一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
NLC2017-17(2017-09)

# 実テキストの情報分析のための頑健な言語処理基盤

河原　大輔† 　黒橋　禎夫† 　林部　祐太† 　森田　　一† 　Arseny Tolmachev†

† 京都大学 大学院情報学研究科　〒606–8501 京都府京都市左京区吉田本町
E-mail: †{dk,kuro}@i.kyoto-u.ac.jp, hayashibe@fairydevices.jp, hmorita@jp.fujitsu.com,
arseny@nlp.ist.i.kyoto-u.ac.jp

**あらまし**　ブログ、tweet、SNS などの実テキストには、物事や商品・サービスに関する評価、意見などの人々の生の声が書かれており、人が意思決定をするとき、あるいは企業が自社の商品の評価を分析する上で貴重な情報源となる。これらの実テキストの分析を計算機で支援するためには、未知語や表記ゆれに対して頑健な形態素解析・構文解析などの基盤解析が必要となる。本論文では、実テキストの情報分析のための頑健な言語処理基盤について議論する。
**キーワード**　実テキスト，形態素解析，構文解析，語彙知識

# Robust Language Processing Infrastructure for Information Analysis of Real Texts

Daisuke KAWAHARA†, Sadao KUROHASHI†, Yuta HAYASHIBE†, Hajime MORITA†, and Arseny TOLMACHEV†

† Department of Informatics, Kyoto University　Yoshida-honmachi, Sakyo-ku, Kyoto, 606–8501 Japan
E-mail: †{dk,kuro}@i.kyoto-u.ac.jp, hayashibe@fairydevices.jp, hmorita@jp.fujitsu.com,
arseny@nlp.ist.i.kyoto-u.ac.jp

**Abstract**　On real texts, such as blog articles, tweets and SNS texts, people's real voices are written about evaluations and opinions on things, products and services. These are valuable information sources when people make decisions, and companies evaluate their products. To automatically analyze such real texts, it is essential to robustly perform fundamental analyses, such as morphological analysis and syntactic parsing, because real texts contain many unknown words and spelling variations. This paper discusses a robust language processing infrastructure for information analysis of real texts.
**Key words**　real texts, morphological analysis, syntactic parsing, lexical knowledge

## 1. はじめに

近年、インターネットが普及し、日本では人口の約 80%にあたる 1 億人が利用するようになっている(注2)。その中で多くの人が、ブログ、tweet、SNS などに、物事や商品に対する評価、意見などを書き、情報発信している。これらのテキスト (ここでは「**実テキスト**」と呼ぶ) は人々の生の声であるため、人が意思決定をするとき、あるいは企業が自社の商品の評価を分析する上で貴重な情報源となる。このような実テキストは「ビッグデータ」の一つであり、莫大な量が存在するため、その中から役に立つ情報を得るためには、計算機で実テキストを解析・分析することが必要不可欠である。

実テキストの解析・分析における問題として、新語、専門用語、固有名詞などの未知語の問題、主にひらがな書きによる単語区切りの曖昧性などがある。以下にこれらの例を示す。

(1)　市ヶ谷有咲 の ねんどろいど 欲しいですねぇ。

(2)　あの店は でもの がよく見つかる。

(1) では「市ヶ谷有咲」という人名 (キャラクター名) および「ねんどろいど」という語、(2) では「でもの」という語を正しく認識する必要がある。

我々は、実テキストの情報分析のための頑健な言語処理基盤を開発しており、本稿ではこれまでの我々の取り組みについて述べる。具体的には、形態素解析のための語彙獲得、形態素解

(注1)：第 3 著者の現在の所属はフェアリーデバイセズ株式会社，第 4 著者の現在の所属は株式会社富士通研究所.
(注2)：総務省 平成 28 年版 情報通信白書: http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/html/nc252110.html

析器 JUMAN++の高速化、JUMAN++における部分アノテーションの利用、形態素・構文統合解析について概説する。

## 2. 形態素解析器 JUMAN++

JUMAN++は、実テキストを頑健に解析するために、自動獲得した大規模な語彙辞書および Recurrent Neural Network (RNN) 言語モデル [1] を利用している形態素解析器である [2]。

### 2.1 Web 上のリソースからの語彙獲得

実テキストの解析では、新語や専門用語や固有名詞などの語彙知識が不足すると解析誤りの原因となる。Wikipedia、Wiktionary、Web コーパスからそれぞれ語彙知識を獲得し、JUMAN++で利用する辞書を構築した。

