Web 上のひらがな交じり文に頑健な形態素解析

工藤拓, 市川宙, David Talbot, 賀沢秀人
Google Inc.
{taku,ichikawa,talbot,kazawa}@google.com

1 はじめに

現在利用されている形態素解析器の多くは，新聞記事といった「きれいな」コーパスをベースに構築されていおり， Web 上のくだけた表現を頑健に解析することは難しい．Web テキストの解析を困難にする要因として，未知語，顔文字，機能語の特有の言い回し，スペルミスなどがある．その中でも頻出する現象に正規の表現がひらがなのまま表記される現象 (ひらがな化現象) がある．例えば,「きょうとに行った」という文において，ひらがな表現「きょうと」は，通常形態素解析器の辞書に登録されていないため，解析に失敗する．

本稿では，Web 上に頻出するひらがな交じり文に対して頑健な形態素解析を提案する．語彙もしくは学習データの整備といった通常の精度向上手法とは異なり，提案法は，既存の辞書と大量の生コーパスを用いた教師なし学習をベースとしている．具体的には，ひらがな交じり文が生成される過程を生成モデルでモデル化し，そのパラメータを大規模 Web コーパス及び EM アルゴリズムで推定することで，ひらがな交じり文の解析精度の向上を試みる．

2 ひらがな交じり文の背景と影響

ひらがな交じり文が生み出される背景として以下が考えられる．

- **日本語入力ソフトの影響**
  電子化されたテキストは通常日本語入力ソフト (以下 IME) を用いて作られる． ユーザはひらがな列をタイプしたあと，変換キー (通常はスペースキー) を押すことで，対象のひらがな列を変換する．しかし，この変換が常に行われるとは限らない．IME の精度が悪い場合は，ユーザは変換することをあきらめ，ひらがなのまま残すであろう．また，チャットやツイッターといった即時性が要求される場面では正確性よりはタイピング速度が重要となり，ひらがなをそのまま残す傾向が強くなる．

- **固有名詞**
  近年，地名といった固有名詞をあえてひらがな化することで，既存の固有名詞の意味は残しつつ，オリジナルとの差別化をねらう試みが増えてきている．「さいたま市」などがそれに該当する．病院や店舗などにその地域の地名をひらがな化した名称をつけることも散見される．

これらの要因を考慮すると， 今後 SNS 上でカジュアルにテキスト入力する機会が増えれば増えるほど，ひらがな交じり文の流通量が増加し、同文の正確な解析の需要が増えるものと予想される．

さらに，ひらがな交じり文の解析誤りは，機械翻訳といった実応用に深刻な影響を及ぼす場合がある．“Yu went to the bottle songs.” これはひらがな化された文「ゆうびんきょくに行った」を Google 翻訳[1]を用いて英語に翻訳した結果である． ひらがな表現「ゆうびんきょく」は形態素解析器の辞書になく,「ゆう」「びん」「きょく」に分割されてしまったため， それに対応する”Yu”，”bottle”，”songs” が訳語として選ばれている． ひらがな交じり文が誤って解析されると，思いもかけないような訳語が選択されることがあり， ユーザに対する印象を著しく低下させる[2]．

3 既存手法の問題点

日本語形態素解析は充分成熟した技術であり，既存の形態素解析器の精度を向上させたり，ドメイン適応のための方法論も確立されつつある [1, 5, 4, 6]．ひらがな交じりの解析も， 通常の日本語の文の解析であることには変わりがないため，以下のような一般的に用いられている既存手法で解析精度を向上させることが可能である．

1. ひらがな単語のユーザ辞書への追加
2. ひらがな交じり文を含む学習データを人手で作成し，再学習

1. は簡単な手法であるが，ひらがなは日本語の機能語に用いられているため，むやみにひらがな語を追加すると副作用により精度が低下する可能性がある．2. の方法は学習データの作成が必要なためコストが高い．また，任意の単語がひらがな化されることを考えると語彙の情報が不可欠であり，十分なタグ付きデータを用意する必要がある．

