部分文字列を用いた綴り誤りに対する訂正候補の高速検索手法

藤澤信夫† 森田和宏† 泓田正雄† 青江順一†
†徳島大学大学院 先端技術科学教育部

概要 本稿では，人が入力した文字列の綴り誤りに対する訂正候補を高速に検索する手法について述べる．提案手法では入力文字列に含まれる n 個の誤りに対して，その文字列を n+1 個に分割することで，分割した部分文字列のいずれかが正しい文字列に含まれることに着目した．予め正しい文字列から部分文字列を検索キーとする訂正辞書を構築し，訂正時には部分文字列の検索に加え，正しい文字列と入力文字列との文字数差や部分文字列の位置を比較することで処理の高速化を図った．1 個以上，文字列長未満の誤りを含む文字列を自動生成した訂正実験では，5 万語の正しい文字列に対し，訂正候補を平均で 0.24ms（Windows XP，Pentium4 3.2GHz）で検索できた．

1. はじめに

近年，インターネットやコンピュータの普及に伴い，コンピュータを利用する機会が増えている．コンピュータの基本操作の 1 つとして，キーボードを用いた文字入力がある．文字入力は電子文書の作成や文字列による情報検索等，様々な場面で用いられる．しかし，人が文字の入力や入力文字列の編集をおこなう際には，綴り誤りをする場合が多く見られる．入力文字列の綴り誤りは人が自ら文字を入力する上で，完全に避けることはできない．

そのため，綴り誤りを自動で訂正する技術の必要性が高まり，綴り誤りの訂正を目的とする研究が古くからおこなわれてきた[1][2]．しかし，それらの多くは英文字に対象を限定した訂正手法や，OCR（光学式文字読み取り装置）で読み取った入力文の読み取り誤りを対象とした訂正手法であり，人が文字の入力やその編集をおこなう際の綴り誤りを対象とした訂正手法ではなかった．

そこで，本手法では綴り誤りに対して，訂正候補を検索する．文字列間の類似度にはレーベンシュタイン距離を用い，正しい文字列の中からレーベンシュタイン距離が小さい文字列を全て訂正候補として出力する．さらに，入力やその文字列の編集といったリアルタイム処理に対する訂正候補の検索であるため，文字列の入力から訂正候補の検索完了までの処理が十分に高速でなければ実用的ではないと考え，高速性を備えた訂正候補の検索手法を提案する．

2. レーベンシュタイン距離

2 つの文字列間の類似度はレーベンシュタイン距離を用いる[3]．レーベンシュタイン距離は，2 つの文字列のいずれかの文字列を，もう一方の文字列に変換するために必要な処理回数を求める．変換には綴り誤りの原因となる置換，挿入，脱落の 3 種類の処理を用いる．各処理には，予めペナルティ値が設定されており，変換を完了させるのに必要なペナルティ値の合計が最終的な距離となる．距離は小さいほど 2 つの文字列が類似していることを表す．本稿では，綴り誤りの訂正候補を検索する際に，置換，挿入，脱落のいずれの誤りも公平な誤り方であると考え，置換，挿入，脱落のペナルティ値を全て 1 として，距離の算出をおこなう．

【ホームラン】変換前の文字列
“ホームベン”（‘ラ’を置換）
“ホームベー”（‘ン’を置換）
“ホームベース”（‘ス’を挿入）
【ホームベース】変換後の文字列

図 1：レーベンシュタイン距離の算出例

例えば，“ホームラン”と“ホームベース”の距離の算出例を図 1 に示す．“ホームラン”と“ホームベース”の距離の算出では，図 1 のように 2 回の置換と 1 回の挿入の処理によって，変換が完了するので距離は 3 となる．

n 文字と m 文字の文字列における距離算出の計算量は O(n×m)となるため，綴り誤りの訂正処理に距離を用いる場合，全ての正しい文字列と誤った文字列との距離を算出したのでは，多くの処理時間が必

要となってしまう．

そこで，本手法では正しい文字列の中から距離の算出対象語を絞り込み，この語に対してのみ，レーベンシュタイン距離を算出することにより，高速な訂正候補の検索を実現する．

