最小記述長原理に基づいた日本語話し言葉の単語分割

松原　勇介 †　　秋葉　友良 ‡　　辻井　潤一 †§*

† 東京大学大学院情報理工学系研究科　　‡ 豊橋技術科学大学情報工学系

§School of Computer Science, University of Manchester　　* National Centre for Text Mining, UK

matubara@is.s.u-tokyo.ac.jp , akiba@ics.tut.ac.jp , tsujii@is.s.u-tokyo.ac.jp

1　はじめに

本稿では、少ない作業量で新規ドメインに特化した N-gram 言語モデルを与えることができる単語分割法を提案し、その手法を日本語話し言葉ドメインに適用した実験結果を報告する。

実用化された言語処理システムの多くが、N-gram 言語モデルを利用している。たとえば、大語彙連続音声認識システムの多くが音響モデルと N-gram 言語モデルの組み合わせで実現されている。

N-gram 言語モデルは、単語の出現が長さ $N-1$ の文脈単語列にのみ依存するとの仮定に基づき、単語列に尤度を与える。高精度な尤度を得るためには、単語区切りが付与された大規模なコーパス（例文）を用いた訓練が必要である。しかし、単語区切りは日本語の正書法では表記されないので、日本語の生コーパスを使って N-gram 言語モデルを訓練することはできない。そこで自動的に単語区切りを与える手法が必要とされている。以降、N-gram 言語モデルの単位としての単語区切りを与えることを、単語分割と呼ぶ。

N-gram 言語モデルを高精度にするためには、対象ドメインに適合した訓練コーパスを用いることが重要である。あるドメインで高い性能を上げた単語分割法であっても、新しいドメインへ適用すると、ドメイン特有の単語（未登録語）や連語、語法に対して良い分割を与えられないことがある。この問題は、人手による辞書の補充 [1] や対象ドメインの単語分割済みコーパスの利用 [2] により解決されてきた。しかしこれらの手法は人手の作業を多く必要とするため、教師なし学習による単語分割 [3] [4] [5] の高精度化が期待されている。

ドメイン特有の連語や語法の一例として、対話ドメインにおける挨拶が挙げられる。MeCab[6] による形態素解析では「よろしくお願いします」という文字列は「よろしく／お願い／し／ます」という単語列に分割される。しかし対話ドメインを対象とする場合、この単語列は高い頻度で連続して現れる。このような頻出する単語列を 1 単語とみなすことにより、N-gram 言語モデルの N を変えないまま、言語モデル性能を向上させることができる [1]。ドメイン特有の連語や語法には、挨拶のように数の限られる定型句だけではなく、組織名などあらかじめ限定しづらいものも含まれる。

ドメインに特化した N-gram 言語モデルを少ない作業量で構築することを目指して、本稿では、コーパス 2-gram 符号化による記述長を目的関数とする最小記述長原理に基づいた単語分割法を提案する。話し言葉ドメインの生コーパスを提案手法で分割する実験を行い、その分割を形態素解析による分割と比較する。さらに、形態素分割済みコーパスを元に、形態素よりも N-gram 言語モデルにとって有効な区切りを与える方法を論じる。

2　関連研究

形態素解析 [6] による分割で N-gram 言語モデルを構築することが広く行われている。母語話者が与えた分割を教師データとして、教師あり学習を行う手法 [7] も提案されている。新しいドメインに特化させるためには、形態素解析の場合、ドメイン用形態素辞書の作成が必要である。教師あり学習の場合は、ドメインにあった分割基準を選定し、教師データを作成しなおす必要がある。

教師なし学習に基づく手法の多く [8] [4] は、単語 1-gram モデルなどの生成モデルを仮定して、尤度最大化を基準として分割を推定する。形態素解析や教師あり学習と比べて作業コストが小さく、ドメイン適応が容易である。 しかし、それらの多くには、学習の最適化基準とテスト時の評価指標が一致していないという問題がある。

