部分文字列の組み合わせ索引を用いた類似文字列検索

吉岡章太郎　　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

1. はじめに

検索キーワードと検索対象の類似性を考慮した検索は，様々な分野において有用な技術である．例えば，電子辞書などのデータベースにおいてキーワード検索をする際，ユーザがキーワード入力を誤っても，キーワード間の類似性を考慮することで，正しい対象を検索できる．また，用例に基づく機械翻訳や，翻訳支援における類似用例検索，スペルの誤り訂正など，自然言語処理の分野において用途は広い．以上の理由から，類似性を許す検索について，多くの研究がなされている[1]．

本研究もそのような研究の 1 つであり，本稿では，類似性のある文字列を類似文字列と定義し，類似文字列検索の高速化について論じる．類似文字列検索には，大きく分けて全文類似検索と辞書類似検索がある．本研究では，後者の辞書類似検索について取り上げる．藤沢[2]による先行研究では，文字列を分割し，得られた部分文字列を索引とする手法を提案している．しかし，この手法では，1 つの部分文字列のみを索引とするため，距離計算の対象を十分に絞り込めない場合があった．そこで，本研究では藤沢の手法を拡張し，複数の部分文字列を組み合わせて索引を作成する手法を提案する．

2. 類似文字列

本研究では 2 つの文字列間の距離尺度として，編集距離を用いる．

- **編集距離**

m, nを文字列とし，文字列aのi文字目からj文字目で構成される部分文字列をa[i~j]とする．また，aの文字数を|a|とする．mとnの編集距離d(m, n)とは，mに対し，以下の操作をおこなってnに等しくするための最小操作回数である．

（挿入）mの中の任意の場所に，任意の 1 文字を挿入する
（削除）mの中の任意の 1 文字を削除する
（置換）mの中の任意の 1 文字を，任意の別の 1 文字に置き換える．

なお，編集距離は距離の公理を満たしている．

編集距離の計算方法として，最も基本的なものは，動的計画法を用いるものである［3］．これは求めたい 2 つの文字列m, nについて，それぞれの prefix 同士の編集距離を順次求めることで，最終的に全体の編集距離を求めるものである．具体的には，以下の計算式で求める．

$$d(m[1\sim i], n[1\sim j]) = \min \begin{cases} d(m[1\sim i], n[1\sim (j-1)]) + 1 \\ d(m[1\sim (i-1)], n[1\sim j]) + 1 \\ d(m[(1\sim i)-1], n[(1\sim j)-1]) + d(m[i], n[j]) \end{cases}$$

ただし，

$$d(m_i, n_i) = 0 \quad (m_i = n_i \text{のとき})$$
$$= 1 \quad (\text{それ以外のとき})$$

この時間計算量は，mとnの prefix の個数の積に比例するので，O(|m||n|) となり，検索対象が多い場合，時間がかかりすぎる．そのため，高速化を目的とした手法がいくつか提案されている．

3. 従来手法

3.1 トライを用いた手法

Oflazer[4]はトライと呼ばれる木構造を用いた類似文字列検索を提案している．これは，深さ優先でトライを探索し，ノードを辿る度に現在位置までの文字列と検索キーワードとの距離を計算し，それがある限度を超えたらその方向の探索を打ち切るという方法である．これは，途中で探索を打ち切ることにより，距離計算回数を削減する手法であるが，限度を超えるまではすべての文字列と計算をおこなうため，計算量は多くなってしまう．あらかじめ計算対象を絞り込む処理を加えることで，さらに高速な類似文字列検索が実現できると考えられる．

3.2 N-gram 索引を用いた手法

小川ら[5]は N-gram 索引を用いた検索を提案している．これは全文検索を取り上げているが，N-gram は辞書類似検索にも応用できる．その場合，計算対象の集合に対し，あらかじめ N-gram 索引を作成しておく．この索引を検索し，計算対象を絞り込み，計算回数を削減する手法である．この手法の特徴として，索引の長さにより，検索精度と検索速度がトレードオフの関係にあるということが挙げられる．uni-gram のような短い索引では，検索漏れが発生しないという利点がある一方，対象を絞り込みにくいため，計算量は多くなってしまう．逆に tri-gram のような長い

Copyright © 2010 The Association for Natural Language Processing.
All Rights Reserved.