Wikipedia や Wiktionary のエントリをそのまま採用すると、長い複合名詞も 1 語として解析してしまう。このような複合名詞の登録を避けるため、1 語である可能性が高い語のみを獲得する。また、表記ゆれなどの異表記をまとめあげるために、各語に代表表記を付与する。

#### 2.1.1 Wikipedia 辞書

柴田ら [3] の手法に基づき、日本語 Wikipedia (2016/06/01 版) のエントリのうち、1 語である可能性が高いもので、かつ基本語彙辞書に含まれない語を自動的に選択し、約 83 万語の名詞からなる辞書を構築した。獲得例としては「爽健美茶」「市ヶ谷」(人名) などがある。各語の品詞細分類および代表表記は次のように決定、付与した。

**品詞細分類**: 品詞細分類として、普通名詞、人名、地名、組織名の 4 種類を候補とした。上位語の主辞の JUMAN カテゴリ、Wikipedia の記事カテゴリを用いて、品詞細分類を決定した。

**代表表記**: Wikipedia のリダイレクトの情報を用いて代表表記を付与した。Wikipedia において、エントリ A からエントリ B にリダイレクトがあり、カタカナをひらがなに正規化した後の編集距離が小さい場合に、B の「表記/読み」を代表表記として A、B に付与した。たとえば、エントリ A「スパゲティー」からエントリ B「スパゲッティ」にリダイレクトがあるとき、「スパゲティー」の代表表記に「スパゲッティ/スパゲッティ」を付与した。

#### 2.1.2 Wiktionary 辞書

日本語 Wiktionary (2016/06/01 版) のエントリにおいて、活用が明示されている動詞・形容詞で、1 語である可能性が高いもののうち、基本語彙および Wikipedia 辞書に含まれない語、約 2,000 語を自動的に選択し、Wiktionary 辞書を構築した。Wiktionary に記載されている品詞や活用型は JUMAN++辞書の品詞体系と異なるため、品詞や活用型を変換して用いた。獲得例としては「掠 (さら) う」「珍貴だ」などがある。

#### 2.1.3 Web コーパス辞書

未知語の自動獲得 [4] (品詞、活用形の推定を含む) を行い、約 1.1 万語の Web コーパス辞書を構築した。獲得例としては「ググる」「ドラえもん」「ねんどろいど」などがある(注3)。

(注3)：コーパスから未知語の自動獲得を行うプログラムは次の URL で公開されている。https://github.com/murawaki/lebyr

表 1 形態素解析器の解析時間 (秒)

| | 1 文 | 10 文 | 100 文 | 1,000 文 | 2,000 文 |
|---|---|---|---|---|---|
| MeCab | 0.013 | 0.014 | 0.017 | 0.047 | 0.080 |
| JUMAN | 0.058 | 0.060 | 0.084 | 0.220 | 0.369 |
| KyTea | 6.010 | 6.068 | 6.047 | 6.377 | 6.737 |
| JUMAN++V1 | 0.157 | 0.592 | 5.794 | 60.460 | 141.730 |
| JUMAN++V2 | 0.085 | 0.120 | 0.371 | 2.318 | 5.106 |

### 2.2 JUMAN++の形態素解析モデルとその高速化

#### 2.2.1 JUMAN++の形態素解析モデル

JUMAN++の形態素解析モデルは次の 2 つから構成される。

- Exact Soft Confidence Weighted [5] ベースの線形モデル
- RNN 言語モデル

RNN 言語モデルを用いることによって、単語の並びの意味的な自然さを考慮した解析を行うことができる。

このモデルでは、$\mathbf{y}$ を単語列、$s$ を入力文、$\mathcal{Y}(s)$ を入力文に対する全ての単語列の候補としたとき、次式を満たす単語列 $\hat{y}$ を求めることにより解析を行う。

$$\hat{y} = \underset{\mathbf{y} \in \mathcal{Y}(s)}{\operatorname{argmax}} \ \operatorname{score}(\mathbf{y}) \tag{1}$$

スコア関数 score($\mathbf{y}$) は次式で表され、第一項が線形モデル、第二項が言語モデルに対応している。

$$\operatorname{score}(\mathbf{y}) = (1-\alpha)\Phi(\mathbf{y}) \cdot \vec{w} + \alpha \ \log(p_r(\mathbf{y})) \tag{2}$$