3　提案手法

本稿の提案手法は Argamon らによる形態素 2 分割を繰り返して形態素辞書を得る手法 [9] を基にしている。Argamon らは、仮の形態素を *morph* と呼び、morph 辞書の文字単位の 0-gram 符号化とコーパスの morph 単位の 1-gram 符号化の符号長の和を最小にするように morph 辞書を修正する手法を提案した。

Argamon らは初期の morph をあり得る最大の単位（morph=単語）としたが、我々が対象とする日本語は単語区切りが与えられない言語であり、あり得る最大の単位は文か文節となる。文や文節はほとんど再現性のない部分文字列であり、それらを単位とする 1-gram 確率の推定は困難である。

そこで我々は、文字単位に分割されたコーパスが与えられたときに、それを仮の単語とみなして単語の連結を繰り返すことにより、よりよい単語分割を得る手法を提案する。学習の基準をテストにおける評価指標として用いられる N-gram パープレキシティに近づけるため、コーパスの符号長は 2-gram 符号化で評価する。

3.1 概観

提案手法は、初期の単語分割（文字単位や形態素単位など）に対して、単語連結の適用を繰り返すことにより新しい分割を与える。

単語連結の手順は、次のように学習する。

1. すべての 2 単語連接について、その 2 単語を 1 単語とみなした場合のコーパスの記述長の減少量を求める。
2. もし、記述長を減少させる 2 単語連接がなければ、その時点の分割を最終結果とする。そうでなければ、減少量がもっとも大きい 2 単語連接を選び、それを 1 単語とするように、コーパスと辞書を書き換える。連結された 2 単語を連結対象リストの末尾に追加する。
3. 以上を繰り返す。

テストコーパスに分割を与えるときには、得られた連結対象リストの先頭から順に、連結を適用する。

このアルゴリズムを素朴に実装すると、訓練コーパスの延べ単語数 N, 語彙数 V に対し、1 ステップあたり $O(NV^2)$ の計算時間がかかる。Argamon らは形態素 2 分割に際する記述長の差分を式で導出することにより、分割候補ごとの $O(N)$ の計算時間を $O(1)$ にした。我々も、以降で連結に際する記述長の差分を導出し、計算時間を削減する。

3.2 定義

ある単語列 $\{w_i\}$ 中での単語列 $v_1v_2\cdots v_m$ の出現頻度を

$$\#(v_1v_2\cdots v_m;\{w_i\}) \tag{1}$$

と書く。たとえば、$\#(ABA;ABABA)=2$ である。文脈から明らかである場合は ; 以降を省略する。

記号列 $S\in\Sigma^*$ が最尤推定された $N-1$ 重マルコフ情報源によって生成されたときのエントロピー符号化を、S の Σ 単位の N-gram 符号化と呼ぶ。

3.3 辞書の 1-gram 符号化とコーパスの 2-gram 符号化

辞書の符号長 $len(Dict.)$ は、文字単位の 1-gram 符号化のときの符号長とする。すなわち、

$$len(Dict.) = -\sum_{w\in Dict.}\sum_{c\in w}\log\frac{\#(c;Dict.)}{L_{Dict.}} \tag{2}$$

である。ここで、$\#(c;Dict.)$ は文字 c の辞書中での出現頻度、$L_{Dict.}$ は文字数で数えた辞書の長さである。

コーパスの符号長 $len(Corp.)$ は、単語単位の 2-gram 符号化のときの符号長とする。すなわち、

$$len(Corp.) = -\sum_{v_1\in Dict.\cup\{\$\},\ v_2\in Dict.}\#(v_1v_2)\log\frac{\#(v_1v_2)}{\#(v_1)} \tag{3}$$

ここで、$ はコーパスの先頭位置を表す特殊文字とする。

3.4 辞書の 1-gram 符号長の差分

2 単語 A,B の連結に際して、辞書に追加される文字の列[*1]を S とする。辞書の符号長は、連結に際して式 (4) だけ増加することが式 (2) から導かれる。

$$\begin{aligned}&\sum_{c\in S}\#(c;Dict.)\left[\cdot\left(\log\frac{\#(c)}{L_{Dict.}}-\log\frac{\#(c)+\#(c;S)}{L_{Dict.}+|S|}\right)\right.\\&\left.+\#(c;S)\left(-\log\frac{\#(c)+\#(c;S)}{L_{Dict.}+|S|}\right)\right]\\&+\sum_{c\in\Sigma-S}|S|\left[-\log L_{Dict.}-\log(L_{Dict.}+|S|)\right]\end{aligned} \tag{4}$$

