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**
- 文脈をまとめた形で表示する必要がある

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

- 文字列 s が文脈集合 C をカバーする面積を，Len(s) x Pref(s, C) で定義する
  - Pref(s, C) は文脈集合 C の内，s を接尾辞とするものの数
- 最大 K 個の文字列集合 S の内，**C をカバーする面積が最大**のものを探す
- ただし，「を」「を押」「を押す」など，S 中で他の文字列の接頭辞にはならないとする

「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

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

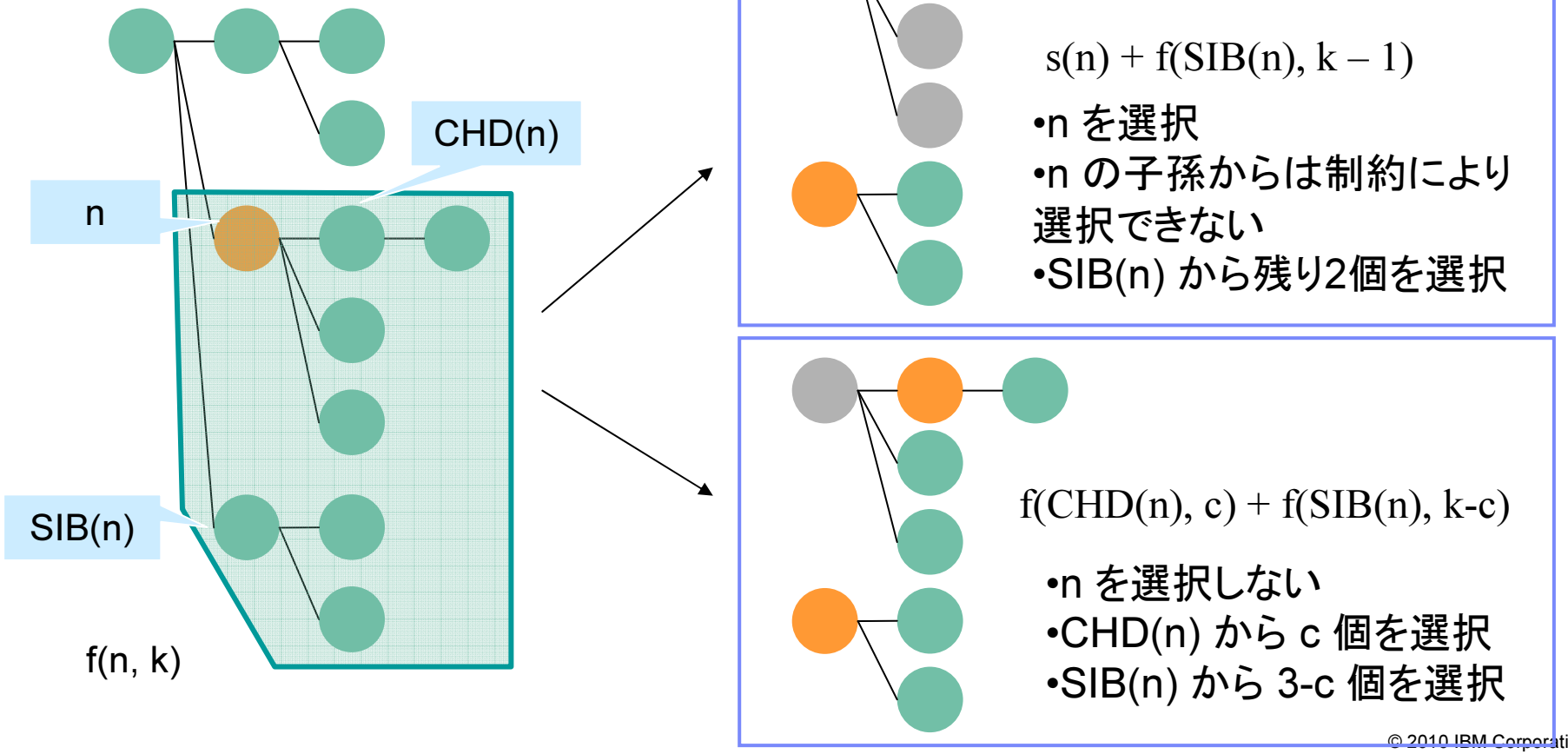
提案手法(K=3)

