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

Proposal for Integration of NLP, Thesaurus, and Ontology on Multi Steps Natural Language Processing

米持 幸寿[1*] 大場 みち子[1]

Yukihisa Yonemochi[1*], Michiko Oba[1]

[1]公立はこだて未来大学
[1]Future University Hakodate

Abstract: On some application systems utilizing combination of Natural Language Processing (NLP), thesaurus, and/or ontology, there are many problems on processing knowledge base of the systems. Text mining, Spoken Dialog System, or Document Classifier can be enumerated as examples of such applications. On processing of those applications, steps of technologies are utilized in same time as a flow of analytics. In that time, some words can be found in Ontology, but not in NLP. This symptom causes processing failure. The purpose of our research is reducing the error rate of the Multi Steps Natural Language Processing. From our investigation, about 60% of nouns cannot be found on WordNet and about 70% cannot be found on DBPedia even if it is extracted in latest NLP when tested using BTSJ dialog corpus data. On the other hand, 260 combined words on WordNet and more than 1,300 combined words on DBPedia can be found even if NLP cannot extract them as nouns. Reducing these differences between processes is important to improve accuracy of language processing. This paper proposes creating a framework to integrate dictionary data for each processor, effectiveness, and its possibility of implementation.

1. はじめに

自然言語処理(NLP), シソーラス, オントロジーといった技術を併用するシステムでは, 知識ベースを活用する上で様々な課題が存在する. この種のシステムとして, テキストマイニング, 音声対話, 文書分類などが例示できる. これらのシステムでは, 分析を行うために同時に複数の自然言語リソースによる多段処理を行う. 本論文ではこれを多段自然言語処理(Multi Steps Natural Language Processing: MS-NLP)と呼ぶ. そのとき, 形態素解析が名詞や動詞として抽出した語がシソーラスやオントロジーに見つからない, あるいは, ある語がシソーラスやオントロジーには見つかるが形態素解析では抽出できない, といった問題が発生し, 処理が失敗する. 本研究では, 多段自然言語処理における処理失敗率の低減を目的としており, 本論文では, 処理プログラムが使うデータを統合することで問題が解決できる可能性と必要性, データ統合の方法を提案する. 本研究では対象を日本語とし, その他の言語への拡張は将来のプランとする.

2. 多段自然言語処理とその必要性

本研究では, 多段自然言語処理での各工程での語彙を問題にしているため, 多段自然言語処理とはなにか, またその必要性について述べる.

2.1 多段自然言語処理の例

自然言語処理は, 音声対話システムの音声入力からのテキスト, ソーシャルネットワークのテキスト, ニューステキスト, 新聞テキスト, 論文テキストなどの処理に使われる. 文章を処理し, 内容からなにかを知り, 情報として取り出すためには様々な処理が必要となる. たとえば, Apache UIMA[1]はテキストや画像など非構造化データと呼ばれるデータを処理するためのフレームワークであるが, タグ付け(Annotation)を処理するプログラムコンポーネントを順次処理するように作られており, 多段自然言語処理が必要であることを示唆している. 表 1 は音声対話システムの多段自然言語処理であり, 音声認識

を利用した音声対話システムの処理手順の一例である．表中の左列は多段で処理される処理内容，右列はその時使われる代表的な辞書データである．

表 1 音声対話システムの多段自然言語処理

処理	辞書データ
音声認識	形態素解析辞書
形態素解析・品詞推定	形態素解析辞書
辞書データ生成	コーパス
固有表現抽出	シソーラス
意味属性抽出	オントロジー
意図推定	表現パターン
アプリケーション処理	セマンティック Web
音声合成	内蔵辞書データ

これらの処理についてそれぞれ解説する．

2.2 音声認識

最初に音声認識が行われる．多くの音声認識時にも言語処理は行われている．音声だけから文字列を生成した場合，正しい言葉は生成されず，特に日本語は漢字に直す必要があるため，単語の並びの頻度を推定して修正をかける処理が行われている．これは「カナ漢字変換」と似た処理である．これによって音声はテキストデータに変換される．この際に形態素解析と同等の処理をしており,形態素解析辞書が使われることがある．

