示す．バイナリ特徴量フラグは，ラベル（MANGA，NOVEL，ANIME，MOVIE など）ごとに次元を用意する．このフラグは，学習時とテスト時に，B-MOVIE と I-MOVIE のラベルから生成することができる．本番環境では，データベースから名前を検索してフラグを追加する必要がある．より正確には，単語が分散表現に変換された後に，テンソル行列にフラグが追加される．

5.5　検証 1 および 2 の実験内容

テスト 1 とテスト 2 の検証では，新たに用意したテストデータを用いて実験を行う．この実験の目的は，典型的な NER 法では UnkownLonger に問題があり，語彙特徴量を加えることで改善できることを検証することである．実験の手順は以下の通りである．

- (テスト 1) 典型的な有効な深層学習モデルを準備し，正しく動作することを確認する．
- (テスト 2) テストデータの選択方法を変え，スコアが下がることで実際に問題が発生することを確認する．
- (テスト 3) 語彙特徴量を追加することで改善できることを確認する．

1 つ目の検証は，UnkownLonger の影響により (テスト 1) の結果が (テスト 2) の結果になること，2 つ目の検証は，語彙機能の効果により (テスト 2) の結果が (テスト 3) の結果になることである．実験環境のより詳細な説明は続きの項で説明する．

5.5.1　実験用データセット

長い題名が問題なので，特別にデザインされたデータセットを使う．データセットは以下の 2 種類のテキストを組み合わせて作成する．

- 口語文型（パターン）
- 作品題名

対象分野は，マンガ，小説，アニメ，映画の題名とする．話し言葉は，各分野ごとに 20 個ずつ手作業で作成する．リスト 1 は，映画の文脈での発言の例である．

リスト 1. 口語文型（パターン）

```
I can’t wait until % is released
Will you go see % next week?
I need to buy a ticket for %
The movie % will be coming out in theaters
% world premiere
```

Wikidata の各エリアから題名を収集する．Wikidata から作品題名を収集する際に，instans_of（P31）プロパティを使用する．例えば，映画の題名を収集する場合，ウィキペディアの映画アイテムは wd:Q11424 である．

Wikidata は，SPARQL[54] クエリを使用して検索できる．コード 1 に，ID とラベル名を持つ米国の映画題名のムービーアイテムを収集するために使用した SPARQL クエリを示す．

コード 1. 米国産英語の映画題名を収集する SPARQL クエリ

```
SELECT ?item ?itemLabel
WHERE {
  ?item wdt:P31 wd:Q11424;
  wdt:P495 wd:Q30.
  SERVICE wikibase:label
   {bd:serviceParam wikibase:language ”en”. }
}
LIMIT 500
```

このクエリにおいて，P495 は「原産国」，Q30 は「米国」，Q17 は「日本語」を表す．SERVICE 引数は，ロケール「en」のラベルを持つアイテムに制限する．LIMIT が含まれていない場合，クエリはタイムアウトすることがある．このクエリ例では，行数を 500 行に制限している．リスト 2 に映画の題名の例を示す．

リスト 2. 映画題名の例

```
The Brain That Wouldn’t Die
Puppet Master: The Legacy
Puppet Master 4
The Lusty Men
Curse of the Puppet Master
```

表 5.1 に，Wikidata から題名を収集するための SPARQL クエリで使用されたアイテムのリストを示す．

表 5.1　List of id and target labels of Wikidata

Type	instance_of(P31)	Japanese	USA
Manga	wd:Q21198342	MANGA	
TV Animation	wd:Q63952888	ANIME	
Written work	wd:Q47461344		NOVEL
Animated series	wd:Q581714		ANIME
Movie	wd:Q111424	MOVIE	MOVIE

英語と日本語で異なるデータセットを使用する．映画，アニメ，マンガは日本語のものを使う．日本ではアニメが最初に制作され，それを元にテレビアニメや映画が生まれるのは一般的だからである．しかし，英語で「comic book series」のインスタンスを集めようとしたところ，172 件しか集まらず，十分な数とはいえない．そこで「written work」（Q47461344）を選んだところ，6,000 以上のインスタンスアイテムを集めることができた．これは，小説と同義であり，これを採用することとする．

パターンと題名を例にして，最初のパターンと最初の題名を組み合わせると，データセットの最初の行は以下のようになる．

I can’t wait until The Brain That Wouldn’t Die is released.

3 つの分野でそれぞれ 500 題名，20 ステートメントを使用し，英語作品と日本語作品で作成した合計 30,000 行のステートメントを使用する．

5.5.2　ツール

実験環境には，既存のコンポーネントを使用する．最先端 NER 技術として BERT Tokenizer および BERT Classifier を選択した．**表 5.2** に，実験に使用した環境とツールの一覧を示す．

表 5.2　環境とツール

Computing environment	Google Colaboratory
Platform	Python3.0
Tool	TensorFlow 2.0
Distributed Representation	BertTokenizer BertJapaneseTokenizer
Classifier	TFBertForTokenClassification
Pretrained model	bert-base-uncased(for English) cl-tohoku/bert-base-japanese-whole -word-masking(for Japanese)
batch size	32

計算環境として Google Colaboratory[8] において，Python3 と，TensorFlow 2.0[1] を利用する．入力テキストのトークンをコード化する分散表現として，英語作品では BertTokenizer と，事前学習モデル bert-base-uncased を使用する．日本語の作品には， BertJapaneseTokenizer と，東北大学が公開している事前学習モデル bert-base-japanese-whole-word-masking を使用する．分類には TFBertForTokenClassification を使用する．

5.5.3　実験手順

実験は，次の手順で行う．

1. データセットの準備
   - 文型（パターン）の準備
   - 小説 (米国)，マンガ（日本），アニメ，映画の題名を Wikidata から収集する
   - パターンと題名を混合して，文章とラベルを生成する
2. NER ジョブを準備する
   - TensorFlow
   - BERT Tokenizer(英語 / 日本語)，BERT Classification
3. 一般的なテスト（テスト 1）
   - データセットを無作為に 7:3 に分割

- NER モデルが正しく動作することを確認（高い精度）

4. 未知の長い語の問題を確認（テスト 2）
   - データセットを短い語を含む文と長い語を含む文に分割
   - 問題が発生（精度が低下）することを確認
   - この結果がベースラインとなる
5. 語彙特徴量の効果の確認（テスト 3）
   - 入力特徴量に語彙フラグを追加
   - ベースラインからの差を確認（精度が回復）

5.5.4 語数を利用したデータの分割

題名の集合は，各題名に含まれる単語の数を用いて，学習データとテストデータに分けられる．題名に含まれる単語数を N とし，以下のものを実験用の学習データとして使用する．

- 英語題名：N <3
- 日本語題名: N <4

データセットが 7：3 に分かれるように，これらの数字を選定した．これは，英語の題名が日本語の題名よりも短いことを示しているとも言える．

5.6 検証 1 および 2 の結果

表 5.3 に実験の結果を示す．

表 5.3 実験結果（テスト 1,2,3）

		count of data		lexical		F1 score			
language	division	training	testing	feature	Epoc	NOVEL	ANIME	MOVIE	weighted avg.
English	random 7:3	21264	9114		20	0.997	1.000	1.000	0.999
	train w/N <3	21198	9180		15	0.7987	0.7593	0.8111	0.7889
	train w/N <3	21198	9180	added	24	**1.0000**	**0.9998**	**0.9959**	**0.9988**
language	division	training	testing	feature	Epoc	MANGA	ANIME	MOVIE	weighted avg.
Japanese	random 7:3	21264	9114		20	0.997	0.998	0.986	0.994
	train w/N <4	21771	8607		6	0.5837	0.6212	0.3045	0.4605
	train w/N <4	21771	8607	added	47	**0.9932**	**0.9820**	**0.9734**	**0.9803**

本節では，その結果について述べる．各言語のテスト結果の各行は，5.5 節で述べたテ

スト 1，テスト 2，テスト 3 の結果を表している．

5.6.1 分類ジョブの準備

表 5.3 の各言語の最初の行は，標準的な方法でデータセットを 70% に分けて訓練を行い，30% に分けてテストを行ったテスト 1 の結果である．加重平均の F1 スコアは，英語で 99.9%，日本語で 99.4% であった．この結果はほぼ完璧であり，データセットが恣意的に用意されたことにより，分類タスクにとって簡単すぎる可能性がある．しかし，このデータセットにおいて用意された分類ジョブが正しく動作することが確認できた．

5.6.2 問題の実在の検証

表 5.3 の各言語の 2 行目は，題名の単語数という基準でデータセットを分割するという従来の方法で訓練ジョブを行ったテスト 2 の結果である．題名の単語数が，英語は 3 語以下，日本語は 4 語以下のテストデータ行を訓練データとし，残りのデータをテストデータとして使用した．F1 スコアの加重平均は，英語で 78.9%，日本語で 46.0% であった．これらの結果から，UnkownLonger の題名だけをテストすると，分類精度が低下することが確認できた．差を見ると，日本語の方が英語よりも UnknownLonger 題名の影響を強く受けていることを示している．これをベースラインとする．