ボタンをクリックし
ボタンをクリックでき
ボタンを押

文脈木とその上での動的計画法による解法

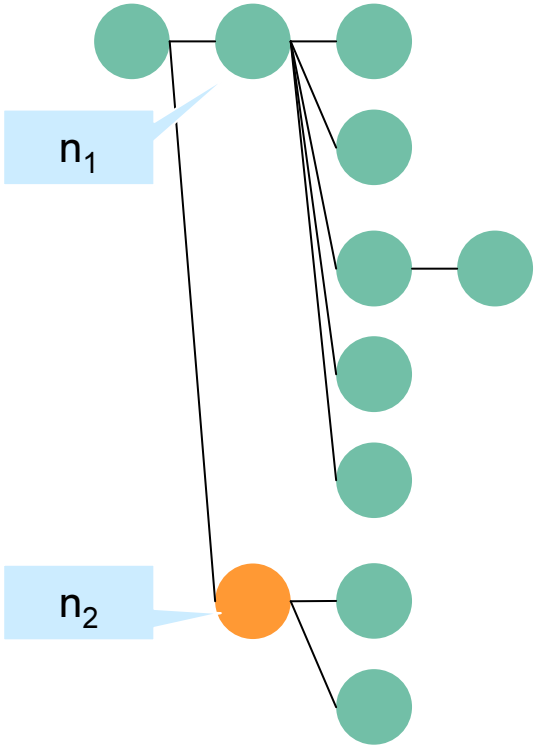
- 全文脈集合をTRIEで表したものを**文脈木**と呼ぶ
- 文脈木は接尾辞木に対して検索を実行すれば直ちに構築できる
- 文脈木のノードに対して面積が定義できるので，面積最大化も文脈木上で計算できる

例）n の子孫と弟の子孫から最大3個選ぶ



上限値関数による枝刈り

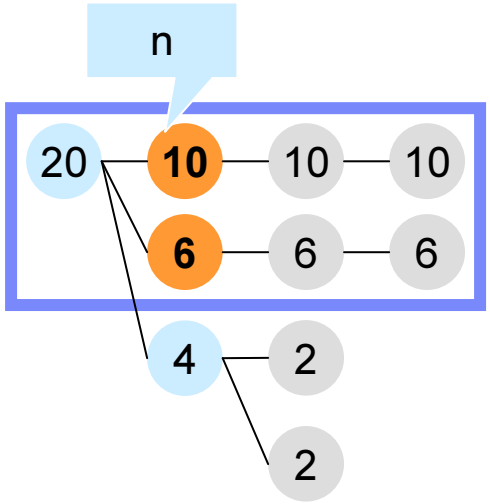
- **計算途中の最大値を越えられない**ことがわかったら，探索しなくて良い
- 関数 f(n, k) に，超えなければならない下限 m を渡した関数 g(n, k, m) を設計する
  - g(n, k, m) は f(n, k) < m なら任意の値を返す
- 上限値関数 u(n, k) は **u(n, k) ≧ f(n, k)** を満たす任意の関数で，これで枝刈りする
  - u(n, k) ≦ m なら，関数 g は直ちに 0 を返す



