SSJoin を用いた曖昧な駅名記述の高速クリーニング手法

相良　毅[†]　　喜連川　優[†]

[†]東京大学 生産技術研究所

1. はじめに

現在インターネット上ではさまざまな検索サービスが提供されている．特に商用サービスでは，検索精度の向上や，検索時に付加価値の高い情報を提供するため，検索履歴の解析（ログマイニング，log mining）が重要である．

われわれの研究では，商用検索サービスの一つである鉄道の乗換案内の利用履歴を対象としている．乗換案内サービスでは，携帯電話や PC 上のウェブブラウザを用いて，利用者が乗車駅と降車駅を指定し，最適な乗換駅や経路を検索することができる．利用履歴には，利用者 ID，乗車駅，降車駅，検索時刻などの情報が含まれるが，駅名は必ずしも正確ではなく，特に携帯電話では文字入力がやや難しいことから，省略や誤字などが含まれている．履歴の解析を行うには，最初にこのような誤りを含む検索文字列と正しい駅名表記を類似度によって結合するクリーニング処理が必要となる（図 1）．

２つのテーブルに含まれるキーの集合をキー間の類似度によって結合する処理は Similarity Join と呼ばれるが，特にキーが文字列で表されている場合には SSJoin（Set Similarity Join）が有効な手法である [1]．本稿では，駅名文字列を対象とした SSJoin を実現するための文字列類似度検索手法について報告する．

2. 提案手法

2.1. SSJoin の概要

SSJoin は，２つのテーブルに含まれるキー文字列を長さ q の部分文字列の集合（*q-gram*）に分解し，キー 1 から得られる集合とキー２から得られる集合がどの程度重複しているかによって類似度を決定する（Set Overlap Similarity）．たとえば $q = 3$ とした場合，“Microsoft” と “Macrosoft” の類似度は，{‘Mic’, ‘icr’, ‘cro’, ‘ros’, ‘oso’, ‘sof’, ‘oft’} と {‘Mac’, ‘acr’, ‘cro’, ‘ros’, ‘oso’, ‘sof’, ‘oft’} の重複を数え，4 / 6 = 0.66 となる．ただし，*q-gram* ごとに重みをつけることもできる．この類似度が一定のしきい値を超えていれば，２つのキーが等価であると判定し，等価結合と同様に２つのテーブルのレコードを結合する．

利用履歴

UID	時刻	乗降	検索名
1001	0903	0	駒場東大
1001	0903	1	渋谷
1002	0905	0	新宿御苑
1002	0906	0	六本木一
：	：		：
1003	0906	1	幕張メッセ

駅マスタ

駅 ID	駅名
224	駒場東大前
22	渋谷
215	新宿御苑前
201	六本木一丁目
：	：
312	海浜幕張

駅名による類似結合

UID	時刻	乗降	検索名	駅 ID
1001	0903	0	駒場東大	224
1001	0903	1	渋谷	22
1002	0905	0	新宿御苑	215
1002	0906	0	六本木一	201
：	：		：	：
1003	0906	1	幕張メッセ	312

図 1 乗換案内履歴データのクリーニング

2.2. 提案手法の詳細

SSJoin で用いられる Set Overlap Similarity は文字列が短い場合に性能が低下するため，提案手法では日本語独特の特徴である仮名表記と漢字表記の混在や，短い文字列を扱えるように拡張した文字列類似度を用いる [2]．

(1) 駅名マスタの作成

正確な駅名と駅 ID が記載されている駅名マスタテーブルを用意する．ユーザが入力する駅名文字列には読みがなと漢字が混在する場合があるため，一つの駅 ID に対して漢字表記と読みの２通りのキーを与える．

例）「駒場東大前」，ID ＝ 224 の場合

コマバトウダイマエ , 224

駒場東大前 , 224

(2) *q-gram* と駅 ID の対応表の作成

駅名マスタの各キー文字列から *q-gram* を作成し，それぞれの *q-gram* を含む駅 ID のリストと，その *q-gram* の重みを要素とする対応表（ハッシュテーブル）を作成する．重みには，キー文字列から得られた *q-gram* の個数の逆数を用いる．たとえば $q = 2$ の場合，「駒場東大前」からは {“ 駒場 ”, “ 場東 ”, “ 東大 ”, “ 大前 ”} の 4 つの 2-*gram* が得られるため，重みは 1/4 = 0.25 となる．ただし，駅名の長さが q より小さい場合（$q = 3$ の場合の「新宿」など）には，駅名の最後にパディング文字を追加する．

An Efficient Cleaning Method for Japanese Station Names based on SSJoin Aproach
Takeshi Sagara and Masaru Kitsuregawa
Institute of Industrial Science, The University of Tokyo