5.6.3 語彙特徴量の効果の検証

表 5.3 の各言語の 3 行目は，題名の単語数という基準でデータセットを分割する語彙特徴量を追加した訓練ジョブであるテスト 3 の結果を示している．データ分割の基準はテスト 2 と同じものを使う．F1 スコアの加重平均は，英語で 99.8%，日本語で 98.0% であった．これは，この仕事とデータセットに対して，語彙特徴量の追加が有効であることを示している．

5.6.4 メカニズム

図 5.2 に，検証 1，検証 2 のためのテスト 1，テスト 2，およびテスト 3 を可視化したグラフを示す．

UnknownLonger のテストでは，明らかに精度が低下していることが確認でき（検証 1），語彙特徴量を追加することで改善することが確認できた（検証 2）．

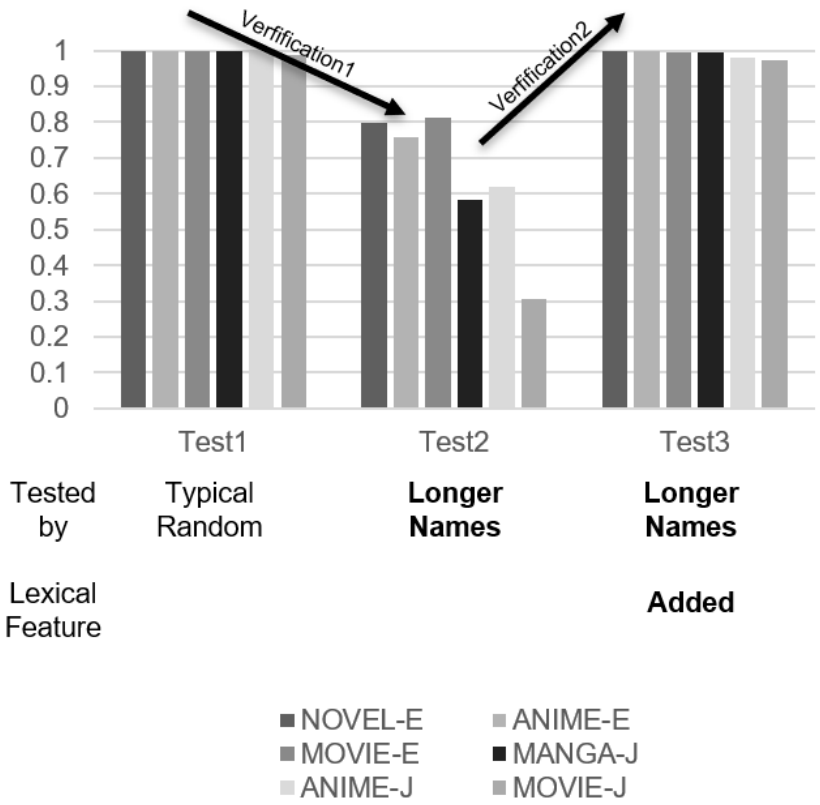

図 5.2　検証 1 と 2

この実験から，問題が存在することを確認し，語彙機能を追加することで改善できることを確認できた．ここで，メカニズムについて議論する．

一般的な NER は，入力として単語列のみを使う．学習時にラベル付けされた名称は辞書として学習済みモデルに格納されるが，未知語推定には周辺語のみが用いられる．UnknownLonger 題名は，文章の抜粋のように見えることが多く，周辺に含まれる単語が含まれていることもある．その場合，名称と周辺語句の境界が極めて曖昧になる．名称と周辺語の境界の可能性を強く示唆するバイナリベクトルとしての語彙特徴量に識別器が強く反応し，問題解決に大きく寄与していると考えられる．

5.7　検証 3 作品題名長さの分布

3 つ目の検証として，題名の長さの分布を調べる．題名ごとに単語数と品詞の種類をカウントする．短い題名のみで深層学習モデルの訓練を行うための語数の決定時に，日本語

のほうが長かった．日本語の方が英語よりも UnknownLonger 題名の影響を強く受けている．これらのことを背景に，日本産の日本語の題名と，米国産の英語の題名の，著作種類および時間軸での分布を検証する．

表 5.4 に作品題名の品詞の例を示す．

表 5.4　品詞の例

The	Bridge	on	The	River	Kwai
定冠詞	名詞	前置詞	定冠詞	名詞	名詞

単語の数は 6 であり，品詞の種類は，冠詞，名詞，前置詞で：品詞数は 3 である．

英語題名では，単語はスペースで区切ることができ，品詞は Apache OpenNLP[3] で品詞推定できる．日本語題名では，日本語形態素解析器 Janome で単語分離し，同時に品詞が推定される．

語数と品詞数を年別に集計し，長さの傾向を確認する．また，作品種類別に集計し，どの種類の作品が長い名前なのかを確認する..

5.7.1　長さが増加

図 5.3 に，公開された年ごとに作品題名の語数と品詞数を可視化したものを示す．

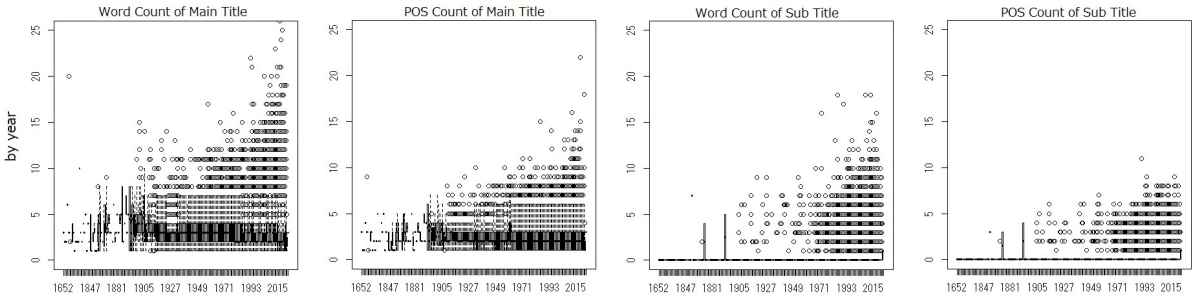

図 5.3　年別の題名の語数と品詞数

左から主題の語数，主題の品詞数，副題の語数，副題の品詞数で，年ごとの箱ひげ図である．これより，作品題名の長さが年々長くなっている傾向がわかる．この傾向は，日米ともに主題名，副題名ともに同じである．このように，作品題名の長さが年々増加していることが確認できた．これは，認識モデルの訓練後に UnknownLongers が出現する可能性があることを意味する．

5.7.2 日本のマンガや映画の題名は他より長い

図 5.4 に，作品の種類ごとの単語数や品詞数の分布を可視化したものを示す．

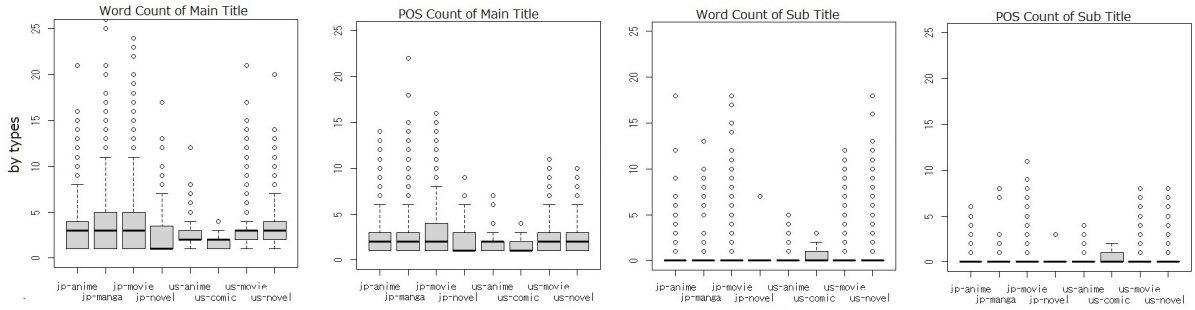

図 5.4　種類ごとの題名の語数と品詞数

表 5.5 に日本産と米国産の作品題名長さを，作品種ごとに独立したサンプル t 検定した結果を示す．

表 5.5　日本産と米国産の作品題名長さの T 検定結果 (P=0.05)

	animation		manga		movie		novel	
	ja-JP	en-US	ja-JP	en-US	ja-JP	en-US	ja-JP	en-US
Mean	3.05	2.74	**3.49**	2.04	**4.00**	2.78	2.83	3.17
Variance	5.89	2.57	10.45	1.08	10.76	1.82	7.59	2.71
Count	1225	139	1295	26	6978	65371	192	7153
t	2.00		6.56		31.03		-1.72	
P($T \leqq t$)	0.023		6.24E-08		7E-199		0.043	

グラフを観察すると，jp-manga，jp-anime の箱が少し高い位置にあることがわかる．t 検定の結果，は日本語のマンガや映画の題名が英語の題名よりも有意に長いことを示している．これにより，**表 5.3** において，日本のマンガや映画の題名の精度が強く影響を受けていることの理由が裏付けできる．

5.8　まとめ

深層学習 NER における長い未知語問題の検証についてまとめる．

5.8.1 検証のまとめ

本章では深層学習 NER において長い未知語に対して未知語推定は問題があること，語彙特徴量を追加することで問題が解決できることを検証した．長い未知語問題は，著作物の題名，マンガ，アニメ，小説，映画などによく現れ，特に，日本語のマンガやアニメで顕著であることを示した．