索引だと計算量は少ないが，検索漏れが発生しやすい．

3.3 部分文字列を用いた手法

藤沢［2］は，部分文字列を用いた類似文字列検索を提案している．これは，3.2 節で説明した索引を用いた手法であり，従来の N-gram といった固定長の索引ではなく，可変長で，検索したいエラーの条件をもとに，検索漏れが発生しない範囲で，なるべく長い索引を作成するという手法である．具体的には，k 個のエラーを含む文字列を$k+1$個に分割し，分割により得られた部分文字列を索引とする手法である．これにより，なるべく長い索引を作成することができるが，索引の長さは最大でも文字列長 m に対して $m/(k+1)$ であるため，距離計算の対象を十分に絞り込めない場合があった．

4. 提案手法

本研究では，索引を用いて対象を絞り込み，さらに計算途中で限度を超えたら計算を打ち切ることにより，計算量を削減する手法を提案する．本研究では，前章で挙げた N-gram といった固定長の索引ではなく，文字列の長さと検索の条件に応じて，可変長の部分文字列を作成し，得られた複数の部分文字列を組み合わせて索引とする．複数の部分文字列を索引とするので，索引の持つ情報量，つまり索引の長さが長くなる．そのため，距離計算の対象をより絞り込むことができる．

4.1 分割完全一致

ここでは，提案手法の原理となる分割完全一致について説明する．pを検索キーワード（入力），qを検索対象の文字列とする．このとき，p,q間にk文字の違いがある場合，pを$(k+s)$個に分割することで，分割により得られた部分文字列の内s個は，必ずqに含まれる．

例）app＊e と $k=1$ である apple
2 分割（$s=1$）app，＊e
3 分割（$s=2$）ap, p＊, e
4 分割（$s=3$）ap, p，＊, e
※__：一致部分文字列

そこで，本研究では，文字列を$k+s$個に分割し，得られた部分文字列からs個を選ぶ組み合わせで索引を作成する．ここで一般的に，sの値を大きくすると一致する文字数は増えるため，計算対象をより絞り込めると考えられる．なお，$s=1$のときが藤沢の提案した手法と一致する．

4.2 検索辞書

ここでは，検索辞書について説明する．本研究では，検索対象のキー集合から，あらかじめ索引を登録した検索辞書を構築し，この検索辞書を用いて類似文字列を検索する．まず，索引の作成方法について説明する．

類似文字列の特徴として，どこにどのようなエラーが発生するかは分からないという点がある．そのため，検索辞書に登録する索引は，N-gram 方式で，分割される可能性があるすべての部分文字列を作成しておく．文字列qに対する部分文字列の文字数N_dは，qの文字数|q|を用いて以下の式で求める．

$$N_d = \frac{|q| \pm k}{k+s}$$

上式で文字数の最大値N_{d_max}と最小値N_{d_min}を算出し，最大値から最小値までの文字数で部分文字列を作成する．なお文字数N_dは小数点以下切り捨ての整数値とする．例えば，“apple”から $k=1$，$s=2$ で得られる部分文字列の集合は，$N_{d\_max}=2$，$N_{d\_min}=1$ より，{“ap”，“pp”，“pl”，“le”，“a”，“p”，“p”，“l”，“e”} となる．こうして作成した部分文字列の多重集合から文字（部分文字列の開始位置）の重複を許さずs個を選ぶ組み合わせで索引 I を作成する．このとき，組み合わせの順は文字数の多い部分文字列を先にする．つまり，“ap_e”という組み合わせ索引は作成するが，“a_le”という組み合わせ索引は作成しない．

また，検索辞書には索引を検索キー，元の文字列をレコード情報として登録するが，索引毎に以下のデータもレコード情報として登録しておく．

- 元の文字列の文字数
- 索引の位置情報

索引 I の位置情報はそれぞれの部分文字列の前後の文字数とし，形式を以下のように定義する．

$$(I_1, I_2, \ldots, I_{k+s})$$

例として，$k=1$，$s=2$，文字列“apple”を検索辞書に登録する場合を考える．この条件から得られる索引は，“ap_le”，“pp_e”などがある．また，元の文字列の文字数は 5 であり，“apple”における索引“ap_le”の位置情報は“ap”の前に 0 文字，“ap”と“le”の間に 1 文字，“le” の後ろに 0 文字なので，(0,1,0) となる．

こうして作成した検索辞書のデータ構造を図 1 に示す．

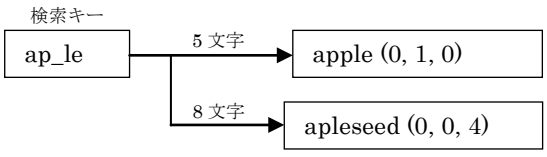

図 1：検索辞書のデータ構造

Copyright © 2010 The Association for Natural Language Processing.
All Rights Reserved.