4 生成モデルによる定式化

4.1 基本的なアイデア

ひらがな交じり文「きょうとに行った」とそれに対応する正規の文「京都に行った」を考えてみる．このふたつの文は表層文字列が異なるため，形式的には異

[1] http://translate.google.com/
[2] ユーザはたとえひらがな交じりでも一般的な単語であれば正しく翻訳されるという期待がある．

なる文である．しかし，これらを異なる文として扱うよりはむしろ，どちらも共通した文「京都に行った」から表記の揺れという形で生成された文と考えるほうが自然であろう．

提案法では，観察可能なひらがな交じり文 $\boldsymbol{w}$ は，隠れた共通の言語モデル $P(\boldsymbol{v})$ から表記がひらがな化する確率 $P(\boldsymbol{w}|\boldsymbol{v})$ を経て生成されたと考える． 言語モデル $P(\boldsymbol{v})$ は実際には観察不可能であるため，$\boldsymbol{v}$ で周辺化することで $P(\boldsymbol{w})$ は式 1 のように定式化できる．

$$P(\boldsymbol{w}) = \sum_{\boldsymbol{v}} P(\boldsymbol{w}|\boldsymbol{v})P(\boldsymbol{v}). \tag{1}$$

式 1 の確率 $P(\boldsymbol{w}|\boldsymbol{v})$， $P(\boldsymbol{v})$ は大量のひらがな交じり生文および EM アルゴリズムを用いて推定する．

提案法のねらいは，これらの確率値の自動推定にある． どの単語も等確率でひらがな化されると考えるよりは単語やコーパスに応じて $P(\boldsymbol{w}|\boldsymbol{v})$ が変化すると考えるほうが妥当であろう．さらに，推定後の $P(\boldsymbol{w}|\boldsymbol{v})$ を観察することで，どのような単語・表現が Web 上でひらがな化されやすいかを知ることができる．これらの情報は，ひらがな語彙の追加といった辞書編さんのためのヒントとして活用可能である．また，式 2 のように，与えられた確率モデルから，$P(\boldsymbol{w})$ を最大化する $\boldsymbol{v}_{opt}$ を探索することで，ひらがな交じり文に対応する正規文を導出することが可能である．

$$\boldsymbol{v}_{opt} = \underset{\boldsymbol{v}}{argmax}\, P(\boldsymbol{w}|\boldsymbol{v})P(\boldsymbol{v}) \tag{2}$$

4.2 クラス言語モデルを用いた定式化

本章では $P(\boldsymbol{w}|\boldsymbol{v})$ および $P(\boldsymbol{v})$ の具体的な定式化について述べる． 提案法は一般性の高い手法であり，任意の確率モデルを適用可能である．しかし，後述する Web データを使った EM アルゴリズム，高速なデコーディング，および形態素解析器としての定式化のしやすさなどを考慮し，単純な確率モデルを採用した．

まず，多くの言語モデルの手法と同様に，文 $\boldsymbol{w}$, $\boldsymbol{v}$ は単語列として表現する．

$$\begin{aligned} \boldsymbol{w} &= (w_1, \ldots, w_n) \\ \boldsymbol{v} &= (v_1, \ldots, v_n) \end{aligned}$$

表記のひらがな化確率 $P(\boldsymbol{w}|\boldsymbol{v})$ については， 単語ごとに独立にひらがな化されると仮定する．

$$P(\boldsymbol{w}|\boldsymbol{v}) = \prod_{i=1}^{n} P(w_i|v_i) \tag{3}$$

ここで， $P(w|v)$ は，単語 v が，w として表記される確率である． 本稿では，単語 v がそのままの形で表記される場合と，ひらがな化されて表記される場合を考える．ひらがな表記に遷移する確率値が高い単語 v は，対象コーパス (Web コーパス) ではひらがな化されやすいことを意味しており，形態素解析器への追加単語候補の列挙に利用できる．

隠れ言語モデル $P(\boldsymbol{v})$ については， 品詞をクラスとするクラス言語モデルを用いる．このモデルは， 永田 [5] や浅原ら [1] による形態素解析手法と同一のものである．クラス言語モデルを用いることで，本手法は形態素解析器として利用するとが可能となる．

$$P(\boldsymbol{v}) = \prod_{i=1}^{n} P(v_i|c_i)P(c_i|c_{i-1}). \tag{4}$$

ただし，c は，単語 v に対する品詞である．

式 3,4 を組み合わせることで，観測ひらがな交じり列 $\boldsymbol{w}$ に対する言語モデル確率 $P(\boldsymbol{w})$ は以下のようになる．

$$P(\boldsymbol{w}) = \sum_{v,\, c} \prod_{i=1}^{n} P(w_i|v_i)P(v_i|c_i)P(c_i|c_{i-1}). \tag{5}$$