$\alpha$ は線形補間の重み、$\Phi(\mathbf{y})$ は単語列 $\mathbf{y}$ に対する素性ベクトル、$\vec{w}$ は重みベクトル、$p_r(\mathbf{y})$ は RNN 言語モデルが単語列に与える確率を表す。$\vec{w}$ は教師有り学習により決定する。

#### 2.2.2 JUMAN++の高速化

現在公開している JUMAN++の実装は解析の計算コストが高く、解析速度が必要な用途に用いることが難しいという問題があった。より広い用途に利用できるようにするため、JUMAN++の解析速度の向上を目的に再実装を行った。この再実装に伴い、ライブラリとして他のシステムから呼び出して利用することが困難であった問題も解消している。

再実装では、次の 2 つの方針に基づいて設計・実装を行った。

- CPU によるキャッシュの利用効率を高める
- 冗長な計算を省略する

この方針に基づき、再実装版である JUMAN++V2 ではデータ構造と処理の仕方を変更している。

従来の実装では、ハッシュマップを過度に利用していた。ハッシュマップは、形態素の表層や品詞の情報など、辞書エントリの要素を表す id の格納に加え、線形モデルの素性を格納する目的にも利用されていた。辞書フォーマットを変更することにより、また、素性の格納にプレーンな配列を利用することで、辞書周りや素性の格納ではハッシュマップを利用する必要がなくなり、パフォーマンスの向上につながった。

さらに、RNN 言語モデルのスコア、状態の計算をバッチ化、ベクトル化することも、再実装でパフォーマンスが向上した大きな要因の一つである。JUMAN++V2 では形態素ラティス上の同じノードから複数のノードへ向けて、RNN 言語モデルの

表 2　部分アノテーションを利用した JUMAN++の精度 (NEWS・WEB は単語境界と品詞の F 値、FMAC-jpp・FMAC-other は単語境界の再現率で評価)

| | NEWS | | WEB | | jpp | other |
|---|---|---|---|---|---|---|
| | 境界 | 品詞 | 境界 | 品詞 | 境界 | 境界 |
| フルアノテーション (NEWS+WEB) のみ | 99.38 | 98.97 | 98.45 | 97.91 | 57.2 | 97.8 |
| +部分アノテーション | **99.41** | **98.99** | **98.46** | **97.93** | **84.0** | **99.4** |

表 3　形態素・構文解析の評価実験に用いたコーパスの記事数と文数

| コーパス | 学習 | 評価 |
|---|---|---|
| NEWS | 2,727 記事 (36,623 文) | 200 記事 (1,783 文) |
| WEB | 4,427 記事 (13,853 文) | 700 記事 (2,195 文) |

スコアと状態を計算する必要がある。この一連の計算をバッチ化、ベクトル化することにより計算効率を高めた。

MeCab(+IPADIC)、JUMAN、KyTea (Full SVM Model(注4))、JUMAN++V1 (従来の実装)、JUMAN++V2 (再実装) の解析時間を計測した。計測に用いた計算機の CPU は Intel i7-6850K、メモリ 64GB、OS は Ubuntsu 16.04 である。すべての実装は gcc 5.3.0 を用いてコンパイルした。コンパイル時には、最適化を有効にするためコンパイルオプションに`-Ofast -march=native` を指定している。1 文、10 文、100 文、1000 文、2,000 文 の 5 つの入力テキストについて、解析時間の比較を行った。JUMAN++は従来の実装、再実装ともにビーム幅を 5 として実行している。

各入力テキストについて、総実行時間を 10 回ずつ計測した中央値を表 1 に示す(注5)。2,000 文の結果を比較すると、再実装版は従来の約 27 倍の速度で動作している。

再実装版は既に従来の実装に比べ速度的に大きく改善されているが、まだ改善の余地は残されている。線形モデルは unigram から trigram 素性を用いているが、各素性はビームサーチをする際、形態素ラティス上のパスをたどるごとに計算されている。しかし、unigram、bigram 素性はラティス構築時に一度、ノードとパスに対してスコアを計算すれば十分である。

同様に、RNN 言語モデルの計算についても冗長な計算が残っている。RNN は形態素の表層と品詞に基いて計算されている。ひらがな表記の形態素など、辞書には同一の表層、品詞を持つ語が多くあり、これらの形態素に対しては RNN 言語モデルは同一のスコアを与える。これらの冗長な計算を省くことで、解析時間のさらなる短縮を見込むことができる。