2.3 形態素解析・品詞推定

テキストデータは再度自然言語処理によって形態素（主に単語）に分解され，品詞が推定される．ただし音声認識器によっては認識時に生成された言語解析モデルをそのまま出力することが可能なものもあるため，その場合には自然言語処理をそれに任せることも可能である．（例：Google Speech API[2]）

音声認識や自然言語処理は，今日では CRF（Conditional Random Field : 条件付き確率場）を使って単語の並び順の確率を推定しているものが多い．たとえば，MeCab[3], lucene-gosen[4], Sudachi[5]などがそうである．そのため，これらの形態素解析器が使う辞書は CRF の確率を計算するための入力データとして作られている．CRF の入力データには単語と別の単語の品詞が並ぶ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．パラメーター推定には，当初コーパスが用いられており，収録されている文章から単語並びの可能性が推定（計算）され，辞書データとして生成されてきた．初期のものには IPA-DIC[6], UniDic[7], NAIST-jdic[8]などがあったが，近年，ネットの情報を頻繁に反映させることを始めた NEologd[9]辞書が登場し，新規の単語が追加で定期的に反映されるようになった．

2.4 固有表現・意味属性抽出

形態素解析と品詞推定によって，品詞ラベルづけされた単語列となったデータから，アプリケーションごとに関連する単語が含まれるかどうかを調べる処理へ移る．入力された文がどのアプリケーションに関連するか調べ，特定のアプリケーションを選定するためである．こうすることで，アプリケーションは文の中に登場するすべての単語を処理する必要はない．アプリケーション自体が処理すべき，興味のある語彙の範囲のものだけを処理すればよい．アプリケーションを「時計アプリケーション」「天気予報アプリケーション」と分離開発できるようにすることもできる．

自然言語解析の分野では特定の分野の表現を「固有表現」と称し，その抽出方法を議論している．たとえば「人名」「地名」「時間表現」などである[10]．また，アプリケーションでは特定の種類の単語だけを抜き出したい場合もある．たとえば「くだもの」「道具」「家電」のようなある範囲のものの一般名称であったり，「商品名」，「ブランド名」のような「固有名称」であったりする．そういったものを入力テキストから捕まえる処理を「固有表現抽出」や「意味属性抽出」などと呼ぶ．

固有表現抽出や意味属性抽出においてシソーラスやオントロジーが辞書データとして使われることがある．シソーラスとオントロジーは類似したデータモデルであり，言葉の抽象化概念構造を持つ．たとえば，「りんご」→「くだもの」というような抽象化関係を is-a 関係という．固有表現抽出処理時や意味属性抽出処理時には「りんご」が文の中に発見されたら「くだもの」があった，とタグ付けをするためなどに使われる．

2.5 意図推定

固有表現や意味属性が抽出できると，その周辺にある言葉を組み合わせて「意図」を推定することができる．その方法として，入力として期待する意味属性と文型との照合を行う方法[11]がある．たとえば「今日食べたいものはいくら」を読み取りたいとき「<時間表現>食べたいものは<食品>」であるし，「今日のいくらはいくら」であれば「<時間表現>の<食品>はいくら」となるかもしれない．ここで時間表現＝「今日」，食品＝「いくら」が意味属性として取り出される．そして，前者は「食べたいものの主張」という意図であるし，後者は「食品の価格の問い合

わせ」という意図である.

2.6 アプリケーション処理

システムが推定した意図に適合するプログラムを選択する. これを一般的に「ディスパッチ:dispatch」と呼ぶ. 入力データをディスパッチされたアプリケーションは抽出された固有表現や意味属性を「引数」として受けとり処理することができる. このようにすることで, アプリケーションを言語理解処理から切り離し, アプリケーション処理に特化することができる. 場合によっては,「時計アプリケーション」「天気予報アプリケーション」「経路探索アプリケーション」といった具合に 1 つのシステムに複数のアプリケーションが搭載されていることもある.