形態素解析器として用いる場合は， 式 6 のように確率値を最大化する単語，品詞列 $\langle \boldsymbol{v}_{opt}, \boldsymbol{c}_{opt} \rangle$ を求めればよい．

$$\langle \boldsymbol{v}_{opt}, \boldsymbol{c}_{opt} \rangle = \underset{\boldsymbol{v},\boldsymbol{c}}{argmax} \prod_{i=1}^{n} P(w_i|v_i)P(v_i|c_i)P(c_i|c_{i-1}) \tag{6}$$

4.3 パラメータ推定

前章で述べたように，確率値 $P(\boldsymbol{w}|\boldsymbol{v})$， $P(\boldsymbol{v})$ は，大量のひらがな交じりの生コーパス (Web コーパス) および EM アルゴリズムを用いて推定する．式 5 は，隠れマルコフモデル (HMM) とほぼ同一の構造をしており，HMM と同様 Baum-Welch アルゴリズムを用いてパラメータ推定が行える．

EM アルゴリズムは一般に初期値の設定に敏感である．また，隠れ言語モデル $P(\boldsymbol{v})$ は，形態素解析器としての出力の一貫性を保つために，生文から直接推定されたクラス言語モデルと大きく変わらないほうが望ましい．そこで，通常の EM アルゴリズムに以下の二つの制約を導入した．

1. $P(v_i|c_i)$， $P(c_i|c_{i-1})$ の初期値として，生コーパスから直接推定されたパラメータを用いる．直接推定とは，MeCab[3]等の形態素解析器を用いて生コーパスを解析し，その結果からパラメータを最尤推定することを意味する．

2. M ステップにおいて，生コーパスから直接推定された頻度と E ステップで得られた頻度を MAP アダプテーションの一種である count merging という手法を用いて混合する [2].

1. は一般的に用いられている手法である．2 に関しては，隠れ言語モデルが，直接推定された言語モデルから大きく離れないようにするための制約である．具

[3] http://mecab.sourceforge.net

体的には，M ステップで最大化される単語生起確率値を以下のように頻度の線型結合で導出する．

$$P(v|c) = \frac{\alpha \cdot f_E(v,c) + (1-\alpha) \cdot f_D(v,c)}{\alpha \sum_v f_E(v,c) + (1-\alpha) \cdot \sum_v f_D(v,c)}$$

ただし，$f_E(v,c)$ は，E ステップで得られた期待頻度，$f_D(v,c)$ は，直接推定された頻度である．遷移確率も同様の手法で更新する．α は実験的に設定するが，一般には 0.01 といった小さな値にする．表記のひらがな化確率 $P(\boldsymbol{w}|\boldsymbol{v})$ については，適当な初期値を用いるのみで，2. のような制約は導入しない．上記の設定により，局所解が複数ある場合，$P(\boldsymbol{v})$ をなるべく動かさずに，$P(\boldsymbol{w}|\boldsymbol{v})$ を大きく更新するような作用が働く．

5 実験および考察

提案法の効果を検証するために，形態素解析単体の評価および実応用 (機械翻訳) での評価を行った．いずれの実験においても，式 6 を用い，単語分割，品詞推定，表記の正規化を行った．

5.1 形態素解析単体の評価

5.1.1 実験設定

提案法の比較対象として以下のふたつのベースラインを用意した．

- **ひらがな化なし**: ひらがなに変化する確率を常に 0 としたモデル．これは，MeCab を直接使うモデルとほぼ等しい[4].
- **一様分布**: $P(\boldsymbol{w}|\boldsymbol{v})$ を一様分布としたモデル．ひらがな化が通常の表記と同程度の頻度で起こると仮定した楽観的なモデル．ユーザ辞書にひらがな語を追加する設定に近い．