上記のように再実装した JUMAN++V2 は、近日中に公開する予定である。

## 3. 形態素解析における部分アノテーションの利用

JUMAN++による形態素解析は 98〜99%と高精度であるが、ブログや SNS のような実テキストでは、新聞記事に比べて解析誤りが多い傾向がある。そのような誤りについて正しい解析を人手で作成した部分アノテーションを学習データとして用いることによって、精度をより向上させることを試みる。

部分アノテーションコーパスとして、林部が構築し公開している Fairy Morphological Annotated Corpus (以下、FMAC) がある [6](注6)。FMAC は、Wikipedia のハイパーリンクに基づく自然アノテーションについて、形態素解析器による単語区切りが異なる箇所を人手でチェックし、アノテーションしたコーパスである。たとえば、次のような部分アノテーションがなされている。

(3)　プロボクサー・医師の|川島?実|は実兄。

“|” は単語区切り、“?” は単語区切りの可能性があることを示す。“|” で囲まれたスパンの外側についてはアノテーションされていない。FMAC には約 2,000 文の部分アノテーションが含まれているが、次の約 1,400 文を用いる(注7)。

**FMAC-jpp**:　JUMAN++による単語区切りと異なる箇所 (741 文)

**FMAC-other**:　機能表現を中心としたその他の部分アノテーション (644 文)

本研究では、FMAC-jpp と FMAC-other を部分アノテーションとして利用して JUMAN++の訓練を行い、その効果を検証する。JUMAN++における部分アノテーションを利用した訓練は、次の手順で行う。

（1）　フルアノテーションの学習データを用いて JUMAN++を訓練する。

（2）　部分アノテーションの各文について、与えられた単語区切りに違反しないように、(1) のモデルで形態素解析を行う。この際、単語区切りの可能性を示す “?” の情報は利用しない。

（3）　フルアノテーションの学習データに (2) のデータをマージし、これを用いて JUMAN++を再訓練する。

フルアノテーションのデータとして、京都大学テキストコーパス（以下、NEWS）[7] と京都大学ウェブ文書リードコーパス (以下、WEB) [8] を表 3 のように学習データと評価データに分割して用いた。NEWS は新聞記事からなり、WEB はさまざまなドメインのウェブページからなる。学習には、NEWS と WEB の学習データをマージしたものを用い、評価はそれぞれの評価データについて行った。NEWS と WEB の評価データにおける単語境界と品詞の F 値と、FMAC-jpp と FMAC-other における単語境界の再現率を表 2 に示す。なお、FMAC-jpp と FMAC-other の評価は closed な評価となっている。表 2 から、

(注4)：`http://www.phontron.com/kytea/model.html`

(注5)：KyTea は実行時間のほとんどをモデルのロードに費やしており、入力文数を変えても解析時間はほとんど変わらなかった。

(注6)：`https://github.com/FairyDevicesRD/FairyMaCorpus`

(注7)：FMAC には、MeCab+Unidic による単語区切りと異なる箇所 (588 文) も含まれているが、本研究では用いない。

NEWS と WEB では精度を落とすことなく、FMAC-jpp では精度が大幅に向上していることがわかる。FMAC-jpp において改善した例を以下に示す。

(4) 同 時期 に は 細川 たかし ら が いた 。

(5) 越石 優 に 1 − 2 判定 負け 。

スペースは、部分アノテーションを利用した JUMAN++の単語区切りを表す。従来の JUMAN++では、(4) は助詞「かし ら」、(5) は副詞「優に」と誤って解析されていたが、これらが改善されている。

FMAC-jpp において改善しなかった例を以下に示す。

(6) 1 9 5 6 年 に 退職 して 、 | 三 共|に 入社 。

(7) かつて は ミッキー 形|どら 焼き | 「 ミッキー スマイル 」 を 取り扱って いた 。

“|” が部分アノテーションの単語区切りを表す。(6) では副詞「共に」、(7) では動詞「形どる」の未然形と誤って解析されている。これらの誤りを解決するには選択選好のような構造的な語彙知識を用いる必要があり、次節で述べる形態素・統合解析モデルでは正しく解析される。

## 4. 語彙知識に基づく形態素・構文統合解析

日本語のような単語区切りが明示されていない言語の解析では、形態素解析を行った後に、構文解析や格解析を行うというパイプライン処理が一般的である。しかしながら、単語区切りや品詞タグ付けの誤りが後段の解析に伝搬するという問題がある。また、前段の形態素解析においては、構文や格の情報を用いずに、単語区切りや原形を決定するのは困難な場合がある。