例）「駒場東大前 (ID:224)」と「駒場車庫前 (ID:521)」から 2-*gram*「駒場」を作成した場合

“ 駒場 ” => {[224, 0.25], [521, 0.25]}

“ コマ ” => {[224, 0.125], [521, 0.143]}

(3) 類似文字列の検索

履歴テーブルから検索文字列を一つ取り出し，*q-gram* の集合を得る．各 *q-gram* に対し，(2) で作成した対応表から対応する駅 ID と重みを検索し，駅 ID ごとの総和を求める．

例）検索文字列が「駒場東大」の場合

“ 駒場 ”=>{[224, 0.25], [521, 0.25]}

“ 場東 ”=>{[224, 0.25], [455, 0.20]}

“ 東大”=>{[224, 0.25], [151, 0.33], [384, 0.33], ...}

ID ごとに集計し，ID:224 が 0.75 で最大となる．

(4) ペナルティの付与

(3) で得られた駅の候補に対し，駅名に含まれない *q-gram* の個数 e_1 と，駅名に含まれるが検索文字列に含まれない *q-gram* の個数 e_2 を数え，ペナルティとして $e_1p_1 + e_2p_2$ を減じる．上の例では「駒場東大前」に含まれるはずの “ 大前 ” が検索語から作成した集合に含まれていないため, p_2 に対するペナルティを 0.1 とした場合，0.1 × 1 を減算して ID:224 の類似度は 0.65 となる．

(5) 結合処理

(4) で求めた値が最大で，かつしきい値を超えた場合に，検索文字列と駅名が等価であると判断し，レコードを結合する（検索履歴に駅 ID を与える）．

3. 実験による評価

ジョルダン株式会社が提供する乗換案内サービスを携帯電話から利用した 1 ヶ月間の検索履歴データを用いて，提案手法の処理速度と精度の評価を行った．

まず，検索語が駅名（読みまたは漢字表記）と完全一致する場合は，結合が容易なため評価の対象から除外した．次に検索語に対する正解を次のように作成した．完全一致する駅名がない場合，ユーザはシステムが示した候補から選択するか，新しい検索語を入力する作業を繰り返す．携帯電話による検索の場合，端末の個体番号に一意に対応する ID がサーバに送られるため，そのユーザが続けてどのような処理を行ったか追跡できる．そこで，最終的に駅名を確定して経路検索に進んだ場合，その駅名を当初の検索語に対する正解とした．

得られた正解付き検索履歴 5,002 件に対し，提案手法による処理を q = 2, 3, 4 と変化させて行った．ただしペナルティは経験的に p_1= 0.3, p_2 = 0.1 とした．

また，他の類似検索手法を比較するため，編集距離による類似度（DP 法，ギャップペナルティ d = 0.5, 1.0）と前方一致（prefix search）による実験も行った．実装は 2.0 GHz Core2 Duo，2GB メモリを搭載した Mac 上で Ruby 1.9.0 による．結果を表 1 に示す．

表 1 提案手法の評価（処理対象 5,002 件，駅名マスタ 9,389 件）

処理手法	正解	不正解	正解率（%）	処理時間（秒）
提案手法 (q=2)	*4,397*	*605*	*87.9*	1.59
提案手法 (q=3)	3,411	1,591	68.2	0.64
提案手法 (q=4)	2,186	2,816	43.7	0.39
編集距離 (d=0.5)	2,981	2,021	59.6	13,165.97
編集距離 (d=1)	2,790	2,212	55.8	12,662.92
前方一致	2,702	2,440	54.0	*0.34*

この結果より，提案手法で q = 2 とした場合の正解率が最も高く，q を大きくするほど低下することが分かる．また，処理時間は前方一致に比べれば 5 倍程度長いものの, 編集距離を用いた場合の 8,000 分の 1 程度となった．

一方，編集距離を用いた場合の正解率が予想より低かったため, 不正解となったケースを調べてみたところ，大半が駅名の先頭部分だけが与えられ，残りが省略されているものであった．たとえば「六本木一」というケースでは, 編集距離では「六本木」の方が「六本木一丁目」よりも類似度が高いため,「六本木」が候補となってしまう．削除操作に要するコスト（p_1 に相当）を減らし，追加操作に要するコスト（p_2 に相当）を増加したところ，正解率は提案手法とほぼ同等となった．

4. おわりに

乗換案内サービスの検索履歴に含まれる駅名文字列を，SSJoin を用いて高速に駅名マスタと結合するための類似文字列検索手法を開発した．また，実履歴データを用いて提案手法を評価し，有効性を確認した．

謝辞　本研究はジョルダン株式会社との共同研究による成果である．貴重な検索履歴データを用いた研究の機会を与えて頂いたことに感謝する．

参考文献

[1] Surajit Chaudhuri, Venkatesh Ganti, Raghav Kaushik, A Primitive Operator for Similarity Joins in Data Cleaning. ICDE 2006, pp. 5 -- 16

[2] 相良 毅，アジアオセアニア地域における生物多様性の現象解決のための世界分類学イニシアティブに関する研究，地球環境研究総合推進費平成 15 年度研究成果報告書，Vol.4/6, pp.381 -- 385