n_2 が明らかに小さいとき，n_1 の与える最大面積を超えられないことが n_2 を探索しなくてもわかる

上限値関数の実装

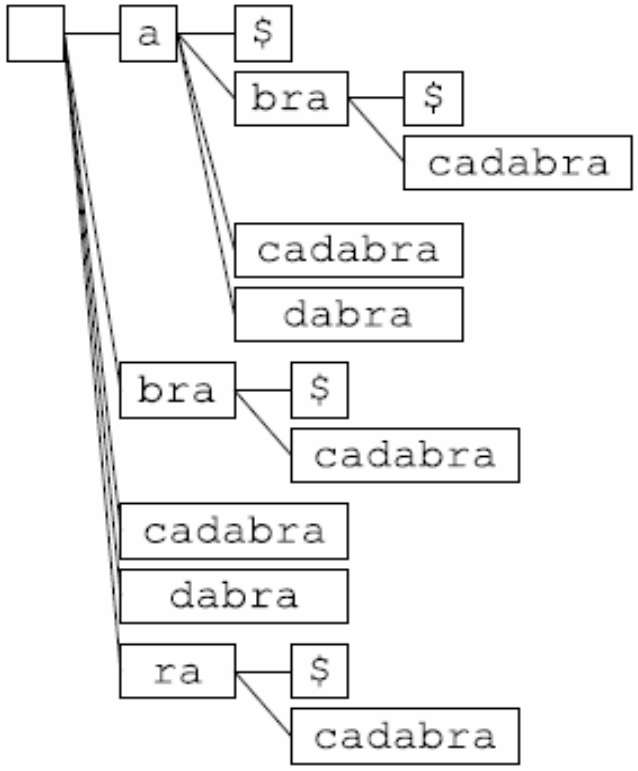
- 以下の2種類の上限値の内，真の f(n, k) に近いほう（小さいほう）を採用する
  - n の兄弟ノードのうち，頻度上位 k 個が分岐がないと仮定したときの面積
  - 過去に計算済みの f(n, k + 1), f(n, k + 2), … の値



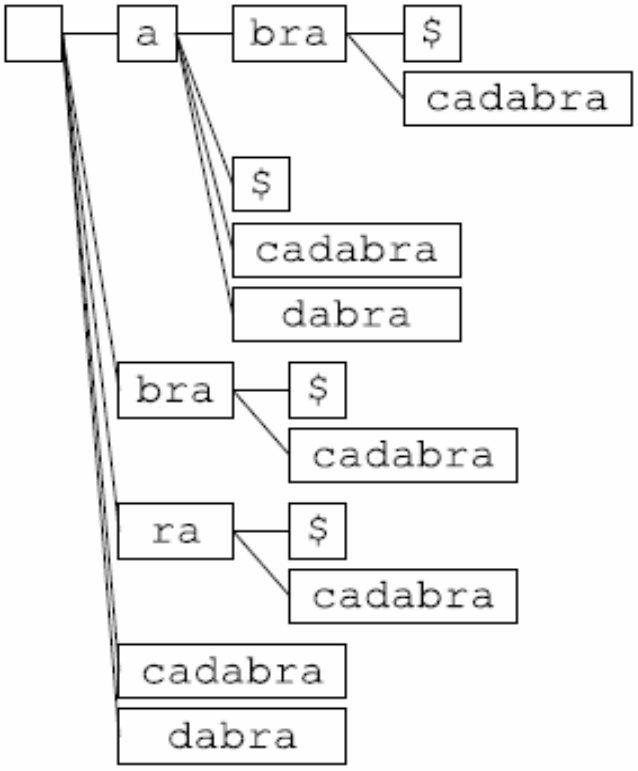
•長さ L = 4 まで，k = 2 個選択する場合を考える
•兄弟ノードが10, 6, 4という情報だけで，再帰計算しなくても青枠の4 x (10 + 6) = 64 を超えることはないことがわかる
•効率的に計算するためには，**子ノードが頻度順**に並んでいると都合が良い

頻度順にソート済みの文脈木を構築するための頻度順接尾辞木

- 文脈木は接尾辞木から直ちに構築できる
- そこで，子ノードを頻度順にソートした**頻度順接尾辞木**を作ることで，頻度順文脈木は直ちに構築できる



（アルファベット順）接尾辞木



頻度順接尾辞木

構築時間に関する実験結果

- 2種類のデータに対して頻度順接尾辞木を構築した
- 実行時間のほとんどが接尾辞木の構築で，**ソートに要した時間は1割以下**

表 2　コーパスごとの頻度順接尾辞木の比較.