たとえば、例 (8) は「ある (歩) か|ない|か」と「あ (有) る|か|ない|か」という異なる単語区切りが考えられる。

(8) 逆転する可能性がまだ あるかないか を確認する

これを正しく解析するためには、「可能性がある (歩) く」という述語項構造よりも「可能性があ (有) る」という述語項構造の方が妥当であることや、「ある」と「ない」が並列構造であることなどを認識しなければならない。このような認識を行うには、語彙レベルの選択選好の知識が必要となる。

これまで、形態素解析と構文解析を統合的に行うモデルが提案されているが、パイプラインモデルと比べて精度が向上しているわけではない [9]。その原因として、従来の統合解析モデルは数万文程度の構文情報タグ付きコーパスを利用した教師有り学習モデルであり、外部の大規模な語彙知識を使っていないことが挙げられる。上記の例のような曖昧性解消は語彙レベルの選択選好が必要となるため、これらのモデルでは正しく解析できないことが多い。

本稿では、語彙知識を用いて、形態素解析・構文解析（並列構造解析を含む）・格解析を統合的に行うモデル [10] について概説する。語彙知識としては、日本語 Web テキスト 100 億文から自動構築した格フレーム辞書 [11]、日本語 Web テキスト 1 億文から word2vec(注8) [12] を実行して求めた単語ベクトルを用いる。

### 4.1 統合解析モデル

本モデルでは、以下の手順で文字単位の CKY テーブルをビームサーチで探索し、最も良いスコアを持つ構文木を最終的に出力する。この構文木は形態素・構文・格の全ての曖昧性を解消しながら生成されている。

**(1) CKY テーブルを初期化する**

形態素解析器を用いて N-best 解を出力し、それを単語ラティスに変換し、単語ラティスに含まれる単語全てを CKY テーブルに配置する。文の部分文字列をスパンとよぶ。あるスパンは CKY テーブルのあるセルと 1 対 1 に対応する。

**(2) 基本句を生成する**

事前に定義した基本句生成規則に基づき、CKY テーブルの単語から基本句(注9)を生成し、CKY テーブルに配置する。基本句生成規則は、品詞列に対するいくつかのルールを基本とする簡単なものであるが、単語分割・品詞に曖昧性があるため、基本句にも曖昧性が生じる。生成した基本句は、それ単体からなる部分構文木 (部分木) とみなせる。

**(3) ボトムアップに構文木を生成する**

隣接する 2 つの部分木を併合することで、新たな 1 つの部分木が作られる。これをボトムアップに繰り返すことで、文全体の構文木が作られる。あるスパンに対応する部分木は一般に複数ある。スコア関数 (後述) で部分木をランク付けし、上位 $b$ 件の部分木のみ保持するビームサーチを行う (この閾値 $b$ はビーム幅である)。なお、部分木のスコアを計算する際に、スコア関数を最大化するような格フレームと項の格ラベルの同定 (格解析) も統合的に行う。たとえば、図 1(a) は「可能性があ (有) るかないか」の構文木、図 1(b) は「可能性がある (歩) かないか」の構文木の生成を示す。

**(4) 最良のスコアをもつ木を選択する**

文全体の構文木が作られたら、その中から最もスコア (後述) の高い構文木を選択する。これにより、形態素・構文・格の曖昧性を全て解消した構文木が得られる。たとえば、図 1 では右上のセルには「可能性があ (有) るかないか」( 図 1(a)) と「可能性がある (歩) かないか」(図 1(b)) の 2 つの構文木が存在する。このうち「可能性があ (有) るかないか」の構文木は、格フレームの選択選好などの語彙知識を考慮すれば、最もスコアが高くなるため、これを解析結果として出力する。