BERT 分類器の精度は，未知の単語の長さに影響され，語彙特徴量を追加することで回復できることを検証した．既存の作品の題名を見ると，日本のマンガや映画の題名は英語のものよりも長く，作品の題名の長さは年々増加している．したがって，UnknownLonger 題名の識別精度を向上させるためには，語彙特徴量を追加するなどの対策が必要であると考えられる．特に，マンガや映画などの邦題には注意が必要である．

5.8.2 課題と展望

Boolean フラグを BERT の入力特徴量に与える方式が効果的であることを示したが，1 形態素に対してフラグを与えた場合に，対象でないものにもタグを生成してしまう要因になる副作用があると考えられる．この問題を解決するためには，語彙特徴量を長い語のみに与えるなど正しく利用する方法を研究する必要がある．

テキストインターフェース，とくに音声対話インターフェースでは，著作物の題名などの長い固有名詞をヒトは正確に完全に発声せず，略語や短縮された名前が使われる．今後，人が長い題名を省略したり短くしたりする傾向を調べ，語彙情報を用いて入力テキストから短縮された名前を認識する手法を研究する必要がある．

GPT-3 や GPT-J を利用する NER においても，長い未知語問題は依然として存在すると考えられ，学習コストの問題はむしろ増大していると考えられる．GPT-3 や GPT-J に関しても，本論文で指摘している問題が同様に存在すると仮定し，提案している手法が有効かどうか確認する必要がある．

日本語形態素文字種境界 (JMCTB) 法の提案

第 6 章

本章では，テキスト中に目的となる語の候補が含まれるか検査する際に，データベース等の検索回数を抑制するための手法「日本語形態素文字種境界（JMTCTB）法」の提案および効果検証について説明する．

6.1 はじめに

4 章にて，Wikidata のような LOD を活用することの価値を示した．5 章にて，深層学習 NER に語彙特徴量を注入することの有用性を示した．これらから，音声対話インターフェースの入力テキスト内に語の候補があるかオンラインで検索する必要があると言える．

音声対話インター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語彙を獲得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，シソーラスデータベースや 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．

この提案技術の思想の背景には，ヒトが知らない語をネット検索する行動にヒントを得ている．ヒトは検索する前に検索すべき箇所をいくつかの情報をヒントに候補化する．周辺語と文の形の他に，日本語であれば文字種は大きなヒントになる．固有表現抽出処理で周辺語が大きく作用していると考えられるが，JMCTB 法は，それに対してさらに文字種

の活用を与えるものである．

6.2　固有名詞抽出手法の現状

NLP において，固有名詞抽出は研究対象として盛んである．本節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，解決すべき課題を本節で述べる．

6.2.1　文字列比較やパターンマッチングの問題と課題

文字列比較や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．

6.2.2　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．

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み合わせ数が大きくなり，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ー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，一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．

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，以下のように抽出される箇所を示す．

- 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**

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．

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．

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[29] 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一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．

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．

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[20] 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，適切な抽出とは言えない．

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．

6.2.3　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 [66]．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．

6.2.4 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 [68] もある．理解しやすいように，**表 6.1** 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示す．

表 6.1 形態素解析と文字種切り出しの比較

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	になあれ
な	読
あれ	んでたなぁ
読ん	
で	
た	
なぁ	

文字種だけを使う場合（表中右側），境界が一致せず，「あした天気になあれ」を取り出しにくいことがわかる．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．

6.2.5 深層学習の問題と課題

学習データを使った深層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 [73] 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み合わ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．

6.2.6　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，小説題名，漫画題名，映画題名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:　「文節のような題名」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深層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わかる．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．

6.2.7　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．

6.2.8 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ー辞書を生成して追加することができる．ユーザー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ーが非力で実メモリーも少なかった．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ー上での小型化，アクセス効率化などを考慮した結果，メモリー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，一旦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ー辞書更新の回避
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ー辞書を更新する方法を回避する．

6.2.9 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，時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 [45]．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．

6.2.10 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ーが発話入力することが考えられるからである．ユーザーとなるヒトは通常，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ーが発したときに，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．抽出出来た場合「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．

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．もっと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．

6.3　課題に対する解決策

6.2 節にて，パターンマッチング，N gram 文字列による検索，深層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ー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 6.1** に示す．

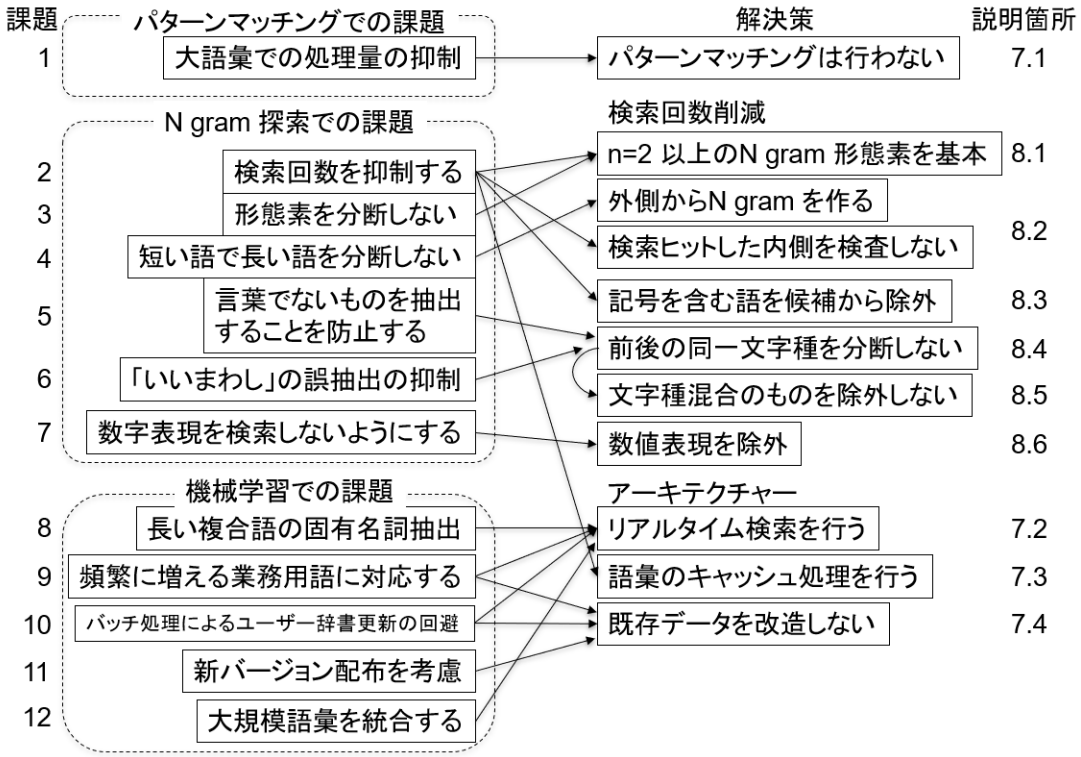

図 6.1　課題と解決策

課題は，パターンマッチングでの課題，N gram 検索での課題，深層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ーの考え方と，検索数削減手法にグループ化できる．それらを 6.3 節および 6.4 節でそれぞれ説明する．

6.3.1　提案アーキテクチャーの考え方

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，語の候補を抽出し検索を行うアーキテクチャーを設計する．そのアーキテクチャーを**図 6.2** に示す，

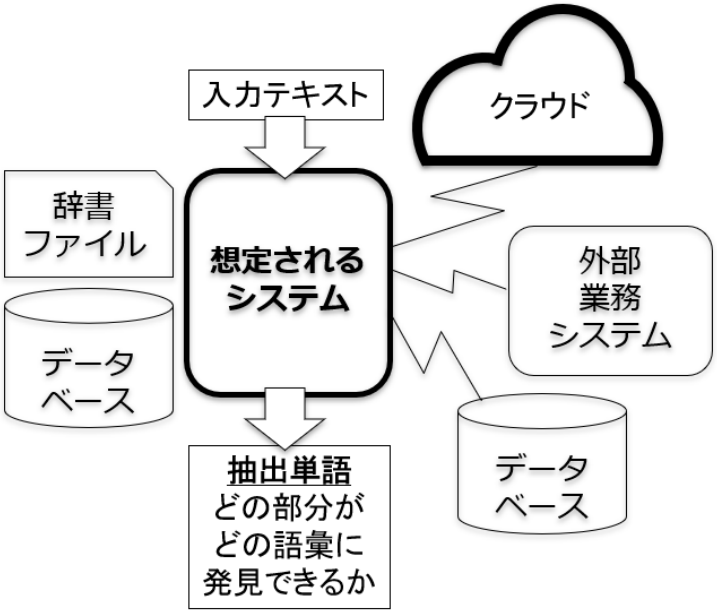

図 6.2　想定されるシステム

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 サービスなどに問い合わ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，アーキテクチャーを設計する上での指針を説明する．

6.3.2　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．

6.3.3　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，普段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資源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．本研究の実験では日本語 WordNet と Wikidata を対象とする．

6.3.4　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 などは外部資源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．

6.3.5　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，ユーザー辞書，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．

6.4　検索回数抑制手法

アーキテクチャーとして，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．

6.4.1　[解決策 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 6.2** に示す．表示例では冒頭部分のみを示している．

表 6.2　N gram 文字と N gram 形態素の比較

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
：	：

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．

6.4.2　[解決策 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み合わせから N gram の組み合わせを作る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 6.3** に示す．下線のある箇所は，Wikidata で見つかる語である．