3. 部分文字列を用いた訂正候補の検索

3.1. 部分文字列

距離の算出対象語を見つけるために，部分文字列の検索を用いる．本手法での部分文字列とは，正しい文字列や誤った文字列を複数に分割した文字列のことである．部分文字列の作成において，本手法では入力文字列に含まれる n 個の誤りに対して，その文字列を n+1 個に分割することで，分割した部分文字列のいずれかは必ず正しい文字列に含まれることに着目した．例えば，1 個の置換誤りを含む“アル×リズム”が入力文字列である場合の分割例を図 2 に示す．図 2 のように，1 個の誤りを含んだ文字列が入力である場合には，その文字列を 2 個の文字列に分割することで，どちらか一方の文字列には誤りが含まれず, 正しい文字列と共通する文字列となる．本手法では，この分割手法を利用して検索キーとなる部分文字列の作成をおこなう．

しかし，この分割手法をそのまま用いるのでは，3 文字を 2 分割する場合や 5 文字を 3 分割する場合等，均等な文字数に分割できない場合も見られる. また，誤りの位置に応じて，共通する文字列の位置も異なる．そこで，本手法では，予め正しい文字列から誤りの個数に応じて分割する際の文字数を算出し，その文字数毎に分割される可能性のある全ての部分文字列を作成する．ここで，訂正候補として提示する距離は 1 以上, 入力文字列の文字列長未満とすると，部分文字列の文字数 K は, 含まれる誤りの個数を α，正しい文字列の文字数を β とした時に, 以下の式(1)を用いて表すことができる．また，文字数 K は小数点以下切捨ての整数値とする．

$$K = \frac{\beta + 1}{\alpha + 1} \quad (1 \le \alpha < \beta, \beta \ge 1, K \ge 1) \quad \cdots (1)$$

例えば，“ホームラン”（β=5）を正しい文字列とした時の部分文字列の文字数 K は，“ホームラン”が 5 文字の綴りであるため，訂正すべき距離の範囲は 1 から 4 である．したがって，α が 1 から 4 となり，文字数が 1 以上の条件を満たす K は 1，2，3 となる．まず，K が 1 では，1 文字ずつ分割した 5 個の部分文字列が作成できる．同様に K が 2 の場合には 4 個，K が 3 の場合には 3 個の部分文字列が作成できる．このように作成した部分文字列を検索キーとし，各部分文字列と正しい綴りをセットで登録することで訂正辞書を作成する．

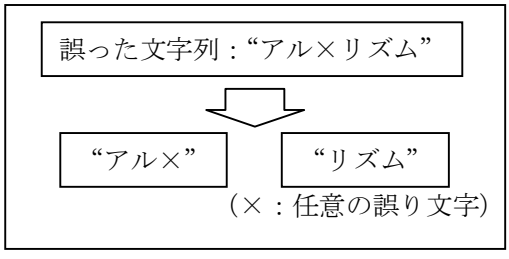

図 2：誤った文字列の分割例

3.2. 距離の算出対象語の絞り込み

部分文字列の検索によって見つかった距離の算出対象語の中には，距離が大きく訂正候補とならないことが明らかな文字列も含まれている．それらの文字列は距離を算出することなく，除外するべきである．そこで，本手法では 2 つの絞り込み処理を用いて，距離の算出対象語を減らすことで，処理の高速化を図る．

3.2.1. 文字数の比較

誤った文字列と正しい文字列の距離が 1 以上となる要因として，2 つの文字列の文字数が違うことがあげられる．文字列の文字数が異なる場合には，挿入，もしくは脱落の誤りが文字数の差分は必ず含まれていることを表す．したがって，距離を α，正しい文字列の文字数を β，誤った文字列の文字数を γ とすると，以下の式（2）の関係が成り立つ．

$$\alpha \ge |\beta - \gamma| \quad \cdots (2)$$

式（2）のように，常に距離は文字数の差以上となるため，文字数の差が大きい場合には，訂正候補とはならない．例えば，“インターフェース”と“ポイント”は共に“イン”という部分文字列を含んでいるが，文字数の差は 4 であるため，必ず距離は 4 以上となる．そこで，本手法では，検索キーである部分文字列とセットで登録する正しい文字列は同じ文

字数毎に保存した．また，検索時には，2 つの文字数の差が小さい正しい文字列から距離を算出し，文字数の差が入力文字列以上である場合には訂正候補とはならないため, 距離の算出対象語から除外した.

