確率的ブロック編集距離

中谷 洸樹[†]　Andrew Finch[‡]　田中 久美子[†]　隅田 英一郎[‡]
九州大学 システム情報科学府[†]　独立行政法人 情報通信研究機構[‡]

{koki, kumiko}@cl.ait.kyushu-u.ac.jp[†]
{andrew.finch,eiichiro.sumita}@nict.go.jp[‡]

1　はじめに

2つの文字列の類似度を測る手法は、自然言語処理や機械学習等の分野で広く利用されており、その中でも編集距離は基本的な手法として頻繁に使われている。基本的な編集距離は、ある文字列を別の文字列に変形する際に必要な1文字ずつの編集操作（挿入・置換・削除）の最小回数、または最小コストである。

編集距離に関する研究においては様々なアイデアが提案されており、編集操作のコストを確率的に捉えたものや、1文字単位ではなく文字列 (ブロック) 毎の編集操作を可能にしたもの、距離そのものの定義を見直したものなどがある。中でも、ブロック編集距離 [1] は過去に提案されているが、実際の実現方法までは示されておらず、未完成に留まっている。これに対して近年、ベイズ推定を用いた文字列ペアのアラインメントを求める手法 [2] が提案されており、このブロック編集距離の実現のために転用することができる。本稿では実際にこれを行い、これまでに提案された他のアイデアを全て導入したブロック編集距離が、一連の編集距離群の中で最も性能が良いことを実験的に示す。

2　関連研究

本節では、基本的な編集距離の再確認の後、その後に提案された主要な編集距離のアイデアをまとめる。本稿での提案は、確率的にブロックの編集操作を行う編集距離に主眼を置いているが、アイデアとしてはそれ以外に前述のように距離の定義を見直すものもある。

以下では、基本的な編集距離、そして編集操作のコストを確率的に捉えた確率的編集距離、ブロックの編集操作が可能なブロック編集距離の後、編集距離の定義を見直したアイデアを概説する。

2.1　編集距離

編集距離は2つの文字列がどの程度異なっているかを表す類似度で、ある文字列を別の文字列に変形する時に必要な編集操作 (挿入・削除・置換) の最小回数、または最小コストを求めることで得られる。

$X = \{x_m\} = x_1^m = \{x_1, x_2, ..., x_m\}$、$Y = \{y_n\} = y_1^n = \{y_1, y_2, ..., y_n\}$ を与えられた2つの文字列とする。この時、各編集操作は以下のように表記する。このとき、$x_i \in \mathcal{A}$、$y_j \in \mathcal{B}$ とし、$\mathcal{A}$ と $\mathcal{B}$ はそれぞれ文字列 X と Y のアルファベットを表す集合である。また、空文字を ϵ とし、空文字の長さは 0 とする。

- 削除操作：$\langle x, \epsilon \rangle, x \in \mathcal{A}$
- 挿入操作：$\langle \epsilon, y \rangle, y \in \mathcal{B}$
- 置換操作：$\langle x, y \rangle, x \in \mathcal{A}, y \in \mathcal{B}$ (但し、$x = y$ のときは一致)

また、編集操作を与えたときに、それに対応するコストを返すコスト関数 c をおく。これにより、編集距離は式 (1) と定義される。

$$d(x_1^u, y_1^v) = \min \left\{ \begin{array}{l} c(\langle x_u, y_v \rangle) + d(x_1^{u-1}, y_1^{v-1}), \\ c(\langle x_u, \epsilon \rangle) + d(x_1^{u-1}, y_1^v), \\ c(\langle \epsilon, y_v \rangle) + d(x_1^u, y_1^{v-1}) \end{array} \right\} \quad (1)$$