4.3 検索の流れ

ここでは，実際に検索辞書を用いて類似文字列を検索する流れについて説明する．

- **索引の作成，検索**

まず，検索キーワードに対して索引を作成し，索引を検索キーとして検索辞書を検索する．ここでは索引の作成方法について説明する．まず，検索キーワードpについて，文字数|p|を用いて以下の式で部分文字列の文字数N_kと剰余 oddを求める．

$$N_k = \frac{|\mathrm{p}|}{k+s}$$

$$odd = |\mathrm{p}| \bmod (k+s)$$

なお，N_kはN_dと同じく小数以下を切り捨て整数値とする．この値を基に，pの先頭から$N_k + 1$ 文字の部分文字列をodd個，N_k文字の部分文字列を（$k + s - odd$）個作成する．例えば，検索キーワード“app＊e”を $k=1$，$s=2$ で検索する場合，$N_k = 1$，$odd = 2$ より，得られる部分文字列の集合は，{“ap”，“p＊”，“e”} となる．次にこの部分文字列の集合から s (=2) 個を選ぶ組み合わせで索引を作成する．この場合，“ap_p＊”，“ap_e”，“p＊_e”という索引が作成される．このとき，索引の位置情報も求めておく．

索引検索により絞り込んだ計算対象を，さらに文字数と索引位置により絞り込む．

- **文字数による絞り込み**

索引を含んでいても，文字数が違いすぎると類似文字列としての可能性はなくなるので，検索キーワードpに対して以下の条件式を満たさない文字列qは計算対象から除外する．

$$|\mathrm{q}| - k \le |\mathrm{p}| \le |\mathrm{q}| + k$$

- **索引位置による絞り込み**

索引を含んでいても，索引位置が違いすぎると類似文字列としての可能性はなくなるので，検索キーワードpの索引p_Iに対して以下の条件式を満たさない索引q_Iをもつ文字列qは計算対象から除外する．なお，abs(n)とは，nの絶対値である．

$$\sum_{i=1}^{k+s} \mathrm{abs}(p\_I_i - q\_I_i) \le k$$

- **編集距離の計算**

以上の条件により絞り込まれた計算対象と検索キーワードとの編集距離を求める．なお，トライを用いた従来手法を参考に，計算途中で限度を超えた文字列は，そこで計算を打ち切り，計算量を削減する．距離が指定した限度以内の文字列を検索結果として出力する．

5 実験

これまでに説明した提案手法により類似文字列検索辞書と類似文字列検索システムを実現した．ここでは，提案手法の有効性を確認するためおこなった実験について述べる．

5.1 設定

今回は検索対象として，Google N-gram［6］から英単語 10,000 語を用いた．この英単語を用いて検索辞書を構築し，入力キーワードに類似する検索対象を検索する．また，入力キーワードは，検索対象 10,000 語の中から 5,000 語選び，無作為に挿入，削除，置換といったエラーを 1～3 回発生させたものを各 5,000 語ずつ用意した．なお，この実験では検索の再現率と，絞り込み語実際に計算した文字数を評価する．検索の再現率とは，検索条件満たす類似文字列をどの程度検索できたかという尺度である．また，各従来手法や提案手法では，索引などにより距離計算の回数を削減することが目的であるため，どの程度絞り込みができたかという点を評価する．その際，実際に計算対象となった文字数を比較する．以上の点を提案手法と従来手法で比較することにより，提案手法を評価する．以下に今回用いた従来手法を示す．

（従来手法 1）トライを用いた手法［4］
（従来手法 2）N-gram 索引を用いた手法
（従来手法 3）可変長の索引を用いた手法［2］

以上の設定で実験をおこなった．実験した際の処理環境を以下に示す．

（処理環境）Intel(R) Core 2 Duo CPU 3.33GHz

5.2 結果・考察

再現率の結果を表 1，計算対象となった文字数を表 2 に示す．表 1 より，bi-gram や tri-gram といった索引では，編集距離が大きいほど検索漏れが発生していることがわかる．これは，N-gram 方式で文字列をずらしながら分割しているので，文字列に対し，文字数 N 以内の間隔でエ

表 1：再現率［%］

	トライ	uni-gram	bi-gram	tri-gram	提案手法
$k=1$	100	100	100	100	100
$k=2$	100	100	100	72.9	100
$k=3$	100	100	87.9	40.3	100

Copyright © 2010 The Association for Natural Language Processing.
All Rights Reserved.