アプリケーション処理では様々なデータが使われることになる. 当然企業が持つデータベースなどを使うこともある. 言葉の関連などを調べて処理をする際に Linked Open Data(LOD)といったセマンティック Web のインターネットデータを活用することもある.

2.7 音声合成

アプリケーションは処理が終わると発話するためのテキストを生成する. これを音声合成することでユーザーに話しかける. 現在実在する性能の良い日本語音声合成のほとんどが商用製品である. 音声合成のインプットは通常テキストであり, そのテキストを製品内部で再度言語処理を行い, その結果「どのように発音すべきか」を推定して音声合成している.

この場合の言語処理にはOSSや公開されている言語リソースも使われているが, 企業が内部で独自に開発したクローズドな言語処理エンジンや言語リソースが使われていることもある.

商用ソフトウェア製品の場合はオープンソースになっていない限り, 内部でどのような処理を, どのような内臓辞書データを使って行なっているかは明らかにならないし, それをカスタマイズすることは困難である.

ただし, 言語の区切り箇所や, イントネーション（強さ・弱さ・音程）を発音の調整データとして付加することができる製品も存在する. このような場合, システムからデータとして渡すことができる可能性を秘めている.

3. 多段自然言語処理の問題点

各工程が利用する辞書データには,2 章で説明したようなテキストの多段自然言語処理で発生する多くの処理エラーの原因となる問題がある.

3.1 辞書データの語彙網羅性欠如

入力文に含まれる単語が, 多段自然言語処理においてすべての辞書データにすべて存在している場合, テキストは処理がうまくいく. しかし, 辞書データ語彙が世の中にある言葉をすべて含むことは現在では不可能である. 当然, それぞれの辞書データには欠けているものが存在しており, 欠けている単語を含む文の処理は失敗する.

音声対話システムのような処理では, 音声認識, 形態素解析を実施した時点では単語が認識され, アプリケーションデータに存在しない場合でもその単語の意味属性を正しく読み取ることができれば, うまく処理することが可能な場合がある.

たとえば, スーパーのアプリケーションを作成する場合に, 花の名前を考えてみる.「薔薇はいくらですか」という問いに対して「一本 500 円です」と回答できることを想定してみる. 百合を商品として取り扱っていない場合「百合はいくらですか」に対して「百合は取り扱っていません」と回答すればよい. しかし,「うしのしたはいくらですか」と質問したら「100g あたり 400 円です」と回答してしまうかもしれない. ここで「うしのした」とは牛肉のタンのことではなく花である. それは「肉のことですか, 花のことですか」と聞き返すこともできたかもしれないが, 辞書に載っていないかぎり「うしのした」は牛肉のタンでしかない. さらに, 形態素解析と品詞推定によって「うし (名詞)」「の (接続詞)」「した (名詞)」となる可能性が高く, その場合, 名詞として取り扱われない.

この例のように, 多段自然言語処理において前段にあたる処理, すなわち, 音声認識, 形態素解析, 品詞推定, 固有表現抽出, 意味属性抽出において語彙不足がアプリケーション処理にとって大きな足枷になっていることがある.

3.2 アプリケーションデータの不在

アプリケーションには特有の語彙が必要なことが多い. これらは先に登場したような一般的な辞書データには存在せず, アプリケーションが独自に提供する必要がある. とくに商品名や社内用語などは一般辞書データには登場することを期待できないため, アプリケーションが用意することになる. そしてそれらは当然のことながら事前に用意されている形態素解析辞書やシソーラスには掲載されていない. これにより, アプリケーションに処理が移るまえに言語処理部で処理を失敗してしまうことになる.