コストは通常、編集操作の回数が用いられることが多いが、各編集操作に人手でコストを割り当てることもできる。そして、編集距離は動的計画法を用いて効率的に計算できる。

2.2　確率的編集距離

編集操作のコストを確率的に捉えることは、編集距離の自然な拡張であろう。文献 [3] では、この提案を行い、編集距離のコストを学習する手法を初めて提案した。

この確率的編集距離では、互いに関連のある文字列ペアを含んだ訓練データから、EM アルゴリズムを用いて各編集操作の確率を学習する。文字列を変形させる一連の編集操作 $e = \langle \cdot \rangle$ の配列をパス $z = \{e_1, ..., e_k\}$ とすると、パスの確率 $p(z)$ は式 (2) となる。

$$p(z) = \prod_{i=1}^{k} p(e_i) \quad (2)$$

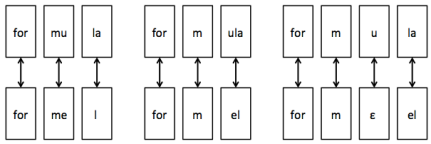

図 1: ブロック毎の編集操作

パス z のコストを $C(z)$ とすると、このコストはパスの確率の対数の負を計算することで得られる。

$$C(z) = -log(p(z))) = \sum_{i=1}^{k} c(e_i) = \sum_{i=1}^{k} -log(p(e_i)) \quad (3)$$

これを元に 2 つの確率的編集距離が定義できる。とりうる全てのパス $\mathcal{Z}$ に対して、式 (4) は、最小コストを持つパスのみ考慮した編集距離、式 (5) は、全てのパスのコストの合計値を編集距離としたものである。

$$d_{viterbi}(x_1^u, y_1^v) = \arg\min_{z \in \mathcal{Z}}(C(z)) \quad (4)$$

$$d_{stoch}(x_1^u, y_1^v) = \sum_{z \in \mathcal{Z}}(C(z)) \quad (5)$$

2.3 ブロック編集距離

ここまでは 1 文字単位の編集操作を対象としてきた。しかし、文字列を 1 つの単位として編集操作を行いたい場合が、特に自然言語処理の分野においては多々ある。ブロック編集距離は、従来の単一文字のみによる編集距離とは対照的に、図 1 のように複数文字を含むブロック単位での挿入・置換・削除も可能にした手法である。

このブロック毎の編集操作が可能になることの利点として、文字列の対応関係を汎化して捉えられることが挙げられる。そうすれば、例えば局所的に対応関係が交差するような一連の編集操作が扱えるようになる。また、接尾辞のような文脈情報もブロックによって捉えることができる。

以上のブロック編集操作に基づく枠組みに、文献 [1] は初めて言及している。しかし、コストの推定方法が明確に提示されていないため実用化するには未完成な論文に留まっている。本研究の意義は、これを完成させることにある。

2.4 編集距離の定義の再考

編集距離に関する提案の中には、距離に着目したものもいくつかある。この節では、その主要なアイデアをまとめる。なお、以下で紹介するアイデアはブロック操作に基づく確率的編集距離に対しても適用可能であり、実際、4 章ではこれらの工夫を確率的ブロック編集距離に適用した結果も合わせて示す。

2.4.1 正規化編集距離

2.1 節で示した基本的な編集距離では、距離の大きさが比較する文字列の長さと関連しているため、類似度を捉える上で不都合な場合もある。例えば、{newspaper,paper}{note,kids} の 2 つのペアを考えた時、直観的には前者の方が似ていると見なせるが、編集距離の観点では両者の類似度は同じと見なされる。

そこで、編集距離を正規化する手法 [4] が提案された。取りうる全パス $\mathcal{Z}$ に対して、コスト $C(z)$ をパスの長さ $L(z)$、つまりパス z に含まれる編集操作の回数で割った値の最小値を正規化編集距離 d_{ned} と定義している。

$$d_{ned}(x_1^u, y_1^v) = \arg\min_{z \in \mathcal{Z}} \frac{C(z)}{L(z)} \quad (6)$$