データ	サイズ	構築時間	ソート	ノード数
san	257 MB	61.7 sec	4.0 sec	13.6 M
ptb	178 MB	27.5 sec	1.1 sec	11.6 M

ソートの時間はほとんど無視してよい

KWICと面積最大化の結果比較

- 面積最大化の方が**意味のありそうな単位**で切れていることが明白
- KWIC+固定長の場合は重要な頻出フレーズを検出できないことが多い

New York以外の地名もとれる

面積最大化	頻度	KWIC	頻度
New (following)			
New York␣	1222	New York Stock Exc	289
New York,	223	New York Federal R	13
New York.	149	New York banks on	13
New York-based	43	New York Mercantil	12
New England	121	New York, gold for	11
New Jersey	130	New York investmen	10
New Hampshire	79	News & World Repor	10
New Orleans	88	New York trading y	9
New Mexico	44	New York Times Co.	9
Newport	56	Newark to Clevelan	9
day (previous)			
␣day	3819	trading yesterday␣	89
yesterday	977	very within 30 day	26
Saturday	229	the end of the day	19
today	917	s closed yesterday	17
Friday	800	ite trading Friday	16
Tuesday	318	ou have a good day	14
Wednesday	229	or the past 30 day	13
Thursday	213	d with the 300-day	13
Monday	468	it with the 30-day	13
Sunday	285	lion barrels a day	12