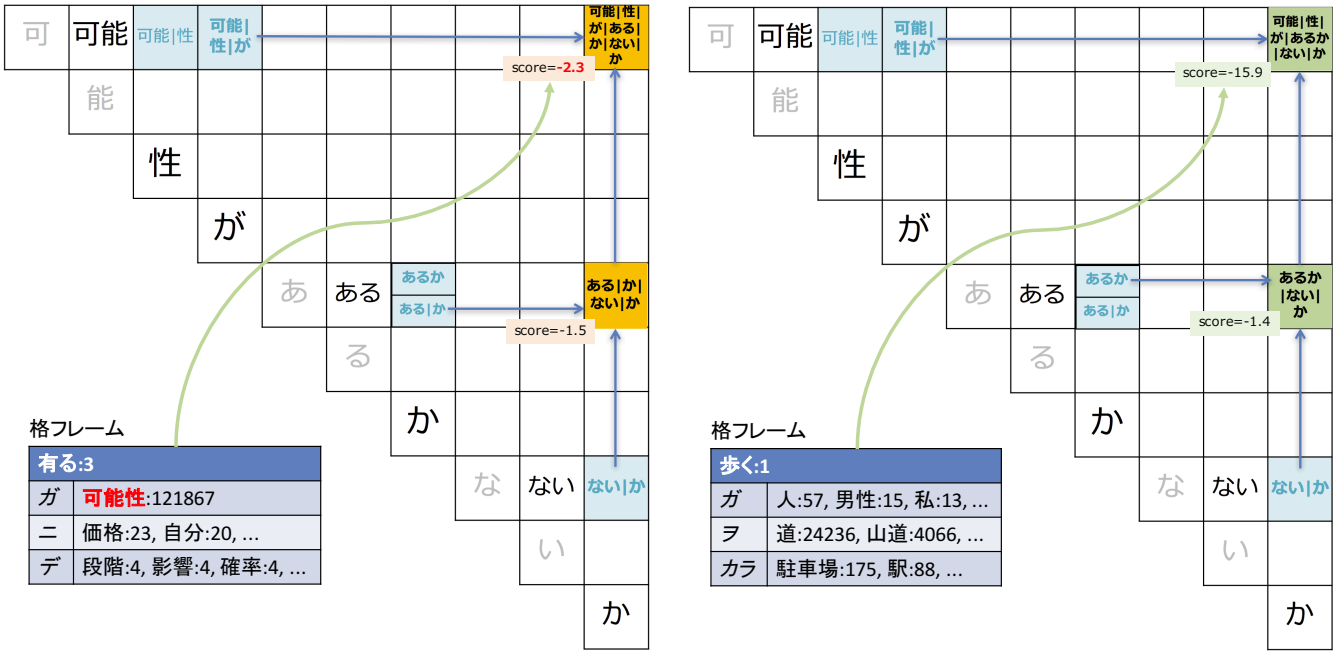
### 4.2 スコア関数の素性と学習

構文木のスコアは、素性の重み付き線形和をとする。素性は、句を構成する単語列や句の係り受け関係、述語項構造などのもっともらしさを評価するものを設定する。単語列や係り受けに関する基本的な素性は、CaboCha(注10)のものを参考に設定した。語彙知識を利用した素性としては、述語項構造のもっともらしさを捉えるために格フレームに基づく生成確率 [13] を、

(注8)：https://code.google.com/p/word2vec/

(注9)：自立語 1 個と 0 個以上の付属語をまとめたもの

(注10)：https://taku910.github.io/cabocha/




(a) 「可能性があ (有) るかないか」の構文木の生成

(b) 「可能性がある (歩) かないか」の構文木の生成

図 1　形態素・構文統合解析モデルによる解析

また並列構造のもっともらしさを捉えるために単語ベクトルに基づく単語列間の類似度を用いた。学習には候補選択学習を用い、各素性の重みは L-BFGS により求める。

学習の手順としては、まず、入力として、学習データに対する単語ラティスを用意する。そして、重みに適当な初期値を与えて、ビーム幅 $b$ で解析を行い、1 文につき最大 $b$ 個の構文木候補を得る。そして、学習データの正解の構文木と比較して、構文候補集合の中から最も構文解析精度が高いものを正例、それ以外を負例とする。全ての文について、このように学習事例を作成し、構文木候補の中から正例を選択するように訓練する。訓練によって得た重みを使い、上記の手順を繰り返すことで、最終的な重みを得る。

### 4.3　実験設定

評価実験には、表 3 に示した NEWS と WEB の 2 つのコーパスを用いた。学習には、NEWS と WEB の学習データをマージして用い、評価はそれぞれの評価データについて行った。

解析器の入力となる単語ラティスの生成には JUMAN++の N-best 出力機能を用い、N-best に含まれる単語から単語ラティスを作成した。学習データの形態素解析は 10-way jackknifing によって行い、評価データについては学習データ全体を用いて JUMAN++を学習し、生成した(注11)。評価データについては、JUMAN++の N-best 出力に加えて 1-best 出力も行い、提案モデルの入力としてこの両方を試し、比較を行った。