3.3 データモデルの不一致

多段自然言語処理で利用される言語処理に必要な言語リソースはデータモデルがそれぞれ違う．このため，別の段が使う辞書データを簡単には転用できない．例えば，DBPedia に掲載されている単語を形態素解析で識別しようとしてもそのままでは使えない．それには処理の目的の違いが背景にある．

1) 形態素解析と品詞推定

日本語形態素解析と品詞推定には「語の切れ目と並び」が重要な情報となる．近年の形態素解析処理では CRF でこれらの処理を行うためである．

2) シソーラス

シソーラスデータ WordNet では個々の言葉 word を重複無しに管理するために synset ID と呼ばれる識別子を振り，概念接続 Sense，概念 Synset，概念間の関係 Synlink などのデータから成っており，概念間はその link で接続されており，Synonims（同義語），Hypernims（上位概念），Hyponims（下位概念），Instances（実例）などといった 25 種類の決められた関係を構築している．

3) DBPedia

DBPedia[12]は主にオントロジーデータとして設計されているため，OWL[13]や RDF[14]の考え方を取り入れている．WordNet[15]が一般常識的な概念を定義しようとしているのに比較して，現実世界の具体的な実例に基づくデータを比較的ゆるい概念と，他の個体との関係として示しているデータとなっている．また，ソースデータが Wikipedia[16]であることからボランティア的に入力され審査されたデータであるため，間違いや抜けも多い．インターネット情報として作られていることもあり，あいまいなリンク情報も大量に結び付けられていることから，ソフトウェアで解析して意味を捉えることに使うには障壁が高い．

このように形態素解析辞書データ，シソーラス，オントロジーには異なったデータモデルと構造が利用されており，相互にデータを交換することはできず，知識を再利用するには障壁が存在する．

3.4 リアルタイム追加処理不可

形態素解析と品詞推定では，語の並びを，確率場を使って推定している．語の候補を検索する必要があるため，効率的に検索できるようインデックスデータが必須である．csv 形式のソースデータから作る辞書データは*.dic ファイルというバイナリーファイルである．そのため，インプットとなる語の並び順を記述したデータからインデックスを生成し確率推定したデータをあらかじめ作っている．これは辞書データのコンパイルとして知られている．これはプログラム起動時にバッファーを作りそのまま読み込むものである．処理速度を上げるためにバッファー上のポインターやオフセットでアクセスできるようにすることが目的であり，バッチ処理的に構築する．このため，辞書データに新規語彙を追加するためには全体のデータに新規データを追加し，コンパイルし直してから再起動する必要がある．

ユーザーデータを追加する機能もあるが，再起動が必要なことに変わりはない．

WordNet のデータも同様で，テキストベースではあるが，ソースデータをファイルにコンパイルしたものであり，バッチ処理的に構築する．

これらの特徴から，形態素解析やシソーラスの言葉の追加にはコンパイルといったバッチ処理と処理プログラムの再起動が必要である．

3.5 API としての更新不可

3.4 で説明したとおり，形態素解析辞書とシソーラスはファイルに単語が登録されていることもあり，プログラムから API で語彙を登録することができない．

4. 関連研究

旧来の日本語形態素解析において，辞書データは IPA-DIC, UNIDIC, NAIST-jdic といった CRF パラメーター推定辞書が長きにわたり作られてきた．しかしどれも一時的なプロジェクトにより作られてきたもので，一度作られると保守が止まり，新しい単語が追加されなくなる．NEologd が登場したことにより，新しい単語が Web から拾われ，追加されるようになった．

WordNet 英語版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(Magnini ら [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(Agirre ら[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(Suchanek ら[19])などがすでに行われている．

しかしこれらの研究は，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．

本研究では多段自然言語処理される各プロセスの辞書データを横串に統合することを試みることであり，既存の研究とは違うアプローチといえる．

5. 各処理段での登録語彙の調査

多段自然言語処理における辞書データの統合による効果を測定するため, 各処理段で使われる辞書データに含まれる単語の網羅性を計測することとした. これは 3.1 で説明した「辞書データの語彙網羅性欠如」に該当する. 世の中にある言葉すべてを使って語彙の網羅性を検証するのは困難である. そのため, 特定のコーパスデータを入力とすることとした. 特定の形態素解析で処理した結果から名詞と動詞のみに対してシソーラス検索, オントロジー検索を行い, 各処理段で使われる辞書データに各語彙が見つかるかどうかを集計することとした.