表 6.3　N gram を内側からと外側から行う比較

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	

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．

6.4.3 [解決策 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，日本語 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，音声対話型インター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」のような映画題名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．

6.4.4 [解決策 4] 前後の同一文字種を分断しない

6.2.2 項で述べた「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 6.5 節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．

6.5 節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．

6.4.5 [解決策 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ーのことである．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，6.5 節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

6.4.6 [解決策 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ーを作成した．これにヒットするものは検索対象から除外する．

6.4.7 アルゴリズム

[解決策 1] から [解決策 6] 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．

6.5 実験

6.4 節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

み合わ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み合わ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，その正確性は次の三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，三種の実験を実施した．

6.5.1 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 [59]
  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，日本語 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．

6.5.2 実装

実験に使用したシステムの構成は以下の通りである．

- 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

- シソーラスアクセス：JAWJAW 1.0.2[63]
- LOD アクセス：Apache Jena 3.14.0[30]

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ひとつに対して，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み合わせを日本語 WordNet や Wikidata からどれくらい効率よく発見できるか，を検証する．**図 6.3** は今回の実験システムの構成図である．

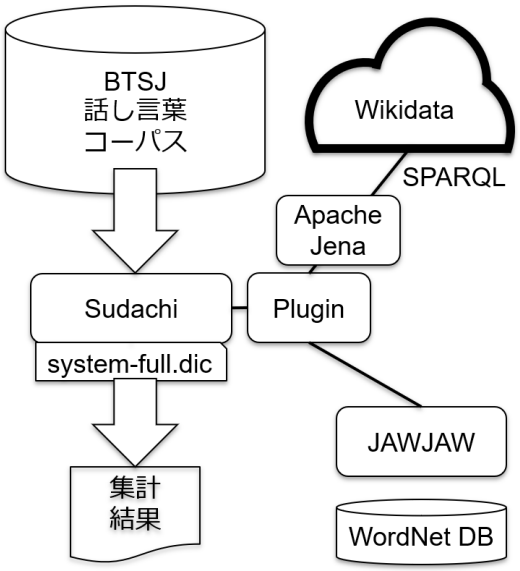

図 6.3　実験システムの構成図

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み合わせを Wikidata に問い合わ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い合わ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．