2.4.2 Sum-over-Paths 編集距離

基本的な編集距離の場合、コストが最も低いパスのみを考慮するか、式 (5) のように全パスのコストの合計値を編集距離としている。それに対して、パスの生起確率を元にしてコストを重み付けする Sum-over-Paths(SoP) 編集距離 [5] が提案されている。

これは、各パスの生起確率 $P(z)$ を用いて算出した全パスに対するコストの期待値に基づいた編集距離で、式 (7) で定義される。

$$\begin{aligned} d_{SoP}(x_1^u, y_1^v) &= \sum_{z \in \mathcal{Z}} C(z)P(z) \\ &= \sum_{z \in \mathcal{Z}} C(z)\frac{\exp\left[-\theta C(z)\right]}{Z} \end{aligned} \quad (7)$$

Z は分配関数、θ はパラメータである。θ は全パスが計算に考慮される度合を制御する。θ を無限に近づけると、分布はコストがより低いパスを重視する分布となる。逆に θ を 0 に近づけると分布は均一になり、全てのパスが等しく距離に考慮される。このとき、SoP 編集距離は確率的編集距離の式 (5) と等価と見なすことができる。そして、これをパスの長さの期待値で正規化した正規化 SoP 編集距離 [6] は式 (8) で定義される。このとき、$\mathbb{E}(L)$ はパスの長さの期待値である。

$$d_{NSoP}(x_1^u, y_1^v) = \frac{d_{SoP}(x_1^u, y_1^v)}{\mathbb{E}(L)} \quad (8)$$

3 確率的ブロック編集距離

3.1 ブロック操作が可能な編集距離

本稿の 2.1 節、2.2 節をふまえて確率的ブロック編集距離の定義を与える。入力された文字列ペア $X = x_1^m, Y = y_1^n$ に対して、式 (9) で再帰的に定義できる。

$$d(x_1^u, y_1^v) = \sum_{i=0}^{u} \sum_{\substack{j=0 \\ (i \neq j \neq 0)}}^{v} \left(c(\langle x_{u-i+1}^u, y_{v-j+1}^v \rangle) + d(x_1^{u-i}, y_1^{v-j}) \right) \tag{9}$$

c はコスト関数である。また、$x_{u+1}^u := \epsilon, y_{v+1}^v := \epsilon, d(\epsilon, \epsilon) := 0$ であり、$d(\epsilon, \epsilon)$ により再帰式は終了する。式は、とりうる全てのパスのコストの合計値を距離として用いており、動的計画法により効率的に計算できる。そのため、本手法は、確率的編集距離における式 (5) を、ブロック操作ができるように拡張したものとして位置づけられる。

3.2 ブロック編集操作のコスト推定

ブロック編集距離を実現するには、各編集操作のコストが必要になる。しかし、文献 [1] はコストの推定方法に関して未完成であるため、これを実現することはできない。

単純に 2.2 節の確率的編集距離の拡張を行い、EM アルゴリズムによるコストの推定も考えられる。しかし、過学習を抑える上でもベイズ推定を用いる方がよいであろう。本稿ではこの目的のために、文献 [2] で提案された手法を転用する。

文献 [2] は、Bayes 推定を用いてアラインメントを行う方法を提案しているものである。アラインメントを利用して、編集距離のコストを推定する事ができる。なぜなら、アラインメントをとるということは、対応するブロック同士の対応関係をとることである。ブロックの対応度合いを確率的に捉えることは、すなわち編集操作のコストを考えていることと同じと見なせる。ベイズ推定によるアラインメントでは、実際にアラインメントの確率を推定している。そのため、アラインメントを編集操作に転用することができる。

このアラインメントによって得た確率の対数の負を、編集操作のコストとして用いる。アラインメントを求める手法の詳細は文献 [2] に記載されている。

4 評価実験

4.1 実験データ

今回の実験では 2 つのデータセットを用いた。1 つは、文献 [2] の評価で用いられた英語とローマ字変換した日本語の翻字ペアのデータセットである。もう 1 つは文献 [7] のタスクで用いられた英語とアルファベット表記に変換したロシア語の翻字ペアのデータセットである。翻字ペアとは、{“PARI”,“PARIS”}{“TEKISUTO”,“TEXT”} のようなに発音に基づいて翻訳された単語のペアである。各データセットがもつ単語のペア数は、英語と日本語のデータセットで 3738 ペア、英語とロシア語のデータセットで 780 ペアである。