[*2]

3.5 コーパスの 2-gram 符号長の差分

AB の連結に際して、コーパスの符号長は式 (5) だけ増加することが式 (3) から導かれる。

$$\#(ABAB)\cdot\left[\log\frac{\#(BA)}{\#(B)}+\log\frac{\#(AB)}{\#(A)}-\log\frac{\#(ABAB)}{\#(AB)}\right]$$

{AB を文脈とする AB の全ての出現[*3]}

$$+(\#(BAB)-\#(ABAB))$$

$$\cdot\left[\log\frac{\#(BA)}{\#(B)}+\log\frac{\#(AB)}{\#(A)}-\log\frac{\#(BAB)-\#(ABAB)}{\#(B)-\#(AB)}\right]$$

{xB(ただし $x\neq A$) を文脈とする AB の全ての出現 }

[*1] たとえば、A,B が一度も連続して現れないなら A,B を構成する文字すべての列。A,B が常に連続して現れるなら空列。

[*2] Σ はアルファベット全体、$|S|$ は S の要素数を表す。

$$+\#(AAB)$$
$$\cdot\left[\log\frac{\#(AA)}{\#(A)}+\log\frac{\#(AB)}{\#(A)}-\log\#(AAB)(\#(A)-\#(AB))\right]$$
{A を文脈とする AB の全ての出現 }
$$+\#(xAB)\cdot\left[\log\#(xA)+\log\frac{\#(AB)}{\#(A)}-\log\#(xAB)\right]$$
{$x\neq A,B$ を文脈とする AB の全ての出現 }
$$+(\#(AA)-\#(AAB))\cdot\left[\log\frac{\#(AA)}{\#(A)}-\log\frac{(\#(AA)-\#(AAB))}{\#(A)-\#(AB)}\right]$$
{A を文脈、B 以外を後続とする A の全ての出現 }
$$+(\#(ABA)-\#(ABAB))$$
$$\cdot\left[\log\frac{\#(BA)}{\#(B)}-\log\frac{(\#(ABA)-\#(ABAB))}{\#(AB)}\right]$$
{AB を文脈、B 以外を後続とする A の全ての出現 }
$$+(\#(A)-\#(ABA)-\#(BAB)+\#(ABAB))$$
$$\cdot\left[\log\frac{\#(BA)}{\#(B)}-\log\frac{\#(BA)-\#(ABA)-\#(BAB)+\#(ABAB)}{\#(B)-\#(AB)}\right]$$
{$xA(x\neq A)$ を文脈、B 以外を後続とする A の全ての出現 }
$$+(\#(xA)-\#(xAB))\cdot[\log\#(xA)-\log(\#(xA)-\#(xAB))]$$
{$x\neq A,B$ を文脈、B 以外を後続とする A の全ての出現 }
$$+(\#(BB)-\#(ABB))\cdot\left[\log\frac{\#(BB)}{\#(B)}-\log\frac{(\#(BB)-\#(ABB))}{(\#(B)-\#(AB))}\right]$$
{ A 以外を文脈、B を後続とする B の全ての出現 }
$$+\#(ABB)\cdot\left[\log\frac{\#(BB)}{\#(B)}-\log\frac{\#(ABB)}{\#(AB)}\right]$$
{A を文脈、B を後続とする B の全ての出現 }
$$+\sum_{x\in Dict.-A-B}\Bigg\{\#(ABx)\cdot\left[\log\frac{\#(Bx)}{\#(B)}-\log\frac{\#(ABx)}{\#(AB)}\right]$$
{AB を文脈とする x の全ての出現 }
$$+(\#(Bx)-\#(ABx))\cdot\left[\log\frac{\#(Bx)}{\#(B)}-\log\frac{(\#(Bx)-\#(ABx))}{(\#(B)-\#(AB))}\right]$$
{$yB(y\neq A)$ を文脈とする x の全ての出現 }
$$+\#(Ax)\cdot[-\log\#(A)+\log(\#(A)-\#(AB))]\Bigg\}$$
{A を文脈とする x の全ての出現 }
(5)