dayを入れると曜日が全部取れた

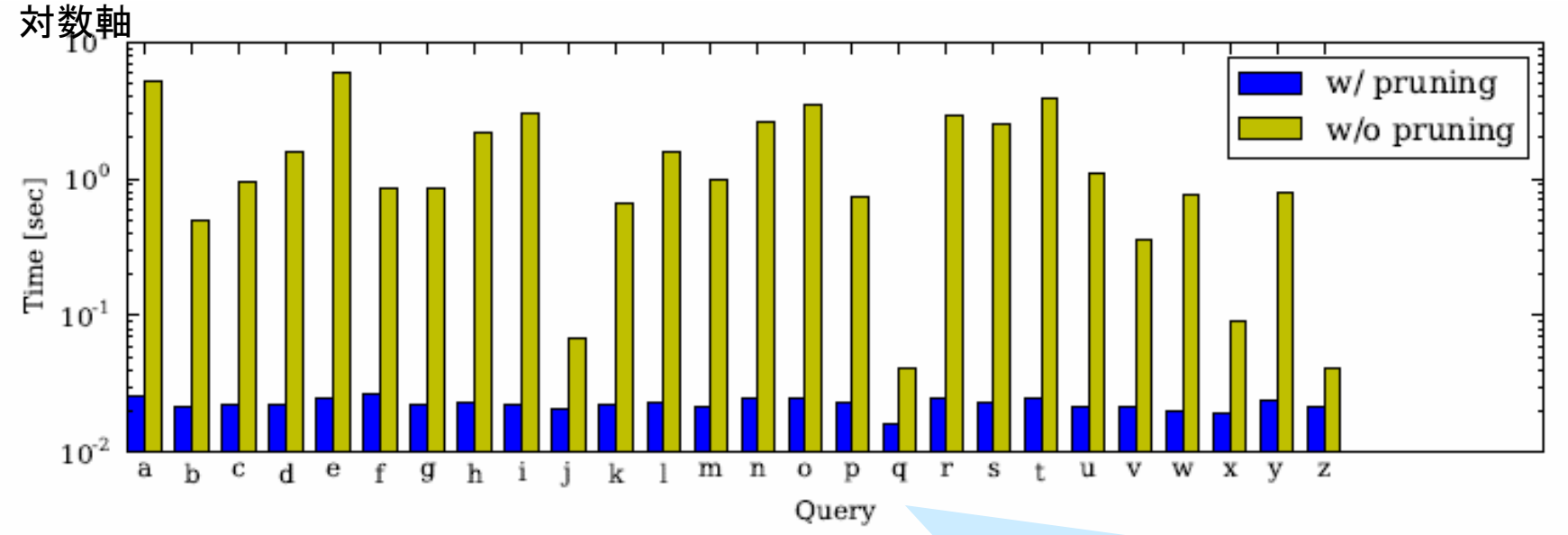
長い複合語も取れる

衆 (following)			
衆院議員␣	1475	衆院有事法制特別委員会	46
衆院議運委員長	148	衆院議員、鈴木宗男被告	34
衆院議院運営委員	119	衆院選でどの党に投票す	32
衆院予算委	429	衆　５５０００　巨␣	28
衆院選	310	衆院選挙区画定審議会が	24
衆院本会議	258	衆院議員（自民党を離党	22
衆院第１委員室	59	衆　４８０００　ダ␣	18
衆院和歌山２区	67	衆院第１委員室を出て、	17
衆院有事法制特別委	62	衆院議運委員長の証人喚	15
衆␣	852	衆院議員の鈴木宗男容疑	12
議会 (previous)			
審議会	724	世界ボクシング評議会	26
協議会	657	救出するための全国協議会	19
世界ボクシング評議会	52	␣都議会	13
パレスチナ評議会	30	教科用図書検定調査審議会	11
国家平和発展評議会	19	生労働省の社会保障審議会	10
県議会	316	衆院選挙区画定審議会	8
、議会	295	府の衆院選挙区画定審議会	8
都議会	186	ーバのカストロ国家評議会	7
市議会	155	フジテレビ番組審議会	7
連邦議会	77	諮問機関「行革断行評議会	7

審議会，評議会以外にもたくさん

上限値による枝刈りの効果

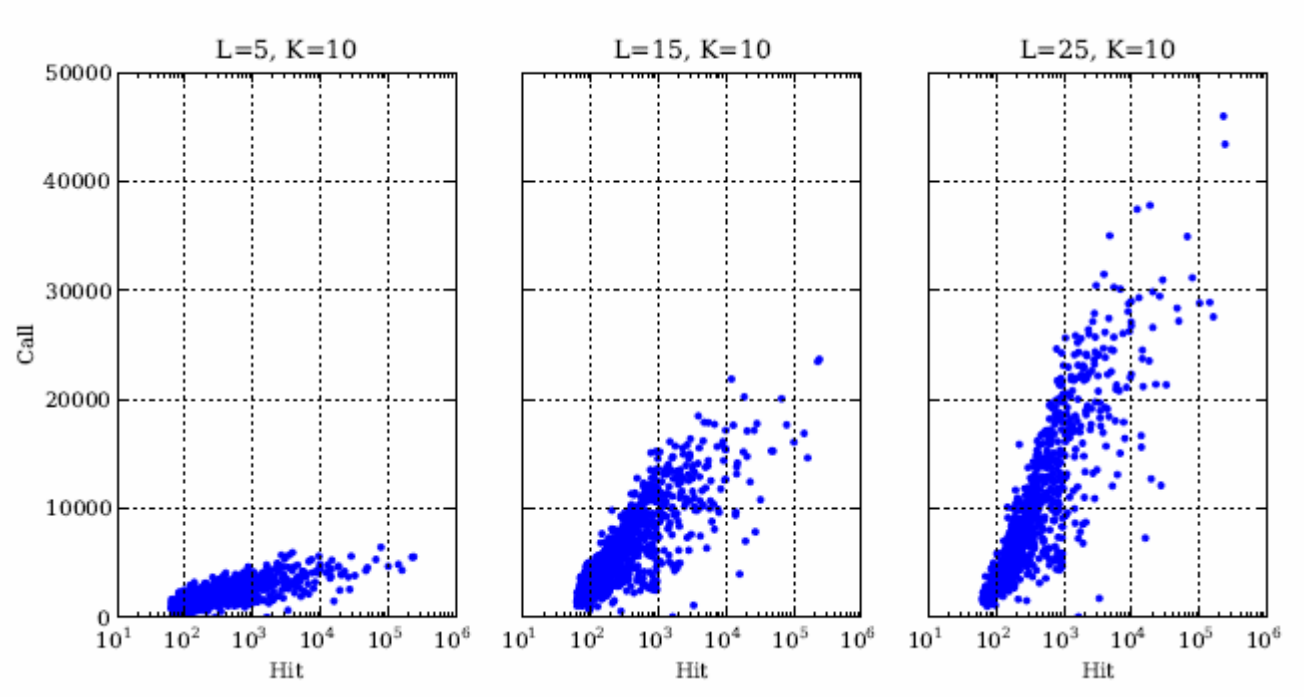
- 26種類のアルファベットをクエリとして与えて実行時間を計測
- 50MB程度のデータでも最大で**100倍速くらい**の実行時間になっている
- ヒット数が少ないと効果も小さいが，その場合はもともと実行時間が短いので問題にならない



