形態素周辺確率を用いた分かち書きの一般化とその応用

工藤 拓

NTT コミュニケーション科学基礎研究所 taku@cslab.kecl.ntt.co.jp

1 はじめに

形態素解析は, 自然言語処理における基礎技術のひとつであり, 構文解析, 機械翻訳, 自動要約など広く応用されている. 同技術は, 入力されたテキストを形態素 (単語) の頻度付き集合 (以下 bag-of-words BOW) に変換するために利用されることがとくに多い. BOW によりテキストは単純な数値ベクトルになり, 既存の機械学習や統計的手法を適用しやすくなる. BOW は, 大規模テキストを扱う情報検索, テキストマイニング, 文書分類といった分野で幅広く利用されている.

テキストを BOW に変換する際の問題点の 1 つに, 語の抽出単位の「曖昧性」や「ずれ」がある. 例えば, 以下のような例はどちらの分割も可能であろう.

- 本部長 → 本部/長 or 本/部長
- 応援団員 → 応援団/員 or 応援/団員

さらに, 日本語には複合語で表現された多くの固有名詞が存在し, その分割単位の定義はさまざまである.

- 横浜市役所 → 横浜/市/役所 or 横浜市/役所 or 横浜/市役所
- 関西国際空港会社連絡橋 → 関西国際空港会社連絡橋 or 関西国際空港会社/連絡橋 or 関西国際空港/会社/連絡橋 ... 以下略

また, 解析器自身も不整合な分割を出力する場合がある. (ipadic + ChaSen の例).

- 成田空港 → 成田空港 (一語)
- 宮崎空港 → 宮崎/空港 (二語)

このような抽出単位の「ずれ」は, 個々の応用でさまざまな問題を生む. 例えば, 形態素解析を用いた全文検索システムでは, 「成田」で検索しても「成田空港」は見つからないといった検索漏れが発生しやすい. また, 「京都大学」とその省略表記である「京大」の BOW に基づくテキスト間類似度[1]は, 前者が 1 語として分かち書きされてしうために, 0 となってしまう. それでは, 最適な単位とはいったい何なのだろうか? 言語学的な考察は可能かもしれないが, 最適な単位は個々の応用に強く依存し, 工学的な意味で万能な単位を得ることは事実上不可能に近い.

一方, このような泥沼的な議論を避け, あえて形態素解析を使わないという立場も存在する. このような立場では, テキスト中の 1 つの文字を 1 つの単語とみなして分かち書きを行う. 再現率は向上するが, それには検索ノイズの影響を犠牲にしなければならない. 例えば, 「京都」で検索して「東京都」が見つかったり, 「パン」で検索して「ルパン」が見つかってしまう問題がある.

さて, このような背景を踏まえると, 1 つのアイデアが生まれるだろう. つまり, 形態素解析技術を最大限に活用する高度な分かち書き処理と, 文字単位 (もしくは文字 n-gram) のような言語処理を使わない分かち書き処理のそれぞれの「いいとこ取り」ができないだろうか? さらに, これらの両極端の立場を 1 つの枠組みで単一化できないだろうか? 本稿の目的ははこのような要望を実現するための 1 手法を提案することにある. 同手法を使うことで, 検索ノイズの影響を抑えながら再現率を上げるといったことが可能となる.

以下, 2 章で提案手法の基本的アイデアを, 3 章でその具体的な実装方法を述べる. さらに, 4 章で検証実験を行い, 5 章で本論文をまとめる.