3.6 連結手順の学習アルゴリズム

以上の式を用いて、連結手順を学習するアルゴリズムを次に示す。

$J := \{\}$
$\# := Corp.$中 の N-gram の頻度 ($N = 1,\cdots,4$)
`while true`
  $\Delta L_{min} = \min_{(A,B)\in V\times V}\Delta L_{AB;\#}$
  $(A,B) = \mathrm{argmin}_{(A,B)\in V\times V}\Delta L_{AB;\#}$
  `if` $\Delta L_{min} < 0$ `exit while`
  $Corp.$ 中の A,B 連接をあらたな単語"AB" に置き換える
  # を更新
`output J and exit`

ただし、$\Delta L_{AB;\#}$ = 式 (4)+ 式 (5)

4 実験

4.1 実験設定

提案手法による分割と MeCab[6] による分割を、2-gram 言語モデルのの文字当たりパープレキシティで比較した。比較した手法は次の三つである。

MeCab　訓練コーパスとテストコーパスそれぞれを MeCab を用いて分割した。

提案手法 A　訓練コーパスに文字単位の初期分割を与え、提案手法で連結手順の学習を行い、テストコーパスに適用した。

提案手法 B　MeCab を用いて訓練コーパスとテストコーパスに初期分割を与え、提案手法で連結手順の学習を行い、テストコーパスに適用した。

用いたコーパスは、日本語話し言葉コーパス [10](以降 CSJ) に収録されている、「対話」（インタビュー 2 種、課題指向、自由）の転記テキスト全てを連結したものである。このコーパスの総長は、259150 文字（前処理[*4] 済み）である。コーパスの前半 20/21 を言語モデルの訓練コーパス、残り 1/21 をテストコーパスとした。

バックオフ 2-gram 言語モデルを構築するために Palmkit [11] を用いた。対象コーパスが小さいため (表 1)、語彙数の制限は行わなかった。

本実験で構築した言語モデルは互いの語彙が一致しないため、言語モデル評価指標として広く用いられている単語当たりパープレキシティではなく、文字当たりパープレキシティ [1]（以降 PP）によって評価を行う。

4.2 結果と考察

単語分割されたコーパスの統計情報を表 1, 表 2 に、構築された言語モデルの PP を表 2 に示す。この実験で、

[*3] $A\neq B$ を仮定している。$A=B$ の場合の差分を厳密に求めるのは困難なため、$A=B$ であってもこの式で近似する。

[*4] 文字化不可能な言語音の変異 (形式 : "<**文字列**>")、コメント（行頭に"**%**"）、無音区間の情報は全て除去した。文字化可能な言語音の変異 (形式 : "(**文字列**)") は括弧を除去して残した。ただし、転記者による注釈（";" 以降）は除去した。

手法	語彙数	延べ単語数	平均単語長
提案手法 A	8644	87075	2.83 文字
提案手法 B	10342	80908	3.05 文字
MeCab	6212	139018	1.78 文字

表 1　単語分割された訓練コーパスの統計情報

手法	平均単語長	文字当り PP
提案手法 A	2.60 文字	19.30
提案手法 B	2.84 文字	15.40
MeCab	1.75 文字	13.43

表 2　単語分割されたテストコーパスの統計情報と文字当たりパープレキシティ

提案手法 A,B はどちらも MeCab に劣る性能を示した。

しかし、訓練コーパスを観察したところ、MeCab が「家/で/ん/ち/ょんちょんちょんちょんできるっていうのも」と誤って解析した部分[*5]を、提案手法 A は「家/でん/ちょん/ちょん/ちょん/ちょん/できる/っていうのも」と分割していた。同様の形態素解析の失敗例は訓練コーパスで 16 だけだったが、形態素解析が困難な文字列上で言語モデルを構築する場合に、提案手法 A は有効だと思われる。