3.2.2. 部分文字列の位置比較

誤った文字列と正しい文字列は共に部分文字列を含んでいるが, その部分文字列の位置が異なるほど, 距離も大きくなる. 例えば, “インターネット” と “スタートライン” における部分文字列の比較例を図 3 に示す. 図 3 のように, “インターネット” と “スタートライン” は共に “イン” という部分文字列を含んでいるが，その位置は大きく異なる．そこで，本手法では，部分文字列の位置情報の比較に，部分文字列の前後の文字数を利用する. ここで, 距離を α，正しい文字列における部分文字列の前後の文字数 S1，S2，誤った文字列における部分文字列の前後の文字数を T1，T2 とすると，以下の式（3）の関係が成り立つ.

$$\alpha \geq |S1 - T1| + |S2 - T2| \quad \cdots (3)$$

式（3）のように，常に距離は部分文字列の前後の文字数差以上となるため，前後の文字数が大きい場合には，訂正候補とはならない．そこで，訂正辞書に部分文字列と正しい文字列をセットで登録する際に，部分文字列の位置情報として，部分文字列の前後の文字数も同時に保存した．また，検索時には位置情報の比較をおこない，前後の文字数の差が入力文字数以上であれば，訂正候補とはならないため，距離の算出対象語から除外した.

3.3. 訂正辞書の構築

3.1 節で述べた部分文字列を検索キーとし, 正しい綴りをセットで登録することで訂正辞書を構築する. また, 3.2 節で述べたように同じ部分文字列に複数の正しい綴りが保存されている場合には，同じ文字数毎に保存し, 位置情報も同時に保存する. 例として，部分文字列“イン”における訂正辞書のデータ構造例を図 4 に示す．図 4 のように，1 つの部分文字列から同じ文字数の正しい文字列毎に，また，正しい文字列と共にそれぞれの位置情報を同時に保存することで，訂正辞書を構築する.

正しい綴り	文字数（前，後）
“インターネット”	(0，5)
“スタートラ イン”	(5，0)

（ ：部分文字列）

図 3：部分文字列の位置の比較例

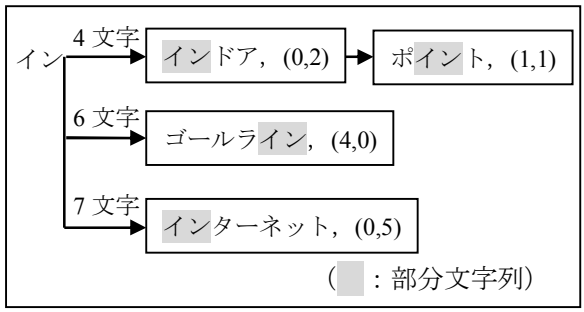

図 4：訂正辞書のデータ構造例

3.4. 訂正候補の検索

誤った文字列が見つかった場合には，誤った文字列を複数の文字列に分割し，訂正辞書中から検索することで，距離の算出対象語を見つける．しかし，含まれる誤りの個数は分からないため，まず，含まれる誤りの個数を 1 個と仮定し，部分文字列を作成する．部分文字列は，訂正辞書の構築時と同様に，その文字数を算出するが，全ての分割文字列を検索に利用する必要はなく，n 個の誤りに対して n+1 個の分割文字列があれば，いずれかは共通するため，n+1 個の部分文字列のみを検索に利用する．ここで，検索時の部分文字列の文字数 K とすると，含まれる誤りの個数を α，誤った文字列の文字数を γ とした時，以下の式（4）を用いて表すことができる．文字数 K は小数点以下切捨ての整数値とする.

$$K = \frac{\gamma}{\alpha + 1} \quad (1 \leq \alpha < \gamma, \gamma \geq 1, K \geq 1) \quad \cdots (4)$$

例えば，“ホームラン” に対して，1 個の置換誤りを含んだ文字列 “ホー×ラン” が入力文字列である場合には, まず, 誤りの個数を 1 個と仮定し, 式 (4) から文字数を算出する．α が 1，γ が 5 であるため，文字数 K は 2 となる．ここで，α は 1 であるため，2 個の部分文字列を検索すれば距離 1 となる正しい文字列は見つかる．そこで，部分文字列は文字列の前方から K 文字間隔に α+1 個の部分文字列を検索