[1] コサイン類似度等

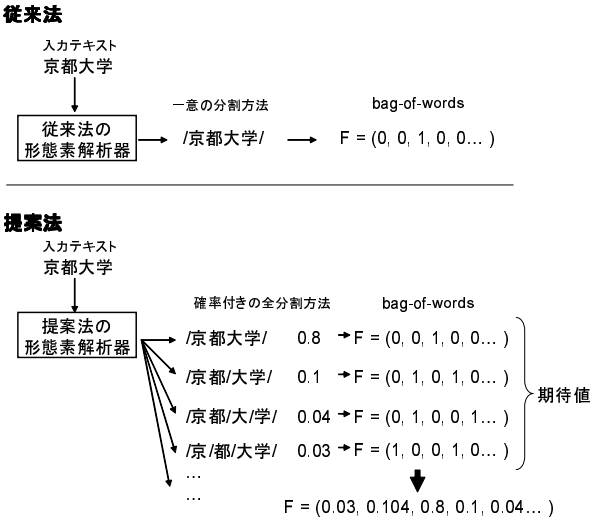

図 1: 従来法と提案法の構成図

2 基本的なアイデア

提案法の基本的なアイデアは, 解析器が出力する一意の解だけでなく, 可能なすべての解を用いて BOW を作成することにある. 図 1 に従来法と提案法の構成図を示す. 提案法は可能なすべての分割方法を, コスト (連接コストや単語生起コスト) を反映した確率値と共に出力する. 最終的な入力テキストの BOW は, 可能な分割方法それぞれの BOW の期待値をとることで算出する. すべての出力候補を用いることで, 辞書にマッチする可能なすべての単語が抽出され, 再現率の向上が計れる. 一方, 連接コストや単語生起コストに基づく確率値を用いるため, 不適切な分割には低い確率が割り当てられ, ノイズの影響を受けにくいという性質を持つ.

3 定式化

本章で具体的な定式化を行う. 提案する形態素解析器は最小コスト法を基にしている. まず, 最小コスト法について簡単におさらいし, その後詳細を説明する.

3.1 準備: 最小コスト法

最小コスト法に基づく形態素解析器 $A = (D, b, e, \pi, a)$ は以下で与えられる.

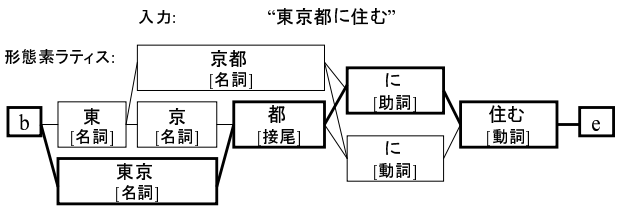

図 2: 形態素ラティスの例

- $m = \langle w, t \rangle$: 形態素
  ただし w は表層文字列 (私, 動く.. 等), t は対応する品詞 (名詞, 代名詞, 動詞 等). 表層文字列 w をここでは単に「単語」と呼ぶ
- $m \in D$: 形態素集合 (一般に辞書と呼ばれる)
- $b \notin D$: 初期状態形態素
- $e \notin D$: 終了状態形態素
- $\pi : m \in D \rightarrow \mathbb{R}$: 形態素生起コスト.
  ここでは, 便宜的に $\pi(m)$ という実数値を返す関数として表記する.
- $a : m \in (D \cup \{b, e\}) \times m \in (D \cup \{b, e\}) \rightarrow \mathbb{R}$: 形態素連接コスト.
  ここでは, 便宜的に $a(m_1, m_2)$ という実数値を返す関数として表記する.

入力文字列 x が与えられると, まず x に対し, 辞書 D にマッチする可能な形態素列すべてを表現した形態素ラティスを作成する. 図 2 にその具体例を示す. 開始状態 b から終了状態 e までには, いくつかの有限の**経路**がある. 例えば, 図 2 には以下の 6 通りの経路が存在する. 図 2 における太枠の経路は, 入力に対する正しい経路 (解析結果) を示す.

ここで, 入力 x に対応する可能なすべての経路の集合を $\mathcal{Y}(x)$ と表記する. さらに, その 1 つを $y \in \mathcal{Y}(x)$ と表記する. y は次のような形態素の系列となる.

$$y = (b, m_1, m_2, \ldots, m_{|y|}, e)$$

ただし, m_k は経路 y の k 番目の形態素である. $|y|$ は, 経路 y 上にある形態素の個数である (ただし文頭/文末は含めない).

次に, 各経路 y についてコスト $cost(y)$ を以下のように定義する.

$$cost(y) = \sum_{j=1}^{|y|} \pi(m_j) + a(b, m_1) + \sum_{j=1}^{|y|-1} a(m_j, m_{j+1}) + a(m_{|y|}, e)$$

$cost(y)$ は, 経路 y 上にある形態素列の形態素生起コスト及び形態素連接コストの総和として定義される.

コスト最小法に基づく形態素解析とは, $cost(y)$ が最小となるような経路 $\hat{y}$ を経路集合 $\mathcal{Y}(x)$ から発見し, それを出力する手法として定式化される.

$$\hat{y} = \underset{y \in \mathcal{Y}(x)}{argmin}[cost(y)] \tag{1}$$

$\hat{y}$ を効率的に発見するアルゴリズムに動的計画法の一種である Viterbi アルゴリズムがある.

コスト (単語生起コスト, 連接コスト) は, 人手で与えたり, 隠れマルコフモデルや Conditional Random Fields (CRFs) [1] 等を用いて統計的に求める[2]. ただし, 提案手法はコストの推定方法に依存せず, 既存の解析器の拡張として実装される.