表 2：計算対象となった平均文字数［文字］

	トライ	uni-gram	bi-gram	tri-gram	$s=1$	$s=2$	$s=3$
$k=1$	1,066	38,420	5,536	1,149	1,737	290	20
$k=2$	3,930	70,085	7,577	1,293	2,022	562	369
$k=3$	8,993	98,630	8,295	1,274	7,423	4,123	2,339

ラーが発生する場合，共通の索引を作成できないためである．一方，提案手法では，編集距離を条件として可変長の索引を作成しているので，検索漏れは一切発生していない．また従来手法 1 も，計算中の編集距離が限度以内だとすべて計算対象とするため，検索漏れは発生していない．従来手法 3 は提案手法における $s=1$ の場合であるため，これも検索漏れは発生しない．

また表 2 より，各編集距離において提案手法が従来手法を上回る結果が確認できる．編集距離 1 においては，先行研究である従来手法 3 に比べ，計算対象を最大で 90%以上削減できている．従来手法 1 は，検索漏れが発生しないよう，編集距離が限度以内だとすべて計算対象として計算をおこなうため，計算回数が多くなっている．また，従来手法 2 では，uni-gram のように短い索引だと，対象を十分に絞り込むことができないため，膨大な計算量となっている．一方，tri-gram のような長い索引だと，計算量はかなり削減され，提案手法を上回っているように見えるが，表 1 より検索漏れが発生しているため，単純に提案手法よりも有効とはいえない．また，提案手法に注目すると，sの値を大きくすることによって，計算量は削減されていくことが確認できる．つまり，従来手法 3 を上回る結果が得られている．以上のことから，提案手法の有効性が確認できたといえる．

また，図 2 に提案手法における編集距離別の平均検索時間を示す．提案手法において，sの値を大きくすることにより，高速化できている．しかし，文字列によってはsの値を大きくすると，個々の索引長は長くなるが，1 つの文字列に対する索引の数が多くなりすぎ，索引を含む計算対象が多くなる場合がある．そのため，各文字列長に応じて最適なsの値が存在する可能性がある．

6　おわりに

本稿では，複数の部分文字列を組み合わせた索引を用いた類似文字列検索の高速化について説明した．また，実験結果により提案手法の有効性示した．今回はあらかじめ分割数を指定するという固定方式をとったが，今後は文字列長や検索したい編集距離を条件として，自動で分割数を設定し，より高速な類似文字列検索を目指したい．

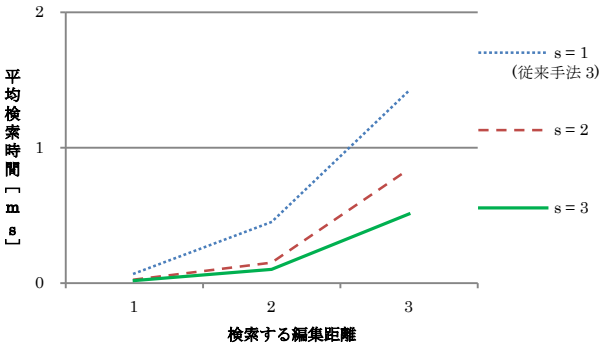

図 2：提案手法の検索時間

参考文献

［1］ 山下達雄，松本祐治：“Suffix Array を用いたフルテキスト類似用例検索”情報処理学会研究報告 97-NL-121, pp. 83-91,1997

［2］ 藤沢信夫：“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” 言語処理学会第 13 回年次大会, pp.962-965, 2007

［3］ D.gusfield ：“Algorithms on Strings, Tree and Sequences: Computer Science and Computati--nal Biology”Cambridge University Press,1997

［4］ Kemal Oflazer： “Error-tolerant Finite-state Recognition with Application to Morphological Analysis and Spelling Correction”, Computational Linguistic, Vol.22, No1, pp.73-89, 1996

［5］ 小川 泰嗣，松田 透：“n-gram 索引を用いた効率的な文書検索法”，電子情報通信学会論文誌，Vol.J82-D-I，No.1，pp.121-129, 1999

［6］ 工藤拓，賀沢秀人著：Web 日本語 N グラム第 1 版，言語資源協会発行

Copyright © 2010 The Association for Natural Language Processing.
All Rights Reserved.