に利用する．ここでは，“ホー”，“×ラ”の 2 個となる．これらの部分文字列を訂正辞書中から検索すれば，置換誤りを含まない文字列の検索により，“ホームラン”を見つけることができる．その後，レーベンシュタイン距離を算出し，得られた距離と分割の際に仮定した誤りの個数が一致しなければ，仮定した誤りの個数が間違っていると考え，仮定する誤りの個数を加算し，同様の処理を繰り返す．得られた距離と仮定した誤りの個数が一致すれば，最後に，距離の小さい順にソートとして出力する．

4. 実験

4.1. 実験設定

提案手法の有効性を示すため, 実験をおこなった．実験では，部分文字列の検索と距離を算出する対象語の絞り込みにより，訂正に必要な処理時間がどの程度減少するかについて調査した．その際，同時に訂正辞書に登録する正しい綴りの語数を 5 万語，3 万語，1 万語と変化させ，正しい綴りの語数に依存して，提案手法の有効性が変化するかについても調査した．実験で利用する誤った文字列は，訂正辞書に登録した正しい文字列の中から無作為に 1000 語（平均文字数は3.17 文字）を選び, それぞれに対し，予め置換, 挿入, 脱落のいずれかの誤りを 1 個以上，文字列長未満付加し，使用した．

本実験では，以下の 3 つの手法を用いた場合における入力文字列 1 語あたりの平均処理時間を比較した．手法 1 は，総当たり方式で全ての正しい文字列との距離を算出し，訂正候補を提示する．手法 2 では，部分文字列の検索をおこない，得られた訂正候補全てに対する距離を算出し, 訂正候補を提示する．手法 3 では，手法 2 に加えて，3.2 節の処理をおこない，得られた訂正候補全てに対する距離を算出し，訂正候補を提示する．

4.2. 評価

3 つの手法を用いた場合の実験結果を図 5 に示す．図 5 の結果から，全ての正しい文字列との距離を算出する場合に比べ，部分文字列の検索処理や絞込み処理をおこなうことで，無駄な距離の算出が少なくなり，大きく処理時間が減少した．手法 3 を用いた場合に必要な平均処理時間は，正しい綴り 1 万語に対し 0.04[ms]，3 万語に対し 0.14[ms]，5 万語に対し 0.24[ms]であった．また，訂正辞書の登録語数の増加に対する処理時間の変化を見ても，手法 1 と比べて，手法 3 の方が処理時間の増加が小さく，高速に訂正候補を検索することができた．

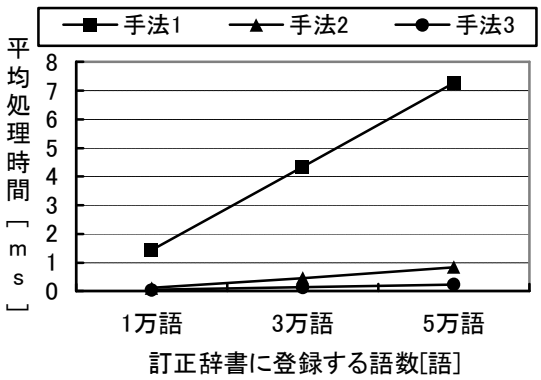

図 5：辞書登録語数と平均処理時間の関係図

5. まとめ

本稿では，部分文字列を用いて，誤った綴りの訂正候補を高速に検索する手法について述べた．本手法では，誤った綴りと正しい綴りに共通して含まれる文字列を部分文字列とし，部分文字列を検索することで距離の算出対象語を見つけた．また，文字数や部分文字列の位置といった情報を用いて，距離の算出対象語を絞り込むことで，より高速に訂正候補を検索することができた．

今後は，距離の算出対象語とその距離を比較し，どのようにすれば，さらに不必要な距離の算出対象語を減らすことができるかについて調査し，さらに高速な訂正候補の検索手法を目指したい．

参考文献

[1]高城泰宏，田中榮一：“構成ハッシュ法に基づく綴り誤りの高速訂正法”，電子情報通信学会，Vol.J80-D2 pp.579-588 1997

[2]永田昌明：“文字類似度と統計的言語モデルを用いた日本語文字認識誤り訂正法”, 電子情報通信学会，Vol.J81-D2 pp.2624-2634 1997

[3]Okuda T., Tanaka E., and Kasai T.: “A method for the Correction of garbled words based on the Levenshtein metric”，IEEE Trans. Comput.，C-25 2,pp.172-178 1976