頻出文脈に基づく分野依存入力支援

海野 裕也　　坪井 祐太
日本アイ・ビー・エム株式会社 東京基礎研究所
{yunno, yutat}@jp.ibm.com

1　はじめに

用語や表現の統一など既存文書に倣って文を記述することは，ビジネス・技術文書作成など様々な場面で必要となる．例えば要求工学の分野では要求文書における自然言語記述の曖昧性が注目されている [1]．そのため，利用者の用意した既存文書中での出現頻度に基づいた入力支援システムが求められる．本稿では，入力文字列の頻出文脈を文書集合から動的に抽出し，それを利用者に提示して入力候補とする方法を提案する．この方法は，あらかじめ辞書を用意するのに比べて事前の準備コストが小さい点，また名詞や動詞に限らずあらゆる文字列に対応できる点で優れる．我々が提案してきた文字列検索結果の周辺文脈の圧縮手法 [5] を用いることで，限定された数の入力候補数内で効率よく複数候補を提示できる．特に従来の予測入力手法と比べて，以下の 2 点が大きく異なる．

- 提示する入力候補の組み合わせを探す問題として定式化し，類似候補を抑制しているため，既存手法に比べて多様性に富む候補を提示できる
- 頻度数取得用文書に対して事前に分かち書きなどの言語処理を必要とせず，あらゆる文字列に対して適応可能である

実験では，既存の予測入力手法や文脈圧縮手法との比較を行う．予測入力手法としては，主に直近の入力履歴を参照する方法の他に，既存資源中での頻度を利用する方法 [2, 4, 6] が知られている．既存文書との整合性を取るという本研究の目的を考えると，比較対照は後者となる．

2　予測入力手法

2.1　文脈圧縮アルゴリズム

我々はこれまで，文字列検索結果の周辺文脈を指定された領域に収まる範囲でもっとも効率よく圧縮して表示する手法を提案してきた [5]．この手法は，最大 k 行 l 文字までの範囲に収まる範囲で，検索ヒット位置の周辺文脈を圧縮して表示する．その際，表示した文字列によってカバーされる元の周辺文脈の面積を最大にするように文字列を選択する．これは以下のように定式化される．検索ヒット位置から始まる全後方文脈集合を C とする．文字列 s の長さを $\textsc{Len}(s)$，C 中で s を接頭辞とする要素の数を $\textsc{Pref}(s, C)$ とする．サイズ k の文字列集合 S で，以下の式を最大にする S^* を求める．

$$S^* = \arg\max_S \sum_{s \in S} \textsc{Len}(s) \times \textsc{Pref}(s, C)$$

ただし，類似文字列が大量に出現しないよう，S 中の任意の 2 要素 $s_i, s_j (\in S)$ に関して s_i は s_j の接頭辞とはならないという制約を課す．この制約により，例えば「ボタン」という検索語に対して,「を」「を押」「を押す」のような類似文字列が同時に選択されなくなる．

以上の探索問題は，後方文脈集合 C を TRIE で表現し，動的計画法を用いることで効率的に求められる．C の TRIE 表現の各ノード n に対して，n の最初の子ノードを $\textsc{Chd}(n)$，n の次の兄弟ノードを $\textsc{Sib}(n)$ で表す．ノード n とその子ノード，弟ノード以下から最大 k 個の文字列を選択したときに得られる最大面積は，以下の再帰式によって計算できる．

$$\begin{aligned} f(n,k) \quad = \quad & \max\{s(n) + f(\textsc{Sib}(n), k-1), \\ & \max_{c \in [0,k]} (f(\textsc{Chd}(n), c) + f(\textsc{Sib}(n), k-c))\} \end{aligned}$$

ここで，$s(n)$ はノード n に対応する文字列を選択したときの面積，すなわち n の深さと頻度の積である．TRIE 中のノード数は接尾辞木同様，非分岐ノードを纏めることで，葉ノード数つまり検索ヒット数のオーダー $O(|C|)$ で抑えられる．従って，上記再帰計算は動的計画法を用いて $O(|C|k^2)$ 時間で実行できる．

また，後続文脈の偏りを利用した枝刈りによって，探索時間を削減できる．C の TRIE 表現の各ノード

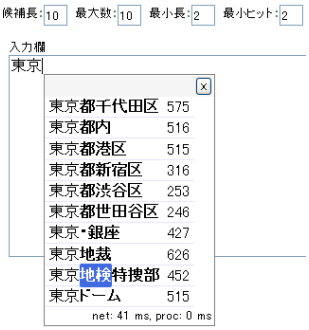

図 1: Web デモ画面.「東京」を入力している.