実行時間の差が少ないのは，もともと実行時間の短いq, x, zなど

枝刈りアルゴリズムの計算量の実験的な見積もり

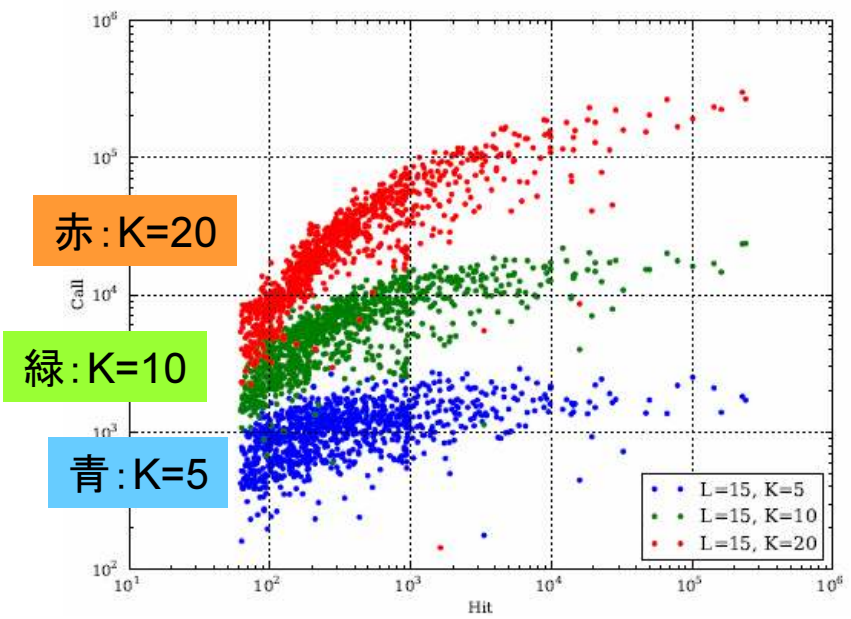
- 横軸ヒット数の対数，縦軸関数呼び出し回数で設定を変えてプロット
- いずれの場合も，おおよそ関数呼び出し回数が**ヒット数の対数に比例**している



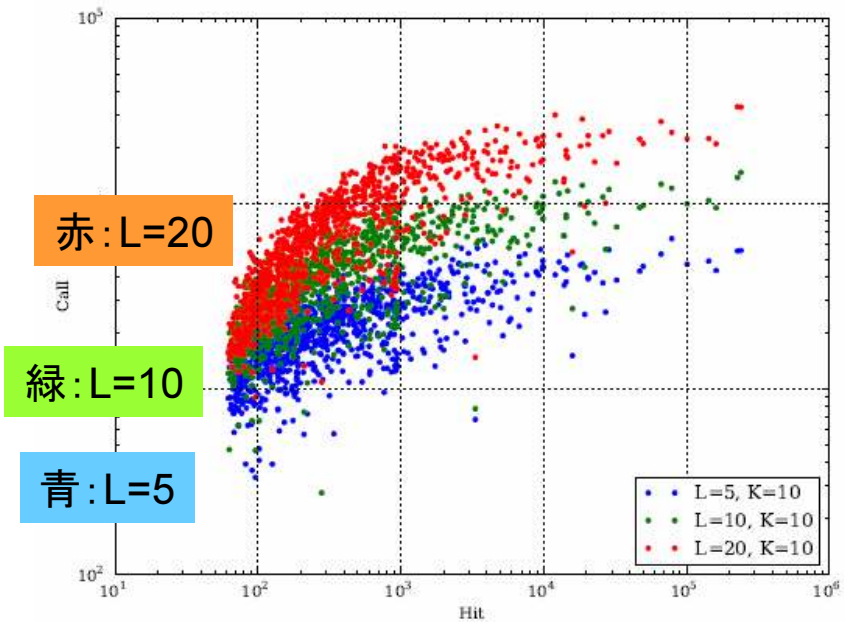
※横軸は検索ヒット数（対数軸），縦軸は関数呼び出し回数

枝刈りアルゴリズムの計算量の実験的な見積もり

- 両対数で，ヒット数と関数呼び出し回数をプロット
- Lを固定してK = {5, 10, 20}（左図）とKを固定してL = {5, 10, 20}で実験
- パラメタを倍にするたびに，**対数グラフ上で同量ずつシフト**している
- KとLに対して指数時間以下（多項式時間？）