3.2 マルコフ確率場

最小コスト法に基づく形態素解析器について, 以下のような条件付き確率 $P(y|x;\theta)$ を与える.

$$P(y|x;\theta) = \frac{\exp[-cost(y) \cdot \theta]}{\sum_{y' \in \mathcal{Y}(x)} \exp[-cost(y') \cdot \theta]} \tag{2}$$

ただし, $\theta \in \mathbb{R}^+$ は逆温度定数である. $P(y|x;\theta)$ は, 出力系列 y が, 全候補 $\mathcal{Y}(x)$ の中でどれだけ出力しやすいかを表現する確率である. ここで, 確率 $P(y|x;\theta)$ を最大にするような y は, 最小コスト法で得られる解と同一になることに注意されたい ($argmax_{y \in \mathcal{Y}(x)} P(y|x;\theta) = argmin_{y \in \mathcal{Y}(x)} cost(y)$). 逆温度定数 θ は確率分布の鋭さを与える. θ を小さくすると, 各出力確率 $P(y|x)$ は均一になる. 逆に θ を大きくすると, 最小コストとなる経路 $\hat{y}$ の出力確率を 1 にし, 残りを 0 にするような効果が得られ, 最適解を重要視する.

式 (2) のような定式化はマルコフ確率場と呼ばれ, 最大エントロピー法や, CRFs に用いられている. CRFs は 式 (2) に対し最尤推定を行うことで, 連接コスト, 単語生起コストを算出する.

3.3 周辺化 bag-of-words

形態素列 y を bag-of-words ベクトルに変換する写像 $\Phi : y \rightarrow \mathbb{R}^L$ は, 通常, 以下のように与える[3].

$$\Phi(y) \quad = \quad \{f(w_1, y), \ldots, f(w_L, y)\} \in \mathbb{R}^L$$

ただし $f(w, y)$ は, 系列 y に単語 w が出現する回数, L は総単語異なり数である. 従来法では, 入力テキスト x に対応する BOW ベクトル $\mathbf{F}(x) \in \mathbb{R}^L$ は, 最適解 $\hat{y}$ を用いて以下のように導出される.

$$\mathbf{F}(x) = \Phi(\hat{y}), \ \text{ただし} \ \ \hat{y} = \underset{y \in \mathcal{Y}(x)}{argmin}[cost(y)] \tag{3}$$

提案法では, 以下のように全系列 $y \in \mathcal{Y}(x)$ に対応する BOW ベクトル $\Phi(y)$ の期待値をとることで最終的な BOW ベクトル $\mathbf{G}(x;\theta)$ を作成する.

$$\begin{aligned} \mathbf{G}(x;\theta) &= \{g(w_1, x;\theta), \ldots, g(w_L, x;\theta)\} \in \mathbb{R}^L \\ &= \sum_{y \in \mathcal{Y}(x)} P(y|x;\theta) \cdot \Phi(y), \quad \text{ただし}, \\ g(w, x;\theta) &= \sum_{y \in \mathcal{Y}(x)} P(y|x;\theta) \cdot f(w, y). \end{aligned}$$

この時, $\theta \rightarrow \infty$ の極限をとると

$$\lim_{\theta \rightarrow \infty} \mathbf{G}(x, \theta) = \mathbf{F}(x)$$

[2] Juman のコストは人手によって, ChaSen のコストは HMM を用いて学習データを用いて統計的に与えられている.

[3] bag-of-morphs, bag-of-POS, bag-of-bigrams といったベクトルを作る場合でも以下同様な議論ができる.

となり, 従来法と同一になる. また, $\theta = 0$ とすると, すべての経路が等確率となる. 長さの短い単位の形態素は分割の組み合わせを増やし, 多くの経路候補の中に含まれる. つまり $\theta = 0$ の場合は, 長さの短い単語の頻度 $g(w, x; \theta)$ が大きく設定され, 結果として文字単位で分かち書きを行うような効果が生まれる[4]. 以下, 周辺化 BOW ベクトル $\mathbf{G}(x; \theta)$ の性質についてまとめる.