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
```

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．

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ー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い合わ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ー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→日本語 WordNet キャッシュ→日本語 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，日本語 WordNet より優先して検索すべきだが，できるかぎりメモリー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.

6.5.3 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 で言及）が起きている箇所を計測.

N gram の形態素から，4.4 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」「日本語 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 [解決策 4] の対策が有効であることを示す.

実験 2 Wikidata のみ，日本語 WordNet のみ，Wikidata を検索したあと日本語 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日本語 WordNet および Wikidata を検索するように作られている．解決策 [解決策 2] はあらかじめ含まれている．[解決策 1]，[解決策 3]，[解決策 4]，[解決策 5]，[解決策 6] を一つずつ追加し，その都度の検索回数，抽出数すなわち日本語 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ー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，キャッシュも含め日本語 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 2.20GHz
- メモリー：SODIM DDR3 1600MHZ 16GB

6.5.4 実験結果

コーパスの談話テキストを処理し，JMCTB 法によって検索すべき候補として抽出され，実際に日本語 WordNet の word または Wikidata の label か altLabel に見つかる語の典型的な例を**表 6.4** に示す.

表 6.4　JMCTB で候補となり語彙に見つかる語

語彙セット	周辺語を含む形態素列
wikidata	で, も, さ,、,**格安, 航空券**, と, か, あっ, たら, ね
wikidata	んーっと,、, 何, だ,、,**門前, 仲町**,、, の, あたり, 笑い
wikidata	首, 切ら, れ, たら,、, さー,、, 教職,、,**教員採用, 試験**, 受けれ, ば, いい, し
wordnet	5,0, キロ,**制限, 速度**, で,、, しかも
wikidata	すご,、,**血縁, 関係**, と, か,、, 家系, を, 重んじる
wikidata	ずっと, 高校, の, 時, から,**日本語, 教師**, に, なり, たく, て
wikidata	国内, に,、,**映画, 産業**, が, ある, から
wikidata	そう,、,**サッカー, ボール**, と, か, 持っ, て, き, ちゃう
wikidata	お母さん, が,**ジャイアント, コーン**, が, 好き, で
wordnet	なん, か,、,**オーラル, コミュニケーション**, と, か,=
wikidata	なん, か,**エントリー, シート**, が, めんど, くさく, って
wikidata	で, なん, か,、,**イアン, ・, ソープ**, が, さ
wikidata	これ,**マイクロ, フォン**, に, 紙, を
wikidata	あの,**あん, ドーナツ**,、, の
wordnet	しか, もー,、,**2, 階, 建て, バス**, な, ん, や, けど
wikidata	えーとー,、,**揚げ出し, 豆腐**, と,、, 味噌汁
wikidata	私立, は,、, やっぱり,**遠い, 道のり**, だ, な
wikidata	やわらちゃん, が,**ある, 日, 突然**, さ,、, 整形, し, たら

太文字の下線が引いてある箇所が，複数の隣接する形態素で構成され，日本語 WordNet や Wikidata で発見できる語である．前後の文字種との境界がどのような状況であるかわかりやすいように，近接する数語の形態素も併せて掲載している．上から順番に「漢字のみの塊」「カタカナの塊」「異なる文字種が混じるもの」の順に並べてある．前後に異なる文字種がある漢字やカタカナの塊ができている部分はひとつの語として成り立ちやすいことがわかる．最後の二つはそれぞれ映画，曲の題名と同じ文字列である．

6.5.3 で説明した 3 つの実験の結果を示す.

6.5.5 実験 1

表 6.5 に実験結果を示す．**表 6.5** では，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.

表 6.5 同一文字種を分断する検索ノイズ

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**

6.5.6 実験 2

表 6.6 に実験結果を示す.

表 6.6 検索回数と抽出数

	Wikidata			日本語 WordNet			Wikidata → 日本語 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
[解決策 1] 1 形態素を除外	16,723,965	7	226	16,633,131	7	294	16,623,145	7	495
[解決策 3] 記号を含む語を除外	1,308,405	93	218	1,302,935	93	198	1,301,675	93	391
[解決策 4] 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
[解決策 5] 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**
[解決策 6] 数字表現を除外	695,857	96	181	692,181	96	160	691,159	96	319

各行は第 6.4 節の解決策を順番に適用して変化を確認するように書かれている．列は Wikidata のみを検索した場合，日本語 WordNet のみを検索した場合，Wikidata →日本語 WordNet の順に検索した場合を示している．列「検索回数」は Wikidata や日本語 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．日本語 WordNet や Wikidata に見つかった語の数とも言える．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 形態素を含める全ての形態素組み合わせを取り出して検索をした時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．

6.5.7 実験 3

表 6.7 に実験結果を，**図 6.4** に各ステップ追加後の処理時間のグラフを示す．

表 6.7 各ステップ追加後の処理時間

	処理時間	一行あたり
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
[解決策 1] 1 形態素を除外	9,974	93
[解決策 3] 記号を含む語を除外	9,275	87
[解決策 4] 前後の文字種が同一のものを除外	9,392	91
[解決策 5] 混合文字種のものを除外しない	9,367	90
[解決策 6] 数字表現を除外	18,346	183
	ミリ秒	マイクロ秒

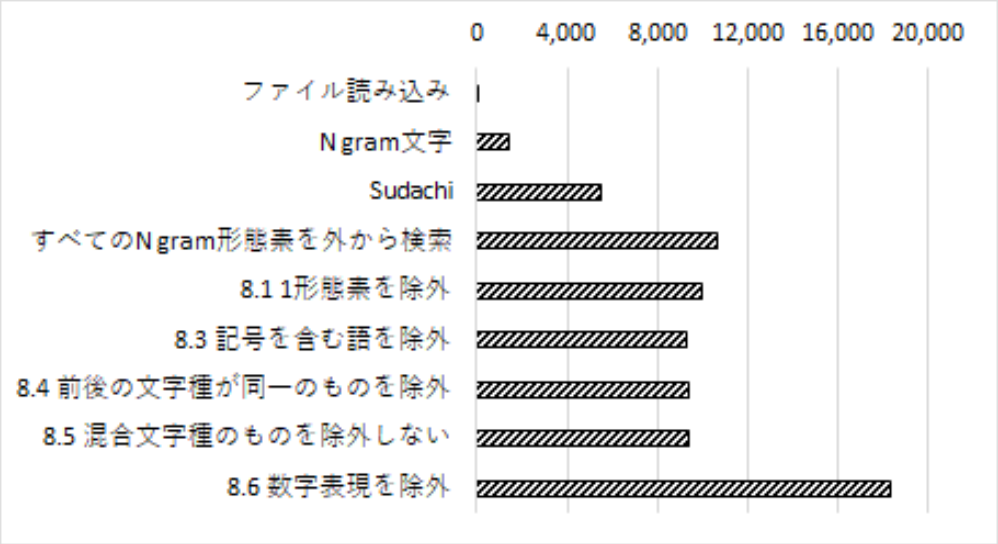

図 6.4　各ステップ追加後の処理時間

「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．

6.6　考察

表 6.5 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日本語 WordNet や LOD で発見できた語（[解決策 5] 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わかる．

表 6.6 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．

[解決策 1] 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 1800 万回弱から 1700 万回弱程度まで検索回数を減らすことができる．

[解決策 3] 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 1500 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．

[解決策 4] 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わかる．このため [解決策 5] 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」という二つの工夫によって，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，この二つの工夫が JMCTB 法の性能に大きく貢献することがわかる．（表中，太文字下線の数値）

[解決策 6] 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」などが挙げられる．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，などが理由として挙げられる．この対策として，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ー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詳細に調査しておらず触れていない．語彙資源に見つかる語であることを示す正解ラベルがコーパスに存在しないため，抽出精度を測ることが困難だからである．

抽出された語の一覧を目で検査したところ，[解決策 4] 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「日本語 WordNet や Wikidata に見つかる」ことを示しており，正しい検索と言える．

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 システムへの検索にとどまっている．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 への問い合わ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．音声対話型インター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，妥当な数値と言える．

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，日本語 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．

6.7 まとめ

日本語形態素文字種境界（JMCTB）法の提案と価値検証についてまとめる．

6.7.1 検証のまとめ

音声対話型インターフェース，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．それは，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」である．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 検索する時に比べて検索回数 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．

テキストから JMCTB 法によって「日本語 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，日本語 WordNet，Wikidata，業務データベースを活用できると考えている．

シソーラスの日本語 WordNet，LOD の Wikidata の利用価値が示すことができ，作品題名のような長い語を語彙を使ってラベルづけする方法の有用性を示し，その語彙特徴量を作るにあたり検索量を削減する手法として JMCTB を提案し有用性を示した．これらの統合指針を示すことで，音声対話インターフェースに LOD を含む外部データの語彙を動的に統合することに貢献できたと考える．

6.7.2 課題と展望

今回の JMCTB 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」にも同様のことが言える．追加的に除外する方向で実装を進めていく．

深層学習の入力に，文字種情報を特徴量として与えることで精度に寄与するかどうか将来試してみたい．

統合アーキテクチャー

第 7 章

本章では，本論文で説明している 3 つの研究成果を総合的に使うということはどういうことか，総合的に使うことでどのような効果が得られるかを説明する．

7.1 はじめに

第 4 章，第 5 章，第 6 章の三つの研究により，音声対話インターフェースの入力テキストの語の抽出処理において，以下を提案し，有効であることを示した．

