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

日本アイ・ビー・エム株式会社
○海野 裕也，坪井 祐太 {yunno,yutat}@jp.ibm.com

背景：大量の文書に対してKWICは無力

- KWIC (KeyWord In Context) は検索結果の前後文脈を概観するのに便利だが，**ヒット数が多いと概観できなくなる**
- 画面内に収まる範囲で，**類似文脈をまとめて表示**して欲しい

全文脈（KWIC）

ボタンが大きくて・・・
ボタンが赤い．・・・
ボタンという表・・・
ボタンに書いてあ・・・
ボタンをクリックしたら・・・
ボタンをクリックして下・・・
ボタンをクリックしよう・・・
ボタンをクリックできな・・・
ボタンをクリックできま・・・
ボタンをクリック．・・・
ボタンを押したら・・
ボタンを押しては・・・
ボタンを押せませ・・・
ボタンを押そうと・・・

3行でまとめた場合

ボタンをクリックし
ボタンをクリックでき
ボタンを押

具体的な応用：入力支援ツール

- 途中まで入力した文字列に後続しやすい単語・フレーズを見つけて，入力候補として提示する

Input Assist Demo

Index: CCAindex 手法: Maximum area(upper bound)

文脈長: 10 表示件数: 10 最小長: 2 最小ヒット: 5 可変長: ☑ 正規化: ☐ 前向き: ☐

入力欄:

データ

入力	右文脈	頻度
データ	ベース	1822
データ	・ソース	510
データ	・リスナー	169
データ	・フィールド	131
データ	・ディレクトリー	83
データ	・スト	75
データ	・オプション」ページ	68
データ	・エレメント	44
データ	をクロール	48
データ	のクロール	68

DB CCAindex : 0.016 ms

デモあります

似てる・・・

面積最大化原理

- 表示する文字列でカバーされる面積を求める
- K個の文字列で**カバーされる面積の合計を最大化**させる
- この問題は動的計画法で，**検索ヒット数に対して線形時間**で計算可能

「ボタン」の後続文字列

全文脈(KWIC)

ボタンが大きくて・・・
ボタンが赤い．・・・
ボタンという表・・・
ボタンに書いてあ・・・
ボタンをクリックしたら・・・
ボタンをクリックして下・・・
ボタンをクリックしよう・・・
ボタンをクリックできな・・・
ボタンをクリックできま・・・
ボタンをクリック．・・・
ボタンを押したら・・
ボタンを押しては・・・
ボタンを押せませ・・・
ボタンを押そうと・・・

「を」のカバー範囲
「を押」のカバー範囲
「を押し」のカバー範囲

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

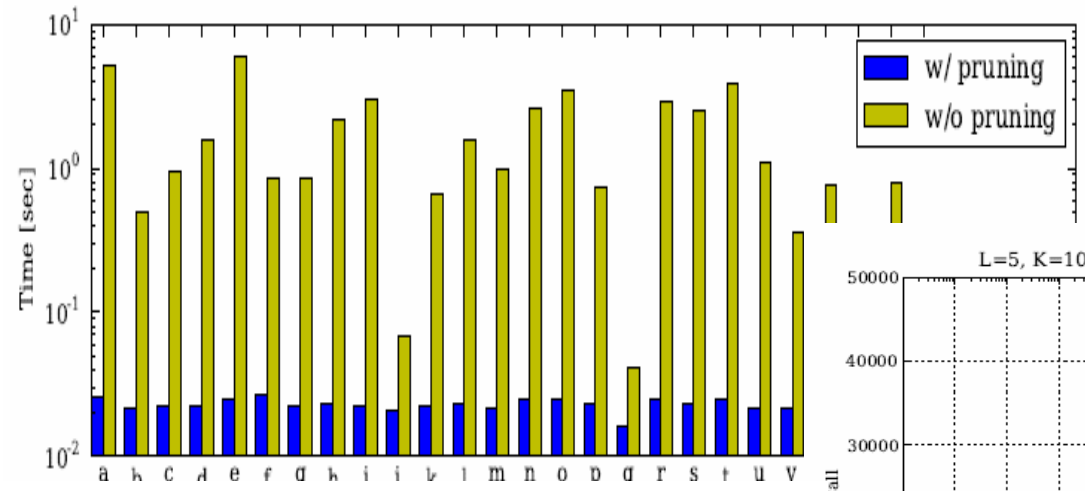
S: 文字列集合, C: 文脈文字列集合
Pref(s, C): sを接尾辞とするC中の要素数

提案手法(K=3)

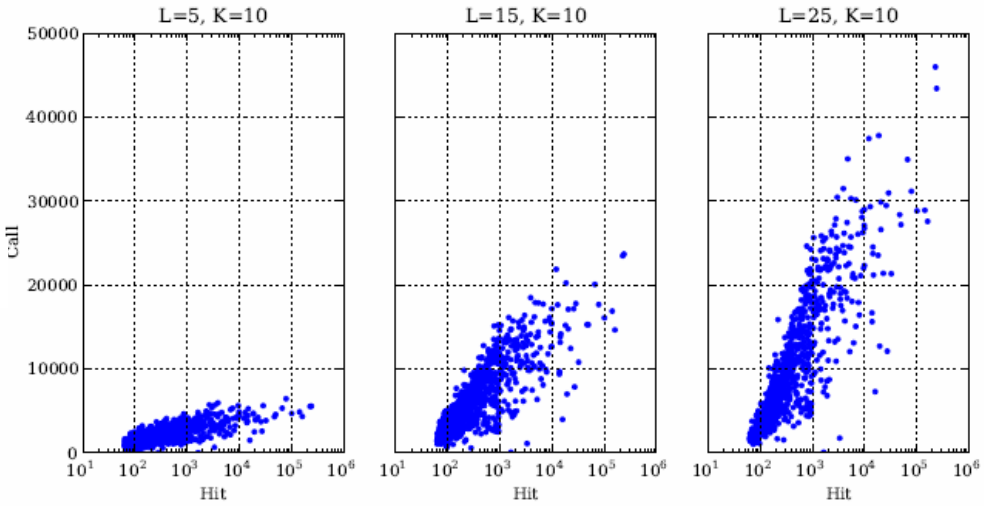
ボタンをクリックし
ボタンをクリックでき
ボタンを押

多項式時間を越えるための枝刈り

- 求めるのは最大面積なので，現在の**最大面積を超えられないなら探索を中断**する
  - 後続文字列の分布に**強い偏り**があるので，効率的に枝刈りができる
- 接尾辞木の子ノードを頻度順に並べた**頻度順接尾辞木**を作っておく
- 実験的には**文字列ヒット数の対数に比例**する程度の実行時間で済む



枝刈りなし（黄）よりあり（青）が100倍近く高速



検索ヒット数の対数（横）に関数呼び出し回数（縦）が比例

Zipfの法則をはじめとして，言語の持つ分布の偏りをうまく利用している？