- 全候補の列挙
  可能なすべての分割方法を考慮するため, 文書の索引語として用いた場合, 検索漏れが生じにくい.
- 頻度の一般化
  $\mathbf{G}(x; \theta)$ の要素 $g(w, x; \theta)$ は, 入力 x に単語 w が出現する**確率的な頻度**とみなすことができる. そのため, 頻度情報を用いる手法 (tf·idf, ベクトル空間モデル, Naive Bayes 等) の入力としてそのまま用いることができ, 処理の独立性が高い.
- コストを考慮した分割
  $\mathbf{G}(x; \theta)$ の要素 $g(w, x; \theta)$ は, 単語生起コスト, 連接コストを反映するように設定される. そのため, 「東京都」の中の「京都」という単語頻度 g は自然と小さく設定され, ノイズの影響を抑えることができる.
- θ による制御
  θ は形態素解析の結果を 100%信用する従来法と, それらを用いず, 文字単位で分かち書きをする手法とを一次元に接続するパラメータである. ユーザはこれらの 2 つのバランスを自由にとることができる.

3.4 動的計画法を用いた効率的な計算

出力系列の数は入力文字列長に対し指数的に増えるため, 式 (4) の適用は非現実的である. そこで, 動的計画法の一種である Forward-Backward アルゴリズムの変種を用い効率良く計算する.

まず, 形態素ラティス中の各形態素 m について以下の値 $\alpha(m), \beta(m)$ を計算する (それぞれ再帰的な定義になっていることに注意).

$$
\begin{aligned}
\alpha(b) &= 1.0, \quad \beta(e) = 1.0 \\
\alpha(m) &= \sum_{m' \in \mathcal{L}(m)} \alpha(m') \cdot \exp[-\big(\pi(m) + a(m', m)\big) \cdot \theta] \\
\beta(m) &= \sum_{m' \in \mathcal{R}(m)} \beta(m') \cdot \exp[-\big(\pi(m) + a(m, m')\big) \cdot \theta] \\
Z &= \alpha(e) \quad (= \beta(b))
\end{aligned}
$$

ただし, $\mathcal{L}(m)$ は, 形態素 m に対し左から接続する形態素の集合, $\mathcal{R}(m)$ は, 形態素 m に対し右から接続する形態素の集合である. α, β, Z を用いると, 式 (4) の左辺は次のように計算される.

$$
g(w, x; \theta) = \sum_{m \in \mathcal{M}(x,w)} \frac{\alpha(m) \cdot \beta(m) \cdot \exp(\pi(m) \cdot \theta)}{Z} \quad (4)
$$

ただし, $\mathcal{M}(x, w)$ は, 形態素ラティス中の全形態素のうち, 単語部分が w である形態素の集合である. $g(w, x; \theta)$ を求めるための計算量は文長のみに依存し, 効率良い計算が行える.

[4]文字単位の分割候補は, 辞書の定義によっては $\mathcal{Y}(x)$ に存在ない場合もある. ただし, 全 1 文字を 1 単語として辞書に登録すれば完全に文字単位の分かち書きが重要視されることになる.

3.5 Kernel 法との関連

Support Vector Machiens に代表される Kernel 法は, 事例の類似度 (内積) のみを入力データの表現方法として用いる機械学習手法である. テキストを学習の対象にする場合は, テキスト間の内積を定義する必要がある. テキストのための Kernel として, 以下のような BOW ベクトルの内積を用いることが多い.

$$
K(x_1, x_2) = \mathbf{F}(x_1) \cdot \mathbf{F}(x_2)
$$

提案手法についても, $\mathbf{G}(x; \theta)$ どうしの内積をとることで, Kernel を与えることが可能である.

$$
\begin{aligned}
K(x_1, x_2; \theta) &= \mathbf{G}(x_1; \theta) \cdot \mathbf{G}(x_2; \theta) \\
&= \sum_{y_1 \in \mathcal{Y}(x_1), y_1 \in \mathcal{Y}(x_1)} \Phi(y_1) \cdot \Phi(y_2) P(y_1|x_1; \theta) P(y_2|x_2; \theta)
\end{aligned}
$$

上式で与えられる Kernel は, Marginalized Count Kernel (MCK) [2] の特殊形となる. MCK は, 観測データ x が任意の隠れクラス y に帰属する事後分布 $p(y|x)$ を求め, x, y のペア $z = \langle x, y \rangle$ を入力とするような任意の Kernel $K(z_1, z_2)$ の値を帰属確率について期待値をとることで内積を求める. 本稿の文脈では, x は入力文, y は形態素系列, z は BOW ベクトル $\Phi(y)$, $p(y|x)$ はマルコフ確率場で与えられる事後確率となる.