- 複数の語彙資源を同時に利用すること
- 深層学習に外部語彙を注入すること
- 語彙検索時に検索候補を絞ること

第 4 章，第 5 章，第 6 章で述べた研究成果は独立しておらず，それぞれが相互に関連しており，相乗効果をもたらす．それを説明するためには，音声対話インターフェースの要件，「幅広い語彙の獲得」「常時継続稼働」「新語の即時反映」「応答性能」の四つを思い起こす必要がある．

- 「語彙増強にシソーラス，LOD を使う，同時に使う」だけで，従来の手法で識別モデルを更新していては，シソーラスが更新されたとき，あるいは LOD に語が新規登録されたとき再学習が必要で「常時継続稼働」「新語の即時反映」を満たせない．
- 「深層学習 NER に語彙情報を特徴量として追加する」だけでは，ローカルコピーされた語彙から語彙特徴量を生成している時「新語の即時反映」を満たせない．LOD など外部データベースを検索するとき，検索回数が多すぎて「応答性能」が満たせない．

- JMTCB 法は語の候補を即時検索することに役立つが「語彙の候補である」ことを処理できる固有表現抽出器がなければ役に立たない.

それぞれ独立では達成できないが，三つの研究成果を同時に活用することで要件が満たされるようになる．本章では，それぞれの研究成果を統合して同時に利用することで四つの要件が満たされたアーキテクチャーとなることを説明する.

7.2　アーキテクチャー

三つの研究成果を同時に使うアーキテクチャーにより，どのように音声対話インターフェースの要件が満たされるかを説明する.

7.2.1　アーキテクチャーの概要

図 7.1 に三つの研究成果を統合して同時に活用する音声対話システムのアーキテクチャー例を示す.

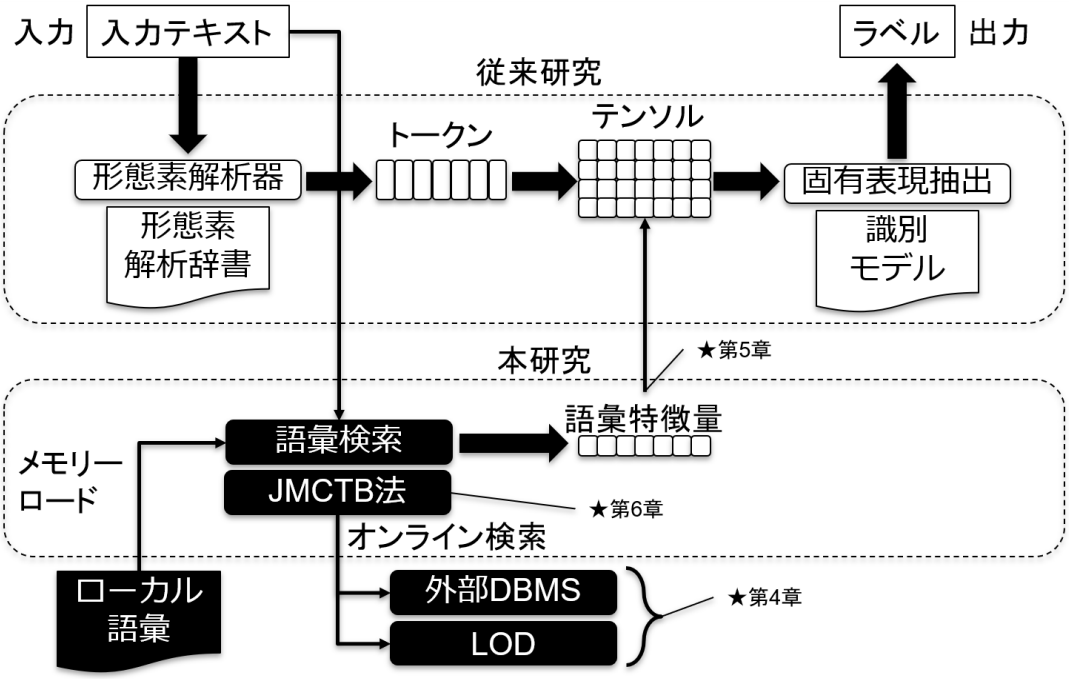

図 7.1　総合したアーキテクチャー

図の上半分は従来研究による固有表現抽出の処理方法を示す．形態素解析器と深層学習 NER 識別器のみの従来研究どおりで，形態素解析器により文字列をトークン化し，その結果を入力し，ラベル付き学習データで学習した NER 識別器により語の抽出を行う．この場合の語彙は識別モデルに内包されており，語彙を増強するには再学習によって識別モデルを再構築する．

図の下半分に本論文で提案している手法を適用した仕組を示し，関連する箇所に，該当する章を示す．本研究で提案するアーキテクチャーでは，複数の語彙セットを活用し，深層学習 NER に語彙情報の特徴量を注入する．

7.2.2 深層学習 NER への外部語彙の利用

語彙情報の特徴量は，入力テキストのフラグメントを順次データベースで検索することで生成され，バイナリフラグとして NER 識別器に入力される．この方法は，第 5 章「深層学習 NER における長い未知語問題」の実験で効果を確認した．これにより「幅広い語彙の獲得」の要件を満たす．

7.2.3 複数語彙資源の直接利用

前節で説明したように，本研究で提案するアーキテクチャーでは外部の語彙情報を特徴量として与える．語彙情報の特徴量を生成するにあたり，外部の語彙を直接活用する．外部の語彙資源から，学習データを作って識別モデルを訓練することは避ける．ただし，ある程度の期間に収集された新語の情報をまとめておき，定期的に再学習を行うなどを行うこともできる．そうすることで，最新の語は外部語彙として語彙特徴量とされ，再学習された語は深層学習 NER が高い精度で抽出できる．

外部の語彙として，CSV ファイルや EXCEL データなどで提供されるビジネスデータ，ビジネスシステムに稼働する外部 DBMS，日本語 WordNet のようなシソーラスすなわち辞書データ，Wikidata のような LOD が考えられる．本研究ではそれらの語彙は更新されることを想定し，それらのデータを直接システムに結合する．たとえば，CSV ファイルなどはメモリーに読み込んでインデックス化し，ビジネス DBMS は SQL を使い ODBC 経由で直接検索し，日本語 WordNet は API で直接検索し，Wikidata は SPARQL 言語を使いインターネット経由で直接クエリする．これにより，システムは次のような語彙を同時に活用してスロットフィリング処理を行うことができる．

- 形態素解析辞書の語彙

- 深層学習 NER の学習データのラベルによる語彙
- テキストファイル，CSV ファイル，EXCEL ファイルなどのビジネスデータ
- SQL などのビジネス DMBS
- 日本語 WordNet などのシソーラス
- Wikidata などの LOD

直接利用することで「新語の即時反映」の要件を満たすことができ，再学習を避けることができるため「常時継続稼働」の要件を満たすことができる．

7.2.4 語彙検索の削減技術

6.3 節で説明したように，深層学習 NER に語彙を注入するにあたり，テキストに含まれるフラグメントが語彙にあたるかどうかを検索によって調べる必要がある．ビジネスデータ，シソーラスは語の抽出処理を行うシステムに内包すれば高速に処理可能である．ビジネスデータベースや LOD は，場合によってネットワーク経由で動的に検索処理する必要がある．従来手法のように総当たりの組み合わせでは検索回数が膨大になり，処理が短時間で終わらない．短時間で処理を終えるため，語彙として検索にヒットすると推測できる組み合わせを事前に絞りだし，優先的に検索したい．そこで日本語特有の文字種を活用する，日本語形態素文字種境界（JMCTB）法を利用する．本研究で提案するアーキテクチャーでは，外部検索を行う語彙システムおいて JMCTB 法によって検索する対象を絞って検索を行う．次のような使い方が考えられる．カッコ内の時間は例である．

1. **深層学習 NER でラベル付けを行う**
   ラベル付けが成功すれば即答する．（1 秒未満）
2. **ローカル語彙で特徴量を与える**
   ラベル付けが成功すれば即答する．（1 秒未満）
3. **JMCTB で絞り込んだ候補を外部語彙で検索した特徴量を与える**
   「お待ちください」など返答したあと，検索を行う．（数秒程度）
   検索結果がヒットし，対話が成り立てば語彙をローカル語彙に追加する．
4. **JMCTB で排除された組み合わせを検索する**
   「申し訳ありません，わかりません」など返答したあと，検索を行う．（1 分以上）
   見つかった場合「もしかして，XXX のことですか」などを発話する．
   対話が成り立てば語彙をローカル語彙に追加する．

上記の処理は，たとえばヒトが誰かに話しかけられて回答しようとしたとき，相手に悟られないようにインターネットを検索して情報を確認し，あたかも既に知っていたかのように回答する様子と似ている．深層学習 NER に語彙特徴量を追加するにあたり，直接検索をしたいが検索回数が多いと「応答性能」の要件が満たせない．JMCTB がそれを助け「応答性能」を満たす役割を果たす．

7.3　期待される効果

第 7.2 節で説明したアーキテクチャーを利用することで，どのような効果が得られるかを説明する．

7.3.1　語彙エラーの削減

前述のアーキテクチャーの例では，日本語 WordNet と Wikidata を用いて説明しているが，複数の語彙資源を統合することで，語彙が増大する．ビジネスシステムにおいて，商品リストなどを使って語彙を準備することがあるが，その語彙の範囲を超えた語をヒトが話すことで処理エラーが発生する．そのようなとき，日本語 WordNet や Wikidata のような辞書，辞典などの語彙が役立ち，エラーを回避することができる．

新しくできた店，始まったばかりのテレビシリーズ，公開されたばかりの映画題名など，名称は次々と生まれる．とくに作品題名は長くなる傾向が示され，そのような場合に新しく発生した題名を「この部分が映画題名」という未知語推定できる精度が下がる傾向を実験で明らかした．そのような場合に，Wikidata のように日々更新され新語に強い語彙資源をオンライン接続し，そこから得られる語彙情報を深層学習 NER に語彙特徴量として流し込むことで，新しい題名の抽出の精度が向上する．

7.3.2　第三者提供語彙の利用

スマートスピーカーを提供するクラウドベンダーは，クラウドシステム上に語彙処理機能を提供している．クラウドベンダーは，スマートフォンや検索エンジンを提供しており，それらから収集した語彙を活用しやすい．それに対してビジネスシステムで音声対話インターフェースを構築する場合，そういった大規模語彙が企業にはないため，なんらかの方法で語彙を増強する必要がある．とくに，社内用語や技術用語ではない一般用語の場

合，百科事典などの語彙は欠かせない．シソーラスや LOD を同時利用することでビジネスシステムにも大語彙を統合して利用することができる．

7.3.3 再学習の回避

深層学習 NER への語彙特徴量注入や検索量削減は，いずれも NER 識別器の外部語彙をオンライン統合することを想定している．オンライン統合するため，ビジネスデータ，シソーラスなどは新版が提供されたとき，単にファイルを置き換えるだけでよい．ビジネス DBMS のデータベースや LOD のデータは日々頻繁にデータが更新されることが想定されることがあるが，更新され追加された語は即時利用可能となる．新語が追加されるたびに再学習処理を行う必要はない．再学習による副作用の心配や，再学習時の多大な計算コストも削減される．

7.3.4 外部語彙発見の履歴活用

外部語彙検索に伴い対話が成り立ったと評価できたとき，あらかじめ保持していない語が使われたとシステムが判断することができる．この情報は，タイムスタンプを付与して管理することで，新しい話題，新語，といった情報の構築に役立つ．エビングハウスの忘却曲線 [15] のような理論を活用することで対話の豊かさを表現することに役立つ．

たとえば，初めて検索にヒットした語は「新しい話題」として記憶に定着し，しばらくの間優先順位を保つ．「この間の映画なんだけど」のようにユーザーが話しかけたとき「映画ＸＸＸですね」と思い出すのに役立つ．その後，しばらく話題にあがらない場合，優先順位を徐々に下げ「忘れた」ことにする．「この間の映画なんだけど」と話しかけれたら「どの映画のことでしょうか」としらばっくれたような返答をしてヒトらしさを演出することができる．

7.3.5 知識を活用した入力文の解析

形態素解析や従来の深層学習 NER のラベル付けはフラットな処理で，シソーラスやセマンティックを使うことで知識との融合が図れる．深層学習 NER で，未知語ではあるがラベル付けが成功したとする．たとえば I-MOVIE / O-MOVIE の組み合わせでラベルが生成されたとする．その場合 NER が出力した結果は「この箇所はおそらく映画題名」という情報である．しかし，語彙注入による NER で出力されるラベルには，最初からオントロジー上のアイテム（日本語 WordNet であれば SynSetID，Wikidata であれば

ItemId）が紐づけられる．これによって，より正確に語義を特定でき，後段の処理に多いに役立つ．

7.3.6　モバイルバックエンドの軽量化

音声対話インターフェースは，口と耳で操作できることから，手と目を別の用途に使っているような場面でのニーズが多い．手袋をしているような手が汚れる作業中や，徒歩，自転車や自動車の運転中などである．作業中のデータの閲覧，記録，機械の操作などに利用される．そのような場面ではしばしばモバイルアプリケーションに音声対話インターフェースを実装する．

モバイルアプリケーションのうち，一部の重い処理，大量データを扱う処理，他のユーザーとのデータ共有などの処理を行う部分をサーバーやインターフェースに置くことをモバイルバックエンドという．スマートフォンやタブレットに搭載されるモバイルアプリケーションでは，クラウドにモバイルバックエンドを置くアーキテクチャーが今日では一般的である．

言語処理のシステムは比較的大型になりやすく，小規模なアプリケーション展開ではモバイルバックエンドはコスト的に負担になりやすい．バックエンドにあるデータベースやLOD の語彙をオンライン検索することができれば，モバイルバックエンドを通さずにモバイルアプリケーションから直接検索することができ，小規模モバイルアプリケーションの展開に役立つ．

7.4　まとめ

本章では，3 つの研究成果をまとめて一つのアーキテクチャーとして使うことによる効果を説明した．

7.4.1　本章のまとめ

複数の語彙資源を直接的に利用し，深層学習 NER の入力に特徴量として語彙情報を追加し，その語彙情報の検索を JMCTB によって検索削減する，というアーキテクチャーを示した．このアーキテクチャーにより，深層学習の欠点の一つである再学習や追加学習を回避でき，新語が即時反映され，応答時間への制約を守ることができることを説明した．

7.4.2 課題と展望

商品リスト外の名称，新しく発生する作品題名などの処理に JMCTB 法で削減した検索により語彙特徴量を生成し，深層学習 NER 入力に追加することの価値を説明したが，実際にどの程度そのような問題があり，どの程度改善されるのか測定できていない．実社会で効果を測定するには，長時間稼働する社会実装が必要で，今後の課題である．実際のシステムが稼働することでデータが取得できれば，その効果を測定する必要がある．

結論

第 8 章

最後に，本論文の全体をまとめ，今後の研究課題と展望について述べる．

8.1 本論文のまとめ

本論文では，音声対話インターフェースの語彙不足を原因としたエラー軽減を目的に，音声対話インターフェースの要件を満たす方法で語彙獲得を行うための研究成果を述べた．音声対話インターフェースには「幅広い語彙を獲得」「常時継続稼働する」「新語を即時反映する」「応答性能を確保する」という要件があることを示した．音声対話インターフェースの要件に対して，語彙獲得をするための次の三つの研究を提案した．大語彙を獲得するための資源としてシソーラスの日本語 WordNet，LOD の Wikidata を利用する価値の検証，深層学習 NER の長い未知語の対策，即時検索の削減手法の日本語形態素文字種境界（JMCTB）法の提案である．

言語資源の利用価値の検証において，形態素解析だけでなく外部語彙が必要であること，日本語 WordNet と Wikidata には相互補完性があり同時に使う価値があることを示した．深層学習 NER の長い未知語問題の検証において，語彙情報をバイナリ特徴量として入力に付加することで性能が向上すること，著作物の作品題名では実際に考慮が必要であることを示した．JMCTB 法の提案においては，形態素解析と文字種の検査を組み合わせることで，総当たりを行うときと比べて 96% の検索量を削減できることを示した．