5.1 対象とするソフトウェア

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, 保守などが行き届いている点などを考慮して, 以下のソフトウェア, およびデータを統合処理の対象と考えている. 主にオープンソースソフトウェアで構成されている.

- 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

ただし, 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.

本研究では, 以下のソフトウェア, およびデータを統合処理の対象と考えている.

5.2 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が必要となる. 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してみることとする.

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

5.3 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 1 に示す.

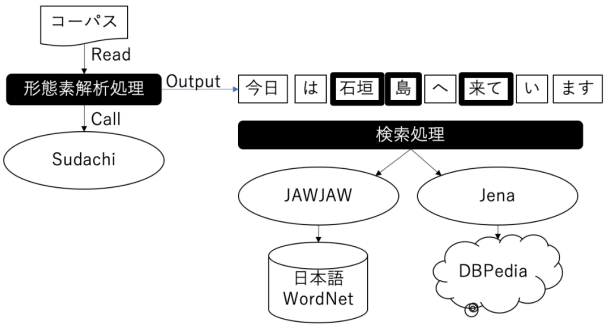

図 1 実験システムのアーキテクチャー

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. コーパスデータの読み取り
   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, 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.

5.4 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, 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で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:

同様に, 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.

5.5 調査結果

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 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 2 NLP 抽出された名詞と動詞の発見数

	品詞	All	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, 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 WordNet と DBPedia で探してみた結果を表 3 に示す. こちらのデータでは, 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みつらないはずだからである.

表 3 2~3 個並んだ名詞の結合語の発見数

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

5.6 語彙調査の考察

表 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 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.

表 3 の調査結果において, 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 WordNet から 200 以上, DBPedia においては 1000 以上が見つかる. 表 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.

このような方法を用いて複合語候補に後段でタグ付けを行う手法は以前から知られている. バッチ処理なら有効であるが, ロボット対話システムのようなレスポンスを重要視する場面ではこの処理自体に時間がかかりすぎてしまい, システム全体の性能を落とすことになる. また, 係り受け解析をする場面においても形態素解析で処理できることが望ましい.

6. 問題に対するアプローチ案

5 章では, 多段自然言語処理において, 辞書データの登録語彙に関して問題があることを示した. 今後, 本研究では, 次のようなアプローチでこの問題を解決することを検討している.

6.1 見つかるべき語の抽出

今回の集計では, 見つからなかった単語が見つかるべきだったかの調査が不十分と考えており, 精査が必要と考えられる. 今後, 数千語に及ぶ単語から, 調整すべき語を抽出する方法を検討し, 抽出できるようにする.

6.2 バッチ的に辞書データを補完

初段処理のエンジンとして Sudachi および NEologd を今回の対象技術としている. この二つの技術では辞書を再コンパイルするという方法で単語登録をアプリケーションごとに追加することができる機能が備わっている. この機能を利用して単語登録をして機能すること, およびその場合の課題点などを明確化していく.

6.3 オンデマンドで追加

Sudachi にはプラグイン機能が備わっている. システムとして後段処理のシソーラスやオントロジーに発見される単語をタイムリーに形態素解析処理に適用するため, プラグインを試作し, 解析処理のエラー改善を試みる.

6.4 シソーラス, オントロジーを補完

WordNet を参考にしたオントロジー構造をシステ

ムの記憶中枢とし, DBPedia, アプリケーション辞書データを統合し, より網羅性の高いオントロジー空間を構築するとともに, 5.2 で試作した形態素解析辞書プラグインへの自動的な反映も行う.

6.5 知識獲得技術との統合

知識獲得技術をシステムに統合し, 本研究成果としての知識データ統合に結合することにより, 解析時に得られたデータを機械的に取り入れ, 知識を再構築することができること, およびその効果を検証する.