4.2 実験

本実験では、全ての編集距離の可能性に関して網羅的に検証を行う。編集距離に関しては様々なアイデアの提案がなされている。これらのアイデアの可能な組み合わせを全て考えた結果、図 2,3 の左側に挙げた全 10 通りの編集距離が考えられる[1]。ブロック編集操作を行わない編集距離は図 2,3 の上 6 つの手法、それに対してブロック編集操作を行う編集距離は下 4 つの手法である。本実験のポイントは、ブロック編集操作を行う手法とそうでない手法の性能の違いを比較することである。

3 節での説明のように、コストを学習するには訓練データが必要となる。そこで、10 回交差検定による実験を行い、エラー率で評価する。テストデータ内の各ペアの英単語に対して、日本語ないしロシア語の全単語のうち、最も編集距離の値の小さい単語をペアとする。これが、もともとの正解ペアに不一致である場合をエラーとし、テストデータ内のペアのうちいくつエラーであったかを計測する。

4.3 実験結果・考察

実験結果を図 2,3 に示す。これらの結果から、ブロック編集操作を行う編集距離の方が全般的に見てエラー率が低いことが見られる。このことから、ブロック化の効果を示しているだろう。とくに、距離に対して何も適用していない単なる確率的ブロック編集距離でも、エラー率が改善されていることが分かる。中でも、日本語においては通常の編集距離のエラー率が 53.05%であるのに対して、SoP 確率的ブロック編集距離では 7.2%まで大幅に改善されている。一方のロシア語においては、通常の編集距離でエラー率が 6.31%であったのが、正規化 SoP ブロック編集距離ではエラー率が 2.11%にまで改善されている。これらの結果から総じて、ブロック毎の編集操作の有効性は示されていると考える。

別の観点から、日本語とロシア語の結果を比較すると、基本的な編集距離のエラー率が日本語の場合で

[1] 確率的編集距離については 2 つの定義がある。そのうち式 (5) は、$\theta \to 0$ の時の SoP 編集距離と等価であるため、SoP 編集距離に含める。したがって、式 (4) を確率的編集距離とする。また、SoP 編集距離のパラメータ θ は予備実験で $\theta = 1$ の時が最も良い性能を示したため、この値を用いる。