これら三つの研究は独立したものではなく，相互に関係しており，総合的に活用することで音声対話インターフェースの要件に合致するような統合アーキテクチャーで語彙獲得ができることを説明した．本論文が報告する三つの研究成果を総合して使う統合アーキテクチャーによって，四つの要件を満たしつつオンラインで語彙獲得ができることを示した．

8.2　今後の研究課題と展望

最後に，今後の研究課題と展望を述べる．

8.2.1　音声認識精度への適用

音声認識の出力テキストでは，依然として間違ったテキストが出力されることはめずらしくない．とくに日本語は同音異義語が多いため，正しくない語が選択されることがある．音声認識器にも言語処理器が内包されており，そこに固定化された語彙が含まれており，語彙が不足していることもある．本論文の成果を適用しても音声認識の精度が向上することには貢献しない．本論文では音声認識のテキスト出力の精度に対して，語彙を増強することを対象としていない．クラウド API や市販の音声認識ソフトウェアの言語処理部は外部から修正が困難であることが理由である．将来，言語処理に適用するための本論文の原理を音声認識部にも適用し，音声対話システム全体の性能を向上することに寄与したい．

8.2.2　AI を教育する

現在流行している AI 技術の多くが，深層学習に傾倒している．その中で認識モデルは train（訓練）される．それは動物の神経系において「短期記憶」に基づくものと似ている．訓練することで犬などが訓練主にとって都合がよい判断と行動をすることと同じである．やってごらん，と言われてすぐできるわけではなく，訓練には時間がかかる．

深層学習 NER は，従来の手法では訓練データ，すなわち経験情報のみから固有表現を抽出しようとする．語彙特徴量を追加する手法の場合，語彙，すなわち「整理された知識」を活用して固有表現を抽出しようとすることが異なる．

次の世代の AI は，educate（教育）される時代に入ると考えている．そのためには，AI にとっての「教科書」や「百科事典」と，それを学ぶ機能が必要である．それらは，ヒトが読むために作られるものではなく，AI，すなわち IT システムが読むためのものであり，IT が読みやすいものである必要がある．本論文の研究は，そのためのものであり，将来，ヒトが丁寧に AI を教育できるようになることを目指す．特に音声対話インターフェースでは，大規模な文章集コーパスや大規模言語資源から言葉を覚えるのではなく，言葉一つ一つを丁寧に教えて覚えられるようになるべきである．本論文の研究の延長線では，AI を教育する環境を構築していく展望をもち，今後そういった分野に貢献したい．

8.2.3 実用活性化への展望

研究分野において巨大で強力なハードウェアに，膨大なデータを入力して作る大語彙システムがよく作られるが，実用システムにおいてそのような考え方は提案しにくい．本論文で提案している手法とアーキテクチャーは，実運用システムで大語彙を取り扱うためのハードルを押し下げる．様々な機関が用意する大語彙言語資源を有効に活用しながら，大語彙を運用する音声対話インターフェースを実現するきっかけになるはずである．音声対話インターフェースについて実用的な手法を開発できたと考えており，この分野の実利用の拡大が期待できる．本論文をきっかけに音声対話インターフェースの実用が活性化することを望む．

謝辞

本論文は，筆者が公立はこだて未来大学大学院システム情報科学研究科システム情報科学専攻在学中にて執筆したものです．多くの方々のご助言や励ましのおかげで完成することができました．

本研究に取り組むにあたり，最後まで親身なご指導と励ましをいただいた指導教官である公立はこだて未来大学システム情報科学部情報アーキテクチャ学科大場みち子教授に心より感謝いたします．晩年にもなり大学院に進学することを決められたのは，大場教授のお誘いがあってのことであり私にとってかけがえのない転換期のひとつとなりました．

本論文の審査を引き受けて頂きました，同じく公立はこだて未来大学システム情報科学部情報アーキテクチャ学科の新美 礼彦教授，村井 源教授，筑波大学システム情報系志築文太郎教授に心より感謝いたします．

新美教授には，技術の評価方法の相談に繰り返し乗っていただいたり，大規模コーパスの入手にご協力いただいたりなどお世話になりましたことを重ねてお礼申し上げます．

大学院生活後半 2 年は，新型コロナウィルス蔓延により世界中が困難にある中，ご指導賜りました皆さまにおきましてはビデオ会議を通してご対応いただくことができました．ビデオ会議システムの普及と，それに迅速に対応いただいた大学運営にこちらをお借りして感謝いたします．深層学習の研究では Google Colaboratory を活用しました．高額な GPU システムを購入することなく深層学習を含めた研究ができました．Google LLC. に感謝します．

2020 年度大場ゼミの学生の皆さんには，深層学習の入力パターンの作成にご協力いただき，この場をお借りしてお礼もうしあげます．

最後に，晩年にもかかわらず大学院に進学し，職を変え，学位取得を目指すことを温かく見守り励ましてくれた家族に心から感謝します．

業績一覧

学術雑誌（査読有）

- 米持幸寿, 大場みち子: 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. 情報処理学会論文誌, Vol. 62,No. 2, pp. 594–606. (2021).
- Yukihisa Yonemochi, Michiko Oba: Continual Lengthening of Titles: Implications for Deep Learning Named Entity Recognition. IJIS2022. (2022)

国際会議（査読有）

- Yukihisa Yonemochi, Michiko Oba: Verifications of Influence by Unknown Longer Titles of Work on Robustness of Deep Learning NER. Proceedings of IWIN2021 (2021).

国内学会・研究会（査読無）

- 米持幸寿, 大場みち子: 日本語 WordNet と Wikidata の語彙差調査. FIT2021. (2021)
- 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 47.1, pp. 1–7. (2019).

受賞

- IWIN 2021 Excellent Paper Award

参考文献

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al.: Tensorflow: A system for large-scale machine learning, *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283 (2016).

[2] Adam L. Berger, S. A. D. P. and Pietra, V. J. D.: A Maximum Entropy Approach to Natural Language Modeling, *Computational Linguistics*, p. 22(1) (1996).

[3] Apache: OpenNLP, http://opennlp.apache.org (2011).

[4] Apache: lucene-gosen, https://github.com/lucene-gosen (accessed 2019-3-10).

[5] Apple: SFSpeechRecognizer, https://developer.apple.com/documentation/speech/sfspeechrecognizer (accessed 2022-1-17).

[6] Arabiki, T.: 日本語形態素解析の裏側を覗く！ MeCab はどのように形態素解析しているか, https://techlife.cookpad.com/entry/2016/05/11/170000 (accessed 2022-1-17).

[7] Atilika: Kuromoji, https://www.atilika.org/ (accessed 2020-4-7).

[8] Bisong, E.: Google Colaboratory, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, pp. 59–64 (2019).

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

[10] Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 708–716 (2007).

[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] developer.android.com: SpeechRecognizer, https://developer.android.com/reference/android/speech/SpeechRecognizer (accessed 2022-1-17).

[13] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).

[14] Diaz, A.: NLP – What happens in Entity Extraction and its value, https://www.marscrowd.com/blog/text/nlp-entity-extraction-and-its-value/ (accessed 2021-12-27).

[15] Ebbinghaus, H.: *Memory: A Contribution to Experimental Psychology* (1885).

[16] Eddy, S. R.: What is a hidden Markov model?, *Nature biotechnology*, Vol. 22, No. 10, pp. 1315–1316 (2004).

[17] EleutherAI: Mesh Transformer JAX, https://github.com/kingoflolz/mesh-transformer-jax/ (accessed 2021-12-27).

[18] EleutherAI: EleutherAI, https://www.eleuther.ai/ (accessed 2022-1-17).

[19] 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).

[20] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J. and Vrandečić, D.: Introducing Wikidata to the linked data web, *Proceedings of International Semantic Web Conference(ISWC)*, Springer, pp. 50–65 (2014).

[21] Floridi, L., C. M.: GPT-3: Its Nature, Scope, Limits, and Consequences. (2020).

[22] Google: Speech-to-Text, https://cloud.google.com/speech-to-text/ (accessed 2019-3-10).

[23] Hochreiter, S. and Schmidhuber, J.: Long short-term memory, *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780 (1997).

[24] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, Vol. 79, No. 8, pp. 2554–2558 (1982).

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

[26] IBM: Watson Speech to Text, https://www.ibm.com/jp-ja/cloud/watson-speech-to-text (accessed 2022-1-17).

[27] Ichihara, M., Komiya, K., Iwakura, T. and Yamazaki, M.: Error analysis of named entity recognition in bccwj, *Recall*, p. 61 (2015).

[28] IPA: IPAdic, https://ja.osdn.net/projects/ipadic/ (accessed 2019-3-10).

[29] Isahara, H., Bond, F., Uchimoto, K., Utiyama, M. and Kanzaki, K.: Development of the japanese wordnet (2008).

[30] Jena, A.: semantic web framework for Java (accessed 2020-4-7).

[31] KUDO, T.: MeCab : Yet Another Part-of-Speech and Morphological Analyzer, *http://mecab.sourceforge.net/* (2001).

[32] Lafferty, J., McCallum, A. and Pereira, F. C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001).