を頻度順にした，頻度順文脈木を用いると，関数 f の呼び出し中で戻り値の上限を見積もることができる．これが探索途中の最大値を超えないことが分かった時点で探索を打ち切ることができる．また，あらかじめ接尾辞木を出現頻度順に並び替えた頻度順接尾辞木を用意しておくことで，後方文脈集合 C の TRIE 表現の構築時間を省ける．詳細なアルゴリズムに関しては文献 [5] を参照していただくとして省略する．

以上のアルゴリズムを用いることで，実験的には検索ヒット数の対数に比例する程度の時間で計算できる．100 MB 程度の文書に対しても 0.1 秒以内に応答でき，本稿で議論する入力支援に応用しても問題ない程度の反応速度を達成することができる．

2.2 文脈圧縮による予測入力

前節で説明した文脈圧縮アルゴリズムを予測入力に応用する．ここで予測入力とは，現在入力済みの文字列 s に対して，次に入力しそうな文字列を利用者に提示し，これを選択することで入力する方法を指す．すなわち， s に応じて対象文書から圧縮文脈文字列集合 S^* を生成し，これを入力候補とする．入力途中の全文字列 s で検索すると極端にヒット数が少なくなる可能性があるので， s を形態素解析し，最終形態素を検索文字列として利用する．

図 1 に作成した Web アプリのデモ画面の例を示した．利用者は入力欄に途中まで文を入力すると，システムは後続しそうな文字列の一覧を表示する．この中に入力したい候補がある場合，利用者はそれを選択して入力することができる．候補の右側に表示されている数字は文書中での出現頻度である．この情報は，提示された入力候補の妥当性を利用者が判断する際に重要となることが予想される．

表 1: 各手法の特徴の比較.

手法	候補選定	スコア *	複数候補
提案	全部分文字列	LF	総和最大化
Masui[2]	単語単位	F	貪欲
市村 [4]	辞書	$cF + L$	貪欲
奥野 [6]	頻出単語列	$(L - c)F$	貪欲
山本 [3]	後続文字種数	$\log(L + 1)F$	貪欲

* 入力候補のスコアを頻度 F，長さ L，手法固有の定数 c で表す

2.3 既存手法との比較

既存の予測入力手法や文脈圧縮手法との比較を纏める．まず，予測入力手法 3 つと比較する．Masui の POBox [2] は対象文書中での単語の出現頻度を利用して候補を選定している．市村ら [4] は入力候補の確信度と有用度の線形和をスコアに利用している．前者はコーパス中での出現頻度に，後者は提示候補の長さに比例する．奥野ら [6] は入力時間の削減量をモデル化し，その期待値によって候補を提示している．予測入力長 L ，その出現確率 P ，予測失敗と入力のコストの比 β/α を用いて，$(L + \beta/\alpha)P$ をスコアとして用いている．ここで， P の推定値はコーパス中の頻度 F に比例する．さらに，文脈圧縮手法として山本らの KIWI [3] とも比較する．この手法は，提示候補は後続アルファベット数の増える位置を区切り位置の候補とする．また，入力候補のスコアは文字列長の対数と頻度の積で表される．

以上の比較を表 1 にまとめた．ここでは，候補の選定の仕方，各候補のスコア，複数候補の選択方法の 3 点で比較を行っている．本手法が他と最も異なるのは，類似候補を出さないように複数候補の組み合わせを求める点にある．これは，スコアの高い上位 k 件を貪欲に選択する他の手法と大きく異なる．そのため，既存手法では類似候補が複数提示される可能性が高い．

また，本手法は探索アルゴリズムで類似候補の出現を抑えているため，候補選定の際に他の手法と違って辞書の用意や単語区切りを行う必要がない．ただし，候補選択に関しては仮定する資源が手法によって異なるため，単純な比較はできない．

スコアに関してはいずれも入力候補の頻度 F と，入力候補の長さ L からなる関数を用いている．そのため，本手法で用いた組み合わせ探索の枠組みの面積 $s(n)$ の代替とすることも可能であるが，本稿では試していない．

```
// m: 予測入力関数
// d: 入力対象の文
// len(s): s の長さを返す
// substr(s, b, e): s[b] ... s[e-1] を返す
// common_prefix(s, t): s[k] != t[k] なる
//  最小の k を返す
function evaluate(m, d):
  p = 1
  key = 1
  while p < len(d):
    t = substr(d, 0, p)
    d' = substr(d, p, len(d))
    cs = m(t)
    k = 1
    for c in cs:
      k = max(k, common_prefix(c, d'))
    p += k
    key += 1
  return key
```

図 2: 入力シミュレーションによる打鍵数評価の擬似コード.