統合解析器のビーム幅 $b$ は 10 とした。学習器の実装には Classias(注12) を用い、L1 正則化を行った。

提案モデルの比較対象として KNP(注13) と CaboCha 0.69 を用いる。いずれのシステムも 1-best 入力しか受け付けないため、N-best 入力の実験は提案モデルのみにおいて行う。

KNP は [13] を実装したシステムである。形態素解析結果を受け取り、句生成規則に従い基本句にまとめ上げた後、格フレーム等の語彙知識を用い、確率的生成モデルに基づいて構文・格の統合的解析を行う。本実験において、KNP と提案モデルは共通の句生成規則と格フレームなどの言語資源を用いる。

CaboCha は [14] を実装したシステムである。形態素解析結果を受け取り、CRF を用いて文節にまとめ上げた後、SVM を用いて Shift-Reduce 型の構文解析を行う。CRF と SVM は提案システムと同じ学習データを用いて学習した。なお、CaboCha の CRF テンプレートは文節にまとめあげることを目的としているため、前提が KNP・提案システムと若干異なる。そこで比較条件を揃えるため、KNP の基本句まとめあげ結果を受け取り、係り受け解析のみを行う場合の実験も行った (以下 KNP+CaboCha とよぶ)。

評価は、形態素は単語境界・品詞 (単語境界+品詞大分類)・全て (単語境界+品詞大分類+品詞細分類+原形+活用型+活用形)、基本句は基本句境界、構文はラベル無し係り受けとラベル有り係り受けについて、いずれも F 値で行う。

### 4.4　実験結果

実験結果を表 4 に示す。N-best 形態素解析結果を入力とし語彙知識を用いた提案モデルが他のモデルよりも有意に高い性能を示している。有意差検定にはブートストラップ法を用いた。

以下に、提案モデルで改善された特徴的な例を 2 つ示す。

(9)　あの 店 は でもの が よく 見つかる 。

JUMAN++の 1-best 解では、下線部を判定詞「で」と名詞「もの」に誤解析していたが、提案モデルは正しく解析することが

(注11)：JUMAN++の学習には 3 節の部分アノテーションを用いていない。
(注12)：http://www.chokkan.org/software/classias/
(注13)：http://nlp.ist.i.kyoto-u.ac.jp/?KNP

表 4　形態素・構文統合解析モデルの精度 (すべて F 値)

| 入力 | | 構文解析器 | 形態素 | | | 基本句 | 構文 | |
|---|---|---|---|---|---|---|---|---|
| データ | 形態素解析器の出力 | | 境界 | 品詞 | 全て | 境界 | ラベル無 | ラベル有 |
| NEWS | 1-best | KNP | 99.38 | 98.97 | 97.50 | 98.35 | 89.68 | 87.98 |
| | | CaboCha | 99.38 | 98.97 | 97.50 | 96.17 | 89.06 | - |
| | | KNP+CaboCha | 99.38 | 98.97 | 97.50 | 98.35 | 91.00 | - |
| | | 提案モデル wo/知識 | 99.38 | 98.97 | 97.50 | 98.38 | 89.89 | 88.20 |
| | N-best | | 99.37 | 98.98 | 97.51 | 98.39 | 90.40 | 88.73 |
| | 1-best | 提案モデル | 99.38 | 98.97 | 97.50 | 98.39 | 91.26 | 89.54 |
| | N-best | | 99.38 | **99.00** | **97.54** | **98.44** | **91.61** | **89.91** |
| WEB | 1-best | KNP | 98.45 | 97.91 | 96.34 | **96.30** | 87.87 | 85.61 |
| | | CaboCha | 98.45 | 97.91 | 96.34 | 92.65 | 86.14 | - |
| | | KNP+CaboCha | 98.45 | 97.91 | 96.34 | **96.30** | 89.05 | - |
| | | 提案モデル wo/知識 | 98.45 | 97.91 | 96.34 | 96.13 | 88.36 | 86.12 |
| | N-best | | 98.48 | 97.93 | 96.39 | 96.26 | 88.79 | 86.52 |
| | 1-best | 提案モデル | 98.45 | 97.91 | 96.34 | 96.11 | 89.54 | 87.27 |
| | N-best | | **98.53** | **97.99** | **96.45** | 96.31 | **89.82** | **87.53** |