L=15, K={5,10,20}



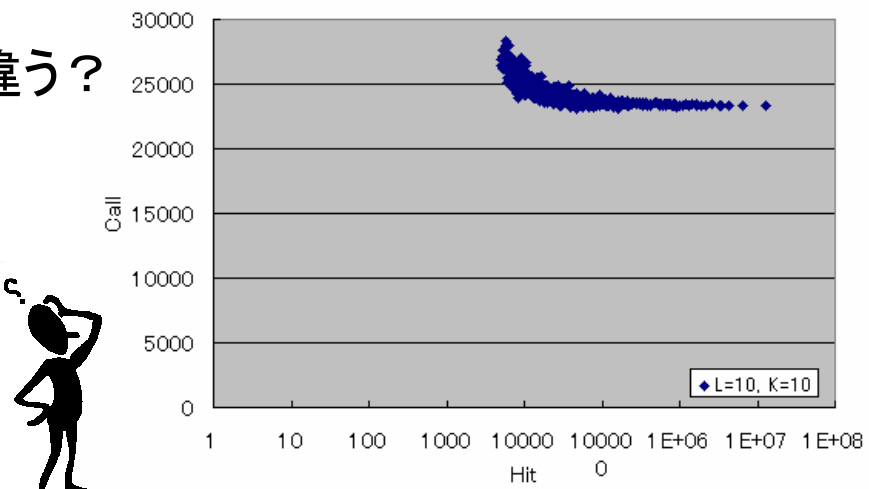
L={5, 10, 20}, K=10

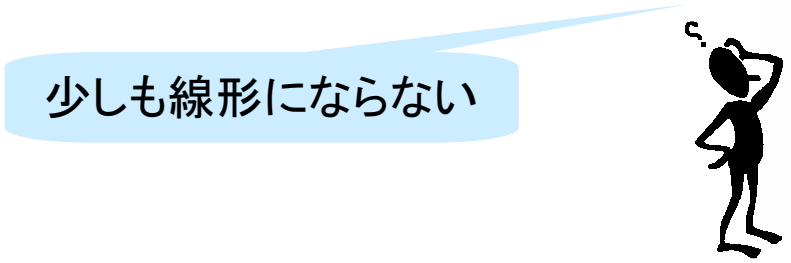
※横軸は検索ヒット数（対数軸），縦軸は関数呼び出し回数（対数軸）

考察：出現する文字の偏りが影響している？

- 一般的に後続文字は偏っていることが多い
  - 「東京」なら「都」や「２」などが極端に多そう
- 後続する文字が偏っているほど枝刈りが効率的に行われる

- 後続文字の分布が **Zipf の法則にしたがうランダムテキスト**を生成
  - 実験的には実行時間が**検索ヒット数に対してほぼ定数**になる
  - 理論的にも定数になった（証明は省略）
  - 実際の分布は Zipf ではないから？
  - 単語分布が Zipf でも文字単位では違う？





関連研究: KIWI との比較

KIWI [山本03]

- **フレーズ単体**にスコア付ける問題
- 頻出する，適当な長さ，後続文字種が多い，が条件
- 複数フレーズを探す方法は自明ではない

本研究

- **複数フレーズの最適な組み合わせ**を探す問題
- 頻出する，文字列長が長い，が条件
- KIWIの指標を面積の代わりに用いることも可能

[山本03] 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. 言語処理学会年次大会 2003.

まとめ

- 文字列検索結果を集約して表示する問題に取り組んだ
- 主に以下の4つの工夫を行った
  1. **面積最大化原理**による最適な文脈表示の定式化をした
  2. 文脈木上での**動的計画法**による多項式時間の解法を提案した
  3. **最大面積の上限を見積もる**ことで枝刈りを可能にした
  4. **頻度順文脈木**を構築することであらかじめ計算に必要なデータを構築した
- 以下のような効果を実験的に検証した
  - 単純なKWICよりも**意味のある単位**で集約された
  - 枝刈りにより，50MB程度のデータでも**100倍以上の速度向上**を果たした
  - **ヒット件数の対数に比例**する程度の実行時間であった
  - 表示する文字列長，文字列候補数に対してそれぞれ**指数時間以下**であった
- 理論的な計算量の検証は今後の課題