[33] LeCun, Y. et al.: LeNet-5, convolutional neural networks, *http://yann.lecun.com/exdb/lenet*, Vol. 20, No. 5, p. 14 (2015).

[34] Ma, R.: Using GPT-3 for Named Entity Recognition, https://ricky-ma.medium.com/using-gpt-3-for-named-entity-recognition-83a95389408e (accessed 2022-1-17).

[35] Magnolini, S., Piccioni, V., Balaraman, V., Guerini, M. and Magnini, B.: How to use gazetteers for entity recognition with neural models, *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pp. 40–49 (2019).

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

[37] Microsoft: Speech to Text, https://azure.microsoft.com/ja-jp/services/cognitive-services/speech-to-text/ (accessed 2022-1-17).

[38] NAIST: Japanese Dictionary, https://osdn.jp/projects/naist-jdic/ (accessed 2019-3-10).

[39] NICT データ駆動知能システム研究センター：BERT 日本語 Pre-trained モデル, https://alaginrc.nict.go.jp/nict-bert/index.html (参照 2022-1-17).

[40] OpenAI: OpenAI, https://openai.com/ (accessed 2021-12-27).

[41] Oracle: Java(tm) Platform Standard Edition 8 - Pattern, https://docs.oracle.com/javase/jp/8/docs/api/java/util/regex/Pattern.html (accessed 2022-1-17).

[42] Oracle: Java(tm) Platform Standard Edition 8 - String,

https://docs.oracle.com/javase/jp/8/docs/api/java/lang/String.html (accessed 2022-1-17).

[43] Ramos, J.: Using TF-IDF to Determine Word Relevance in Document Queries. (2003).

[44] Ruder, S.: NLP-progress, *London (UK): Sebastian Ruder* (accessed 2020-01-18).

[45] Sato, T.: Neologism dictionary based on the language resources on the web for MeCab, *https://github. com/neologd/mecab-ipadic-neologd* (accessed 2020-4-7).

[46] Schuster, M. and Paliwal, K. K.: Bidirectional recurrent neural networks, Vol. 45, No. 11, p. 2673–2681 (1997).

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

[48] Shanaz, A. L. F. and Ragel, R. G.: Named Entity Extraction of Wikidata Items, *2019 14th Conference on Industrial and Information Systems (ICIIS)*, IEEE, pp. 40–45 (2019).

[49] Takaoka, K., Hisamoto, S., Kawahara, N., Sakamoto, M., Uchida, Y. and Matsumoto, Y.: Sudachi: a Japanese Tokenizer for Business, *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)* (chair), N. C. C., 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.), Paris, France, European Language Resources Association (ELRA) (2018).

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

[51] Tolmachev, A., Kawahara, D. and Kurohashi, S.: Design and Structure of The Juman++ Morphological Analyzer Toolkit, *Journal of Natural Language Processing*, Vol. 27, No. 1, pp. 89–132 (2020).

[52] Uchida, T.: janome, https://mocobeta.github.io/janome/ (accessed 2021-12-27).

[53] w3c: Resource Description Framework(RDF), https://www.w3.org/RDF/ (accessed 2019-3-10).

[54] w3c: SPARQL Query Language for RDF (accessed 2020-4-7).

[55] w3c: W3C SEMANTIC WEB ACTIVITY, https://www.w3.org/2001/sw/ (accessed 2022-1-17).

[56] Wada, K.: N-gram の作り方, https://qiita.com/kazmaw/items/4df328cba6429ec210fb (accessed 2020-4-7).

[57] Wallach, H. M.: Conditional random fields: An introduction, *Technical Reports (CIS)*, p. 22 (2004).

[58] Zhou, G. and Su, J.: Named entity recognition using an HMM-based chunk tagger, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 473–480 (2002).

[59] 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).

[60] ウィキメディア財団：WikiPedia, https://www.wikipedia.org/ (accessed 2019-3-10).

[61] マーケティング・データ・バンク：MDB 有望市場予測レポート「音声認識システム」を公開，https://mdb-biz.jmar.co.jp/news/20190628 (参照 2022-1-17).

[62] 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 (2016).

[63] 嶋　英　樹：Java Wrapper for Japanese WordNet, http://www.cs.cmu.edu/ hideki/software/jawjaw/index.html (accessed 2019-3-10).

[64] 岡本真由子，重田和弘：音声対話システムにおける話速の検討，*IEICE Conferences Archives*, The Institute of Electronics, Information and Communication Engineers (2018).

[65] 翠　輝久, 河原達也, 正司哲朗, 美濃導彦：質問応答・情報推薦機能を備えた音声による情報案内システム，情報処理学会論文誌，Vol. 48, No. 12, pp. 3602–3611 (2007).

[66] 川口久光，菅谷奈津子，畠山　敦，多田勝己，加藤寛次：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).

[67] 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 (2017)，一般社団法人 人工知能学会，pp. 2J2OS16a4–2J2OS16a4 (2017).

[68] 下畑光夫，杉尾俊之：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).

[69] 米持幸寿：音声対話システム向け意味属性抽出と意図タイプ推定実装小型化，技術報告 5，情報処理学会 研究報告 自然言語処理 (NL) (2018).

[70] 志和敏之，神田崇行，今井倫太，石黒　浩，萩田紀博，安西祐一郎：対話ロボットの反応時間と反応遅延時における間投詞の効果，日本ロボット学会誌，Vol. 27, No. 1, pp. 87–95 (2009).

[71] 西村良太，中川聖一：応答タイミングを考慮した音声対話システムとその評価，技術報告 22，豊橋技術科学大学情報工学系, 豊橋技術科学大学情報工学系 (2009).

[72] 工藤　拓：単語の追加方法, https://taku910.github.io/mecab/dic.html (参照 2020-4-7).

[73] 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).

[74] 坪井祐太，鹿島久嗣，工藤　拓：言語処理における識別モデルの発展-HMM から CRF まで.，言語処理学会第 12 回年次大会チュートリアル資料 (2006).

[75] 東中竜一郎，増村　亮：対話システムにおける深層学習の適用，人工知能， Vol. 34, No. 4, pp. 460–466 (2019).

[76] 富士経済研究所：スマートスピーカーやスマートホーム対応照明の市場を調査，https://www.fuji-keizai.co.jp/market/detail.html?cid=18096&view_type=2 (参照 2022-1-17).

[77] 矢野経済研究所：対話型ＡＩシステム市場に関する調査を実施，https://www.yano.co.jp/press-release/show/press_id/1946 (参照 2022-1-17).

[78] 吉田裕介，萩原将文：複数の言語資源を用いたユーモアを含む対話システム，知能と情報，Vol. 26, No. 2, pp. 627–636 (2014).

[79] 鈴木貴仁，緒方　淳，綱川隆司，西田昌史，西村雅史：咽喉マイクを用いた大語彙音声認識のための特徴マッピングによるデータ拡張と知識蒸留，情報処理学会論文誌，Vol. 62, No. 6, pp. 1373–1381 (2021).

図目次

1.1 対話型 AI システム市場に関する調査（矢野経済研究所） 4
1.2 音声認識システム市場概況（MDB Digital Research） 4
1.3 スマートスピーカーやスマートホーム対応照明の市場（富士経済研究所） 5
1.4 音声対話インターフェース・アーキテクチャー 6
1.5 アプリケーションシステムの言語処理と語彙の例 7
1.6 要件から課題の導出 11

2.1 語彙の管理 17
2.2 形態素解析の例 23
2.3 題名を深層学習で固有表現抽出する 27

3.1 日本語 WordNet と Wikidata のデータ構造 31
3.2 各研究分野と本研究の位置づけ 35

4.1 実験システムのアーキテクチャー 38
4.2 甲殻類の上位概念の探索 42
4.3 語彙獲得数 45
4.4 語彙獲得比率 46

5.1 題名に語彙特徴量を付加する 51
5.2 検証 1 と 2 58
5.3 年別の題名の語数と品詞数 59
5.4 種類ごとの題名の語数と品詞数 60

6.1 課題と解決策 71

6.2 想定されるシステム 72
6.3 実験システムの構成図 80
6.4 各ステップ追加後の処理時間 87

7.1 総合したアーキテクチャー 92

表目次

2.1 スロットフィリングの例 16
2.2 語の抽出の例 18
2.3 N gram 検査の例 21
2.4 固有表現抽出の例 25
2.5 NER の入力と出力 26

3.1 日本語 WordNet 1.1 の語数等 30

4.1 複合語の検証環境 37
4.2 NLP 抽出された名詞と動詞の発見数 40
4.3 2〜3 個並んだ名詞の結合語の発見数 40
4.4 語の検索に利用する識別子 43
4.5 日本語 WordNet と Wikidata の語彙数 43
4.6 日本語 WordNet と Wikidata の語彙比率 45

5.1 List of id and target labels of Wikidata 54
5.2 環境とツール 55
5.3 実験結果（テスト 1,2,3） 56
5.4 品詞の例 59
5.5 日本産と米国産の作品題名長さの T 検定結果 (P=0.05) 60

6.1 形態素解析と文字種切り出しの比較 67
6.2 N gram 文字と N gram 形態素の比較 74
6.3 N gram を内側からと外側から行う比較 75
6.4 JMCTB で候補となり語彙に見つかる語 84

6.5 同一文字種を分断する検索ノイズ 85
6.6 検索回数と抽出数 85
6.7 各ステップ追加後の処理時間 86