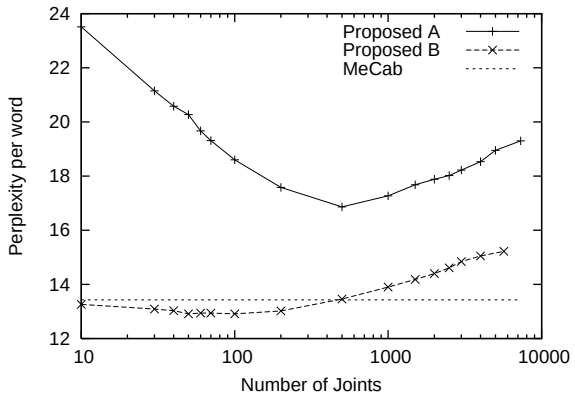

図 1　連結数を制限した場合の文字当たりパープレキシティの推移

提案手法 B が有効でなかったのは連結が多すぎてコーパスが疎になったためだと推測し、連結を途中で打ち切った場合の PP の評価を行った。横軸に打ち切るまでの連結の数、縦軸に PP をとったグラフを図 1 に示す。提案手法 (Proposed) の PP の谷は全連結が終わるより前であり、連結の学習が過学習に陥っていたことが分かる。適切な終了時点を選ぶことができれば、提案手法 B は形態素解析による分割を改善できるといえる。

本実験ではコーパスが小さいため無視できたが、より大規模なコーパスに適用する場合、学習に要する時間[*6]（1 ステップ毎に $O(N + V^3)$）を削減する必要がある。

5　おわりに

本稿では、2-gram 符号化に基づく最小記述長基準によって単語の連結を学習する、教師なし単語分割の手法を提案した。生コーパスに提案手法を適用し、形態素解析が失敗する文字列で提案手法が有効であることを示した。形態素分割済みコーパスに提案手法を適用し、連結を適切な回数で打ち切ることができれば、形態素分割より高精度な言語モデルが得られることを示した。今後は、形態素解析と提案手法の組み合わせ方、連結を打ち切る回数を学習する方法を検討する予定である。

参考文献

[1] 中川聖一, 赤松裕隆, 西崎博光. 音声認識用言語モデルのためのタスク適応化と定型表現の利用. 自然言語処理, 第 6(2) 巻, pp. 97–115, 1999.

[2] Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of International Conference Computational Linguistics*, 2004.

[3] Laura Mayfield Tomokiyo and Klaus Ries. What makes a word: Learning base units in Japanese for speech recognition. In *ACL Special Interest Group in Natural Language Learning*, pp. 60–69, 2004.

[4] 山本博史, 菊井玄一郎. 教師なし学習による文の分割. 言語処理学会第 8 回年次大会, pp. 579–582, 2002.

[5] 森大毅, 阿曽弘具, 牧野正三. 単語知識を必要としない高精度な言語モデル. 自然言語処理, 第 6(2) 巻, pp. 29–40, 1999.

[6] 工藤拓. *MeCab: Yet Another Part-of-Speech and Morphological Analyzer*, 2006.

[7] 伊藤秀夫. Suffix array を用いた日本語単語分割. 情報処理学会研究報告, 第 1997-NL-49 巻, pp. 47–54, 1999.

[8] 永田昌明. 単語頻度の再推定による自己組織化単語分割. 情報処理学会研究報告, 第 1997-NL-85 巻, pp. 9–16, 1997.

[9] Shlomo Argamon, Navot Akiva, Amihood Amir, and Oren Kapah. Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of International Conference Computational Linguistics*, 2004.

[10] 国立国語研究所, 情報通信研究機構. 日本語話し言葉コーパス, 2004.

[11] 伊藤彰則. *Palmkit version 1.0.31*, 2006.

[*5] この発話では「ちょんちょん」は三味線の音の擬音語だった。

[*6] 本実験の場合、Xeon 3.0GHz の計算機で約 18 時間を要した。