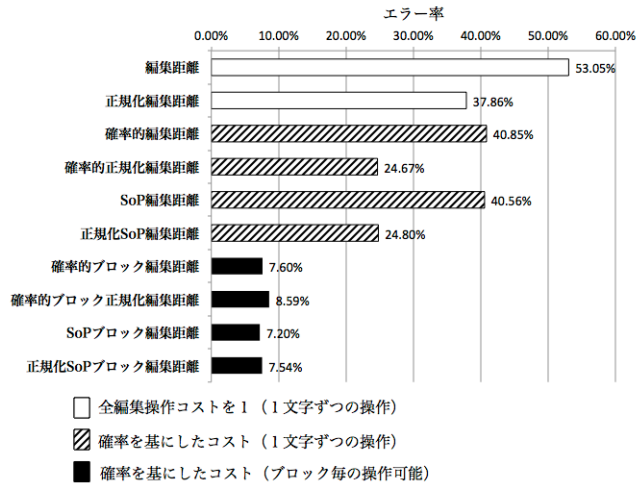

図 2: 実験結果：英語-ローマ字化日本語

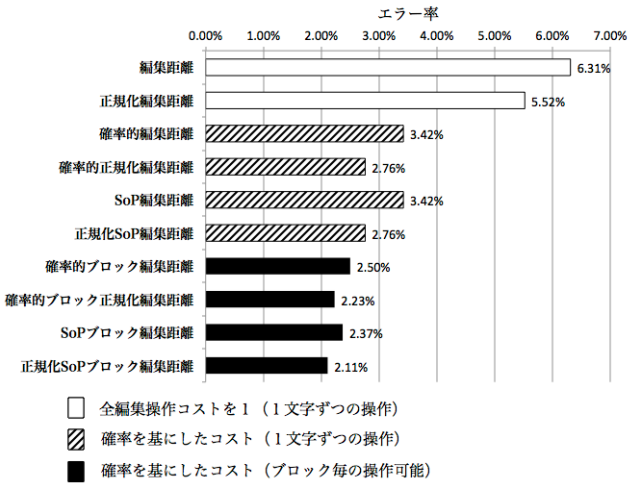

図 3: 実験結果：英語-ローマ字化ロシア語

53.05%、ロシア語の場合では 6.31%と大きな差があることが分かる。その原因として、ペアとなっている言語間で対応するブロックに関係があると考えられる。ペアの言語間に多対多で対応するブロックがより多く存在するほど、基本的な編集距離のエラー率は当然高くなる。英語とローマ字変換した日本語の多くは図 4 のようにブロックのアラインメントを多く持つ。そしてこのような場合こそ、ブロックの概念を編集距離に適用することが重要となり、その結果としてエラー率減少につながる。

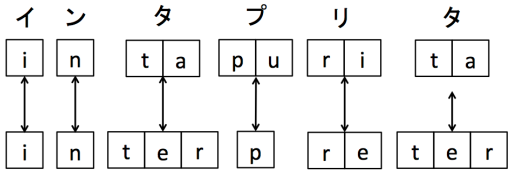

図 4: アラインメント

今後はこの手法の有効性をより一般的に示すために、翻字データだけでなく、単語に発音記号列を割り当てる Grapheme-to-phoneme タスクや、DNA 配列や画像データなどの言語データ以外のデータに対しても、追加の評価実験を行っていく必要があると考える。

5 まとめ

本稿では、文献 [2] のベイジアンアラインメント手法をブロック編集距離の編集操作の確率的なコスト学習へと適用し、それに対する評価実験を翻字データを用いて行った。その結果、2 つのデータセットに対し、既存手法を上回る性能を示した。さらに、ブロック毎の編集操作を確率的に学習する際に直面する過学習問題を克服することもできた。

今後は、この手法のより一般的な有効性を示す為に、異なるデータセットを用いて追加の評価実験を行っていく。

参考文献

[1] Daniel Lopresti and Andrew Tomkins. Block edit models for approximate string matching. *Theoretical Computer Science*, Vol. 181, No. 1, pp. 159–179, 1997.

[2] Takaaki Fukunishi, Andrew Finch, Seiichi Yamamoto, and Eiichiro Sumita. A bayesian alignment approach to transliteration mining. Vol. 12, No. 3, pp. 9:1–9:22, August 2013.

[3] Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. In Douglas H. Fisher, editor, *ICML*, pp. 287–295. Morgan Kaufmann, 1997.

[4] A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 15, No. 9, pp. 926–932, September 1993.

[5] Silvia García-Díez, François Fouss, Masashi Shimbo, and Marco Saerens. A sum-over-paths extension of edit distances accounting for all sequence alignments. *Pattern Recogn.*, Vol. 44, No. 6, pp. 1172–1182, June 2011.

[6] Silvia Garcia Diez, Francois Fouss, Masashi Shimbo, and Marco Saerens. Normalized sum-over-paths edit distances. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pp. 1044–1047, Washington, DC, USA, 2010. IEEE Computer Society.

[7] A Kumaran, Mitesh M. Khapra, and Haizhou Li. Whitepaper of news 2010 shared task on transliteration mining. In *Proceedings of the 2010 Named Entities Workshop*, pp. 29–38, Uppsala, Sweden, July 2010. Association for Computational Linguistics.