3 実験

3.1 シミュレーションによる評価方法

入力候補予測の性能を定量的に測るために，入力シミュレーションによる評価を行う．これは，既存文書を入力することを想定して，予測入力候補を与えることでどれだけ打鍵数が減るかを評価する．

入力候補予測器 m は，与えられた事前入力文字列 t に対して，その後方に続きそうな文脈集合 C を出力する．利用者が入力したい候補が C の中にある場合はこれを選択し，ない場合は自ら入力する．インターフェースにもよるが，提示された候補の部分文字列も入力できるものとする．例えば，候補として「東京都」が与えられたとしても，利用者はその途中の「東京」を選択・入力することができるとする．

以上を入力対象の文 $d = \{d_1, \cdots, d_{|d|}\}$ に対してシミュレーションする．各要素 d_i は文字である．まず，利用者は最初の 1 文字 d_1 を入力し， $t = \{d_1\}$ となる．これ以降，t に対する予測器 m の出力 $m(t)$ から選択する．この後ろに入力したい文字列 d' は $d' = \{d_{|t|+1}, \cdots, d_{|d|}\}$ なので，$m(t)$ 中で d' との共通接頭辞が最長のものを入力する．この操作の後，入力された共通接頭辞が t に追加される．$m(t)$ 中に接頭辞の同じものがないときは，自ら $d_{|t|+1}$ を入力する．いずれを行っても入力回数 1 回としてカウントする．以上の操作を繰り返し， $t = d$ となるまで操作を繰り返し，入力回数を求める．図 2 に以上の評価手法を擬似コードで示した．

表 2: コーパス一覧.

略称	説明	サイズ (UTF-8)	文数
san	産経新聞	100 MB	87 万
twi	twitter	53 MB	50 万

3.2 実験設定

表 2 に利用したコーパスの一覧を示す．san は産経新聞一年分，twi は twitter[1] の書き込みを独自にクロールし，日本語の書き込みだけを抽出したデータである．いずれも 10,000 文をテスト用に，残りを参照用に利用する．

比較対照として，既存の文脈圧縮手法として山本らの KIWI [3] ，予測入力手法として奥野らの [6] 頻出 N グラムを提示する手法と比較する．前者は，山本らの手法を使って得られたスコアの高い文脈上位 k 件を，文脈圧縮アルゴリズムの代わりに利用する（kiwi）．後者は，あらかじめ対象文書を形態素解析し， 1 から N グラムまでの頻度を数えておき，クエリを接頭辞として含む形態素列をその長さと頻度でスコア付けする（freq）．なお，奥野らは仮名漢字変換の変換候補ラティスへ組み込んでいるため，本実装とは若干異なる．形態素解析には Sen[2] を用いて，1 から 4 グラムまでの形態素列を候補とした．

最大候補数 $k = 10$ 個，また最大長 $l = 10$ までの候補しか出せないものとする．長さ 1 の候補，コーパス中で頻度 1 の候補は有用性が低いので，いずれの手法でも選択しないものとする．

3.3 実験結果

表 3 に各コーパスと各手法による実験結果を示した．頻度参照と入力対象のコーパスを同一にしたときと別にしたときのそれぞれを試した．まず，いずれの手法でも同一分野のコーパスを利用したときの方が打鍵数削減の効果が大きく，20 % 前後の削減を達成していることがわかる．一方で異なるコーパスでは削減の度合いは 10 % 前後と少ない．これは，分野依存の表現を提示できていることを示唆している．手法間で比較すると，提案手法と freq で差はほとんど現れなかったが，同一分野では僅かに提案手法が勝った．

いくつかのクエリを与えて，提示される候補の例を表 4 に示した．比較しやすいよう提示された候補をア

[1] http://twitter.com/
[2] http://sen.dev.java.net/

表 3: シミュレーションによる入力削減率の比較.

参照	入力	提案	kiwi	freq
san	san	24.7 %	20.9 %	24.4 %
twi	twi	25.0 %	20.9 %	23.4 %
san	twi	8.8 %	7.4 %	8.5 %
twi	san	13.5 %	9.6 %	13.6 %

ルファベット順に並べ替えている．それぞれ右に示した数字は参照コーパス中での出現頻度である．まず,「安全」は後続文脈に大きな偏りがある例である．例えば「安全保障」は 1,300 回以上出現するが,「安全基準」は 60 回程度である．そのため，提案手法以外の方法では「安全保障」から始まる類似候補に偏っている．一方,「東京」は後続文脈の頻度にあまり大きな偏りがない．こうしたクエリの場合，手法間で提示される候補に違いは少ない．最後は単語区切りの難しい例である．twi に対して “(” の後続文字列を探すと顔文字が見つかる．freq ではこれらの単語区切りに失敗したため，類似した候補が多数出現している．