3.6 実行例

本手法を形態素解析器 MeCab[5] 上に実装した. コスト値は 文献 [1] の CRFs に基づく手法を使い導出した. 表 (1) に θ とそれに対応する BOW の関係を, 3 つの入力文「京都大学」「東京都庁」「本部長」について示す.

θ が大きいとき ($\theta = 10$), 一意の最適な分割に基づいて BOW が作成されている. これは従来法と同一である. θ を徐々に小さくしていくと, 細かい分割に対し大きい頻度が割り当てられるようになる.

また, 「東京都庁」の中の部分文字列「京都」や「本部長」の中の部分文字列「部」の頻度は, θ を小さくしても, 他に比べ大きくなりにくい (例えば $\theta = 0.5$ の場合のそれぞれの頻度に注目). 連接コストや単語生起コストに基づいて確率値を計算しおり, これらの部分文字列が入力文中で単語として抽出される確率が自然と低く設定されるためである.

4 実応用での検証実験

4.1 要約文 − 要約元文の対応付け

人手で作成された要約と, それに対応する要約元文を対応付ける実験である. 実験には TSC2 の複数文要約のデータを用いた. ただし, 複数の要約文から要約が作成されている要約文は削除した. 実験には, 465 文の要約文を用い, 要約文 1 に対し平均 98.5 文の要約元文候補がある. 要約文をクエリとし, 要約元文候補各文とのコサイン類似度を計算する. 類似度順に候補文をソートし, 1 位に正解文がくる割合 (**正解率**), および正解文の**平均順位**で比較を行った. また, BOW の作成の際, 助詞といった機能語は削除している. これは式 (4) の $\mathcal{M}(x, w)$ に品詞の制約を加えることで実現できる.

表 (3) に θ と正解率, 平均順位の関係を示す. θ を徐々に小さくしていくと, 正解率, 平均順位とも性能が向上している. ただし, 小さくしすぎると性能が低下してい

[5]http://chasen.org/~taku/software/mecab

表 1: θ と BOW ベクトル ($g(x, w; \theta)$) の関係
上から「京都大学」「東京都庁」「本部長」

	θ				
位置/単語	10	1.0	0.5	0.1	0.01
0-2 京	0.0000	0.0000	0.0014	0.5356	0.8520
0-4 京都	0.0000	0.2512	0.3514	0.3589	0.1339
0-6 京都大	0.0000	0.0314	0.1440	0.0720	0.0111
0-8 京都大学	1.0000	0.7174	0.5032	0.0335	0.0030
2-4 都	0.0000	0.0000	0.0014	0.5356	0.8520
4-6 大	0.0000	0.0004	0.0255	0.6670	0.8659
4-8 大学	0.0000	0.2509	0.3272	0.2275	0.1200
6-8 学	0.0000	0.0318	0.1695	0.7390	0.8771
0-2 東	0.0000	0.0004	0.0266	0.7173	0.9322
0-4 東京	1.0000	0.9996	0.9734	0.2827	0.0678
2-4 京	0.0000	0.0000	0.0006	0.4572	0.8132
2-6 京都	0.0000	0.0004	0.0259	0.2601	0.1190
4-6 都	0.0000	0.0135	0.1526	0.5909	0.8180
4-8 都庁	1.0000	0.9861	0.8215	0.1490	0.0630
6-8 庁	0.0000	0.0139	0.1785	0.8510	0.9370
0-2 本	0.0000	0.2770	0.4083	0.6894	0.8720
0-4 本部	1.0000	0.7230	0.5917	0.3106	0.1280
2-4 部	0.0000	0.0000	0.0044	0.5286	0.8244
2-6 部長	0.0000	0.2769	0.4039	0.1609	0.0476
4-6 長	1.0000	0.7231	0.5961	0.8391	0.9524

図 3: 対応付け結果

θ	正解率	順位
10	60.9	13.4
1.0	60.9	12.7
0.1	61.6	11.0
0.05	62.2	11.0
0.04	62.4	11.0
0.03	62.2	11.1
0.01	60.9	11.2
0.00	60.9	11.3

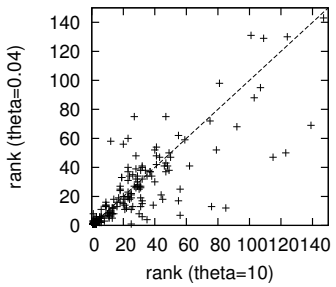

図 4: $\theta = 0.04$ vs $\theta = 10$

る. また図 (4) に $\theta = 0.04$ と $\theta = 10$ の順位関係をプロットした. グラフの右下に多くの点が集中しており, $\theta = 0.04$ の場合の順位が $\theta = 10$ の順位に比べ高くなっていることを示している.