7. おわりに

本研究では多段自然言語処理において辞書データの不整合が招く処理失敗率低減を目的として, 辞書データの統合を提案した. 多段自然言語処理とは音声認識, 形態素解析, 品詞推定, 固有表現・意味属性抽出, 意図推定, アプリケーション処理, 発話生成, 音声合成などで構成される. 各段では異なる自然言語リソースを利用するが, その不一致が原因で処理失敗が起こることが知られており, その問題を解決することを目的にしている.

今回の調査において, BTSJ 日本語話し言葉コーパスを処理したときの Sudachi による形態素解析結果では WordNet および DBPedia を組み合わせた場合, 60%～70%の名詞の処理が正しくできなかった. また, WordNet において約 260, DBPedia において約 1,300 の複合語が見つかるにも関わらず, 形態素解析がそれを名詞として抽出できていない. これらは処理失敗を招く可能性があり, 辞書を統合することによってその失敗率を低減できると考えられる. 今後, 網羅性を高めるために 6 章で示した方法によって試作を進め, 効果を検証する計画である.

参考文献

[1] Apache, U. I. M. A. Apache software foundation., URL http://java.apache.org/ ,(2011).

[2] Google Speech API(Speech To Text), URL https://cloud.google.com/speech-to-text/ ,(2018).

[3] Kudo, Taku.: Mecab: Yet another part-of-speech and morphological analyzer. URL http://mecab.sourceforge.jp , (2006).

[4] Apache, lucene-gosen, URL http://lucene.apache.org/, (2019).

[5] Takaoka, Kazuma, et al.: Sudachi: a Japanese Tokenizer for Business., Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)., (2018).

[6] IPAdic legacy, URL https://ja.osdn.net/projects/ipadic/, (2019).

[7] Ogiso, Toshinobu, et al.: UniDic for Early Middle Japanese: a Dictionary for Morphological Analysis of Classical Japanese., LREC. , (2012).

[8] NAIST Japanese Dictionary, URL https://ja.osdn.net/projects/naist-jdic/, (2018).

[9] Sato, Toshinori, Taiichi Hashimoto, and Manabu Okumura.: Implementation of a word segmentation dictionary called mecab-ipadic-NEologd and study on how to use it effectively for information retrieval., Proceedings of the Twenty-three Annual Meeting of the Association for Natural Language Processing. The Association for Natural Language Processing, (2017).

[10] Chinchor, Nancy, and Patricia Robinson.: MUC-7 named entity task definition, Proceedings of the 7th Conference on Message Understanding. Vol. 29, (1997).

[11] 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., 研究報告,自然言語処理 (NL) 2018.5, (2018).

[12] Auer, Sören, et al.: Dbpedia: A nucleus for a web of open data. *The semantic web*. Springer, Berlin, Heidelberg, 2007. 722-735., (2007).

[13] OWL, URL https://www.w3.org/OWL/, (2019)

[14] RDF, URL https://www.w3.org/RDF/, (2019)

[15] Isahara, Hitoshi, et al.: Development of the Japanese WordNet., (2008).

[16] Wikipedia, URL https://ja.wikipedia.org/, (2019).

[17] Magnini, Bernardo, and Gabriela Cavaglia.: Integrating Subject Field Codes into WordNet., LREC., (2000).

[18] Agirre, Eneko, and David Martinez.: Integrating selectional preferences in wordnet., arXiv preprint cs/0204027, (2002).

[19] Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. Web Semantics: Science, Services and Agents on the World Wide Web 6.3, p203-217. , (2008).

[20] 宇佐美まゆみ.: 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., 人間の相互作用研究のための多言語会話コーパスの構築とその語用論的分析方法の開発, (2011).

[21] Apache POI, URL https://poi.apache.org/, (2019).

[22] JAWJAW, URL https://code.google.com/archive/p/jawjaw/, (2019).