隠れ言語モデル $P(\boldsymbol{v})$ の初期値は，MeCab 0.98 及び ipadic-2.7.0 を使い擬似的な解析済みデータを構築し，そこから最尤法を用いて推定した．ひらがな化確率 $P(\boldsymbol{w}|\boldsymbol{v})$ の初期値は，表記がひらがな化される場合は 0.01 それ以外は，0.99 とした． 上記ふたつのベースラインでの隠れ言語モデル $P(\boldsymbol{v})$ は，EM アルゴリズムを用いず，初期値パラメータをそのまま使用した．学習用のコーパスには Web からクロール・サンプリングした約 200 億文を，EM アルゴリズムの実行には並列分散処理システム MapReduce[3] を用いた．

評価方法としては，正解データに対する適合率・再現率を計算することが望ましいが，作成コストがかかるため，提案法との変更点のみを比較・検討した．具体的には，Web からランダムサンプリングした 5000 文をふたつの形態素解析で解析し，その差分を集め[5]，各差分の要素を以下の基準でスコア付けし，その平均点で比較した．0 点: 不正解，1 点: 単語分割のみ正解，2 点: 単語分割+品詞付与のみが正解[6]， 3 点: 単語分割+品詞付与+単語の正規化 (ひらがな語をもとに戻す) のが正解[7].

表 2: 形態素解析の結果

比較対象	変更数	スコア
提案法 vs ひらがな化なし	334	2.25 vs 0.55
提案法 vs 一様分布	645	2.58 vs 2.38

5.1.2 結果

表 2 に, 各ベースラインと提案法との比較を示す．結果，提案法はどのベースラインよりも高い精度を示していることが分かる．ベースライン毎に評価データが変わる相対評価であるため，ベースライン間の直接比較はできないことに注意されたい．

ひらがな化なしは，平均スコアが 1 を下回っており，単語分割で多くの誤りを生じている．以下に，提案法では正解し，ひらがな化なしで解析誤りとなった例を示す．

- 誤: かん (動詞) た (助動詞) ん (名詞) に 弾く
  正: かんたん (名詞/簡単) に 弾く
- 誤: 活躍 し て いる 「 はや (副詞) ぶさ (動詞)
  正: 活躍 し て いる 「 はやぶさ (名詞/隼)
- 誤: .. くん の え (フィラー) ほん (名詞)
  正: .. くん の えほん (名詞/絵本)
- 誤: 空港 近く の りん (副詞) くう (動詞) 公園
  正: 空港 近く の りんくう/(名詞/臨空) 公園

一様分布は，提案法との変更点が多い (645 箇所) にもかかわらず，提案法と同程度の高いスコアを示している．これは，一様分布が，非自立の用言など，ひらがな・漢字両表記とも許容できる場合，漢字に変更する傾向が高かったためである．提案手法は，すでにひらがな表記が辞書に登録されており，その表記が一般的である場合は，モデルの性質上漢字表記にすることは少ない．一様分布では，ひらがなの辞書項目を無視してまで許容可能な漢字表記に正規化する傾向が強かった．

また，一様分布は，ひらがな語の副作用，つまり機能語を誤って内容語として解釈する現象が散見された．さらに，どの単語も等確率でひらがな化されるため，一般にはひらがな表記が主流の表記を無理に漢字として解釈することによる単語分割誤りもあった．以下に提案法では正解し，一様分布で解析誤りとなった例を示す．

- 誤: 輸出 が さ れ て はいる/(動詞/入る) が
  正: 輸出 が さ れ て は (助詞) いる (動詞) が

[4] MeCab の大量の解析結果から同時確率モデルを構築するため，MeCab の解析結果を再現するようなモデルが構築されることが期待できる．

[5] diff -u コマンドを用いた．

[6] 品詞大分類のみを考慮した．

[7] ひらがな表記も一般的な場合 (例えば，非自立の動詞など) は，ひらがな語を元に戻さなくても正解とした．

表 1: 機械翻訳例

入力文	MeCab (ひらがな化なし)	提案法 (ひらがな化あり)
やすゆき (2)	Ease of Snow (2)	Yasuyuki (2)
たかおみゆき	Miyuki Takao us	Miyuki Takao
こうべ de カード EC が ご利用いただけます。	EC de card should be available this way.	Kobe EC de card available.
2 ちゃんねる Viewer を使うと、 すぐに読めます。	Using the Viewer 2 channel chan, I can read now.	Using the Viewer channel 2, you can read quickly.
おちゃのこ通信バックナンバー	Back number of Tea can communicate your	Back Issues Ochanoko communication