形態素解析器の結果と文字単位の結果を本手法によって混合することで性能向上が認められ, 我々の期待通りの結果となった.

4.2 テキスト分類

テキスト分類の応用で本手法の効果を検証した. テキスト分類の代表的なアルゴリズムに Naive Bayes がある. 同手法はテキスト中の各単語の出現を独立と考え, 単語の出現頻度を元にカテゴリ c とテキスト d との同時確率 $p(c, d)$ を計算する. 式 (4) で与えられる頻度 $g(w, x, ; \theta)$ を同時確率を求める際の頻度として利用する. 実験データとして**「掲示板」「論文」**の 2 つを設定した. **掲示板**は, Web の掲示板の投稿を入力に, 掲示板のカテゴリ (インターネット, エンターテイメント, グルメ.. など計 10 カテゴリ) を出力するタスク, **論文**は論文のタイトルを入力に, 雑誌名 (計測自動制御学会, 電気学会, 日本薬学会.. など計 10 カテゴリ) を出力するタスクである. それぞれ, ランダムに 200 事例 (**small**), 1,000 事例 (**mid**) 10,000 事例 (**large**) を選択し, それぞれのサイズについて 5 交差の交差検定で評価を行う. また, 確率を求める際には, 0 頻度を避けるためスムージングを行う. 本実験では単純な加算スムージングを用い, そのパラメータは交差検定を用いて設定した.

表 2: テキスト分類の結果 (マイクロ平均 F 値)

	掲示板			論文		
θ	small	mid	large	small	mid	large
10	0.409	0.596	**0.730**	0.410	0.558	0.718
1.0	0.397	0.597	0.728	0.427	0.559	0.719
0.5	**0.414**	0.605	**0.730**	**0.433**	0.563	**0.722**
0.1	0.368	0.619	0.725	0.426	**0.570**	0.720
0.01	0.341	**0.620**	0.714	0.394	0.543	0.712

表 (2) に, 各 θ に対する F 値を示す. 掲示板の *mid* 以外は, おおよそ $\theta = 0.1$〜0.5 付近に F 値のピークがある. この結果からも, 全候補の期待値を取る本手法の有効性が確認できた.

5 まとめと今後の課題

従来の形態素解析の研究の多くは, 事前に与えられた唯一の単語分割定義をどれだけ精度良く再現できるかに焦点が当てられていた. しかし, ある基準で分割方法を定義しても, その定義が個々の応用に対して最適なものとは限らない. また, 現実タスクに潜む分割の多様性に柔軟に対処できているとは言えない.

本手法と従来法の決定的な違いは, 本手法が複数の分割の期待値を算出し, 分割方法の**多様性を認める立場**をとっている点にある. 浅原らは名詞複合語の構成語を木構造で表現し辞書に格納する手法を提案し, 分割の多様性を別の立場, 方法論から議論している [3]. また, Unidic プロジェクト [4] も, 単語の抽出単位について複数の定義 (0〜3 階層) を与え, 多様性を表現している. 提案手法は浅原らの手法や Unidic と相補的に利用可能である. 例えば, 複合語の木構造を半自動で 構築する目的に本手法が使えるだろうし, $\theta = \infty$ の解の選択に辞書の情報が利用できる.

今後は, θ をタスクに応じて自動的に設定きるような手法を提案したい. 少なくとも, データ量や使用語彙のバラツキによって, θ の最適点は変わりそうである. 最適点の推定法として, 単語分割方法まで含めたテキスト生成モデルを考えるのも面白いだろう. また, 提案法の一般的な拡張を考えると, テキスト中の全部分文字列への重み付けが BOW 作成と言える. この観点から議論や手法を発展させていくことも興味深い.

参考文献

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

[2] Koji Tsuda, Taishin Kin, and Kiyoshi Asai. Marginalized kernels for biological sequences. *Bioinformatics*, Vol. 18, No. 7, pp. 268–275, 2002.

[3] 浅原正幸, 米田隆一, 山下亜希子, 伝康晴, 松本裕治. 語長変換を考慮したコーパス管理システム. 情報処理学会論文誌, Vol. 43, No. 7, pp. 2019–2097, 2001.

[4] 伝康晴, 宇津呂武仁, 山田篤, 浅原正幸, 松本裕治. 話し言葉研究に適した電子化辞書の設計. 第 2 回話し言葉の科学と工学ワークショップ講演予稿集, 2002.