できた。提案モデルでは、格フレームを用いることによって、述語「見つかる」が「でもの」のような物をガ格としてとりやすいという知識を利用している。

(10)　おい　や　めい　と　わかれた　。

JUMAN++の 1-best 解では、下線部を接頭辞「お」と形容詞「いやだ」の語幹と誤解析していたが、提案モデルは正しく解析することができた。格フレームを用いることによって、述語「わかれる」が「おい」や「めい」などをト格としてとりやすいという知識が有効に働いている。また、「おい」と「めい」が類似していることが単語ベクトルから分かり、これらが並列構造を構成することが認識されている。

表 4 から、構文解析の大きな精度向上が見られる一方で、形態素解析や基本句境界の精度向上は有意であるものの限定的であることがわかる。日本語では、次の例のように、同じスパンにおいて同じ品詞と原形をもつ単語が候補として挙がることが多いが、これは単語区切りや品詞同定の精度には影響しない。

(11)　皮　を　むく

動詞「むく」は「向く」と「剥く」の曖昧性があり、どちらも形態素解析の N-best 解に含まれている。ここでは「剥く」が正しいが、このような曖昧性については正解コーパスにアノテーションされていないため、形態素解析の評価では区別されない。しかし、このような曖昧性は、提案モデルでは語彙知識によって正しく解消することができ、これが構文解析精度の向上に貢献していると考えられる。

## 5. おわりに

本稿では、実テキストの情報分析のための頑健な言語処理基盤について概説した。再実装版 JUMAN++、100 億文 Web コーパスから獲得した格フレーム、形態素・構文統合解析システムなど全てのリソースを近日中に公開する予定である。今後は、実テキストの情報分析に必要となる省略・談話解析の高度化に取り組んでいきたいと考えている。

## 文　献


[1] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model,” Interspeech, vol.2, p.3, 2010.

[2] H. Morita, D. Kawahara, and S. Kurohashi, “Morphological analysis for unsegmented languages using recurrent neural network language model,” Proceedings of EMNLP2015, pp.2292–2297, 2015.

[3] 柴田知秀，村脇有吾，黒橋禎夫，河原大輔，“実テキスト解析をささえる語彙知識の自動獲得，” 言語処理学会 第 18 回年次大会，pp.81–84，2012.

[4] Y. Murawaki and S. Kurohashi, “Online acquisition of Japanese unknown morphemes using morphological constraints,” Proceedings of EMNLP2008, pp.429–437, 2008.

[5] J. Wang, P. Zhao, and S.C. Hoi, “Exact soft confidence-weighted learning,” Proceedings of ICML-12, pp.121–128, 2012.

[6] 林部祐太，“日本語部分形態素アノテーションコーパスの構築，” 情報処理学会 第 231 回自然言語処理研究会，pp.9:1–8，2017.

[7] D. Kawahara, S. Kurohashi, and K. Hasida, “Construction of a Japanese relevance-tagged corpus,” Proceedings of LREC2002, pp.2008–2013, 2002.

[8] 萩行正嗣，河原大輔，黒橋禎夫，“多様な文書の書き始めに対する意味関係タグ付きコーパスの構築とその分析，” 自然言語処理，vol.21，no.2，pp.213–248，2014.

[9] 俵雄貴，東藍，松本裕治，“係り受け情報を利用した日本語形態素解析，” 情報処理学会 第 220 回自然言語処理研究会，pp.1:1–7，2015.

[10] D. Kawahara, Y. Hayashibe, H. Morita, and S. Kurohashi, “Automatically acquired lexical knowledge improves Japanese joint morphological and dependency analysis,” Proceedings of IWPT2017, 2017 (to appear).

[11] 林部祐太，河原大輔，黒橋禎夫，“格パターンの多様性に頑健な日本語格フレーム構築，” 情報処理学会第 224 回自然言語処理研究会，pp.14:1–8，2015.

[12] T. Mikolov, C. Kai, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” Proceedings of Workshop at International Conference on Learning Representations, 2013.

[13] 河原大輔，黒橋禎夫，“自動構築した大規模格フレームに基づく構文・格解析の統合的確率モデル，” 自然言語処理，vol.14，no.4，pp.67–81，2007.

[14] 颯々野学，“日本語係り受け解析の線形時間アルゴリズム，” 自然言語処理，vol.14，no.1，pp.3–18，2007.