事例で差がある一方，定量的な評価指標では差が現れない．原因のひとつとして，入力候補に偏りがある場合，低頻度の候補は選択されにくいことが挙げられる．例えば「安全基準」を候補に出しても，それが入力される機会は「安全保障」の 5 % 程度である．そのため打鍵数削減にはあまり効果が期待できない．しかし用語の統一を図るために用例を概観するには，多様な入力可能性を示唆できることは有用なはずである．打鍵数は評価尺度のひとつであるが，その他の尺度も必要であろう．

4 おわりに

本稿では文脈圧縮アルゴリズムを利用して，既存文書との整合性を取るための入力支援手法を提案した．まず，入力途中の文字列に対して形態素解析を行い，最終形態素で既存文書に対して検索する．検索結果の後方文脈を圧縮して提示し，入力候補とすることで入力支援を行う方法である．特に既存の予測入力手法と比較すると，提示候補の組み合わせを探索する点，また事前に単語分割を必要としない点で大きく異なる．既存文書を入力することを想定した入力シミュレーションによる評価により， 20 % 前後の入力打鍵数の削減効果があることを確認した．しかし，参照するデータと入力対象のデータを変えると，効果は 10 % 程度まで落ち込んだ．また，手法間で比べると，既存手法と比べて多様性に富む入力候補を提示できることを確認した．これは本手法が候補の組み合わせの最適化を図ることに起因していると考えられる．

表 4: “安全”，“東京”，“(” の後続候補と頻度の比較.

提案		kiwi		freq	
安全	(san)				
・保安院	80	・保安院	80	確保	151
を確保する	46	を確保する	46	管理	104
委員会	41	確保	151	性を	120
確保	151	確保のため	30	対策	136
管理	104	性を	120	保障	1309
基準	61	対策	136	保障の	112
性に	73	保障	1309	保障会議	111
性を	120	保障上の	60	保障問題	94
対策	136	保障上級代表	22	保障理事	101
保障	1309	保障理事会	100	保障理事会	100
東京	(san)				
・銀座	427	ドーム	515	・銀座	427
ドーム	515	ドーム（(	119	ドーム	515
地検特捜部	275	株式市場	164	地検	452
地裁	626	地検特捜部は	100	地裁	626
都港区	515	地裁	626	都港	515
都渋谷区	253	都の石原慎太郎知事	51	都港区	515
都新宿区	316	都新宿区	316	都新宿区	316
都世田谷区	246	都千代田区の	172	都千代田	575
都千代田区	575	都中央区	248	都千代田区	575
都内	516	都文京区	178	都内	516
(	(twi)				
´ ▽ ‘)	809	´ ▽ ‘)	809	*´	3214
*´	3214	´ ▽ ‘) ノ	710	^^	2810
^^	2810	*´∀`*)	573	^o	3117
^-^)	1605	^-^)	1605	^o^	3114
^_^	1534	^o^)	2951	^0	2703
^o^)	2951	^o^) ／	1853	^0^	2698
^0^)	2690	^0^)	2690	´･ω	1990
´･ω･	1961	^0^) ／	1329	>_	2386
>_<)	2306	´･ω･`)	1290	>_<	2384
笑)	5841	>_<)	2306	笑)	5841

* 表示の都合で半角カナは全角で表示している

参考文献

[1] Daniel M. Berry, Erik Kamsties, and Michael M. Krieger. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. Technical report, 2003.

[2] Toshiyuki Masui. An efficient text input method for pen-based computers. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 328–335, 1998.

[3] 山本真人, 田中久美子, 中川裕志. 検索エンジンに基づく多言語用例指南ツール: KIWI. 言語処理学会第 9 回年次大会発表論文集, pp. 15–18, 2003.

[4] 市村由美, 齋藤佳美, 木村和広, 平川秀樹. 入力予測機能を組み込んだ仮名漢字変換システム. 電子情報通信学会論文誌, Vol. 85-D-II, No. 12, pp. 1853–1863, 2002.

[5] 海野裕也, 坪井祐太. 文字列検索結果に対するコンパクトな文脈集合の高速抽出. NLP 若手の会 第 5 回シンポジウム, 発表 7, 2010.

[6] 奥野陽, 萩原将文. インターネットを用いた日本語入力システム. 情報処理学会研究報告 自然言語処理（SIGNL-190）, pp. 1–6, 2009.