- 誤: ええ 加減 にし/(名詞/西) ぃ や
  正: ええ 加減 に (助詞) し (動詞) ぃ や
- 誤: 今朝 起き たら 、 こんなん/(名詞/困難) なっ て まし た
  正: 今朝 起き たら 、 こんな (連体詞) ん (名詞) なっ て まし た
- 誤: ひょうが/(名詞/氷河) ら (名詞) き てる
  正: ひょう (名詞) がら (動詞) き てる

5.1.3 ひらがな化確率

提案法のもう一つの利点は，ひらがな化確率 $P(w|v)$ を確率の高い順に並べることで，Web 上でどのような単語がひらがな化されやすいか分析ができることにある．表 3 に，ひらがな化単語の例を確率とともに示す．「おせち/お節」「もののけ/物の気」といったひらがなが一般的な表記に高い確率が付与されている．また，「これほど/これ程」といったようにどちらの表記も同程度使われる場合は，確率値もその事実を反映し，極端に高い (低い) 値にはなっていない．逆に，確率値の小さい単語はアルファベットや数字であった．これらがひらがな化されることは稀であり，直感的にも妥当な結果となっている．

表 3: ひらがな化単語の例

$P(w\|v)$	w	v
0.999	おせち	お節
0.999	おやつ	お八つ
0.999	あらすじ	荒筋
0.999	もののけ	物の気
0.952	しんしんと	津津と
0.893	これほど	これ程
0.864	なう	ナウ
0.845	そろばん	ソロバン
0.741	おのぼりさん	お上りさん
0.000	いー	e
0.000	れい	0

5.2 機械翻訳における効果

フレーズベースの日英統計翻訳システムの前処理として，MeCab を使ったシステムと提案法の形態素解析器を使ったシステムのふたつを作成し，提案法の効果を調べた．比較対象のコーパスには，Web からランダムサンプリングした 9536 文から，異なる英語訳文が生成された入力日本語文約 3409 文 (35%) を抽出した．その中から，200 文をランダムサンプリングし，訳文を人手により 6 点満点で評価し，その平均点で比較を行った．

結果，平均点は 1.75 から 1.91 に向上し，本手法の実応用での有効性が検証できた．表 1 に，翻訳精度が向上した例を示す．

6 おわりに

本稿では，Web 上のひらがな交じり文を頑健に解析する形態素解析器を提案し，その効果を機械翻訳で検証した．未知語は形態素解析における主要な問題である．これまで未知語とひとくくりに議論されることが多かったが，未知語には固有名詞といった真の未知語と，既知語の屈折現象のふたつに分けて考えるほうが，その生成過程の違いから考察しても自然だろう．前者の詳細は語彙獲得研究に譲るとして，後者は，既存語の知識を再利用できる点で，真の未知語よりも取り組みやすい．また，既知語への正規化が行える実用上の利点がある．提案手法は，ひらがな化という特殊な屈折を扱ったが，Web 上には，小書き (食べよぅ) や長音化 (食べたーい) といった現象も多く，このような表現の解析にも，提案手法と同様の方法が適用可能であろう．

参考文献

[1] Masayuki Asahara and Yuji Matsumoto. Extended models and tools for high-performance part-of-speech tagger. In *Proc of COLING*, pp. 21–27, 2000.

[2] Michiel Bacchiani and Brian Roark. Unsupervised language model adaptation. In *Proc. of ICASSP*, p. 224227, 2003.

[3] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proc. of OSDI*, 2004.

[4] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP*, 2004.

[5] 永田昌明. 統計的言語モデルと n-best 探索を用いた日本語形態素解析法. 情報処理学会論文誌, Vol. 40, No. 9, 1999.

[6] 中田陽介, NEUBIG Graham, 森信介, 河原達也. 点予測による形態素解析. 情報処理学会研究報告 NL198, 2010.