高速な類似文字列検索アルゴリズムと自然言語処理への応用

岡崎　直観，辻井　潤一

東京大学
THE UNIVERSITY OF TOKYO

類似文字列検索

- 文字列の集合 V から以下の部分集合を求める

$$Y_{x,\alpha} = \{ y \in V \mid \mathrm{sim}(x, y) \geq \alpha \}$$

  - x: 検索クエリ文字列（V に含まれなくてもよい）
  - y: 検索されて見つかる文字列
  - sim: 類似度関数（コサイン類似度など）
  - α: 類似度の閾値
- 単純には，クエリ x と $|V|$ 回の類似度計算が必要
- これを出来るだけ高速に行いたい

類似文字列検索の応用

- 文字列集合 V を辞書と見なせば
  - 曖昧検索，スペル訂正
    - 入力されたクエリに近い辞書エントリを探す
  - 単語セグメンテーション，固有表現抽出における辞書素性（gazetteer）
    - 人名リストや地名リストと柔軟なマッチングで素性を作る
  - 高速な文字列クラスタリング
    - 類似度の高いペアだけを効率よく見つけてクラスタを形成
- 文字列集合が U と V の２つあるときは
  - データベース統合，重複レコード検出

本研究における前提

- 文字列は文字n-gramの集合で表現する
  - n の長さは任意だが，本発表では $n = 3$ (trigram) とする
  - “methyl sulphone”なら，{‘$$m’, ‘$me’, ‘met’, ‘eth’, ‘thy’, ‘hyl’, ‘yl_’, ‘l_s’, ‘_su’, ‘sul’, ‘ulp’, ‘lph’, ‘pho’, ‘hon’, ‘one’, ‘ne$’, ‘e$$’}
  - 文字列 x の n-gramの数（サイズ）を $|x|$ で表す
    - 文字列 x が q 文字で構成されるならば，$|x| = q + n - 1$
- 文字列の類似度は，n-gramの集合で計算
  - 類似度尺度として，ダイス係数，ジャッカード係数，コサイン類似度，オーバーラップ係数を扱う
    - これらはn-gram検索を行うだけで正確な解が求まる
    - 編集距離にも適用可能だが，正確な解が求まらない

類似文字列検索の必要（十分）条件

尺度	$\min \lvert y \rvert$	$\max \lvert y \rvert$	$\min m(x, y)$
Dice	$\frac{\alpha}{2-\alpha} \lvert x \rvert$	$\frac{2-\alpha}{\alpha} \lvert x \rvert$	$\frac{1}{2}\alpha(\lvert x \rvert + \lvert y \rvert)$
Jaccard	$\alpha \lvert x \rvert$	$\frac{\lvert x \rvert}{\alpha}$	$\frac{\alpha(\lvert x \rvert + \lvert y \rvert)}{1+\alpha}$
Cosine	$\alpha^2 \lvert x \rvert$	$\frac{\lvert x \rvert}{\alpha^2}$	$\alpha\sqrt{\lvert x \rvert \lvert y \rvert}$
Overlap	N/A	N/A	$\alpha \min\{\lvert x \rvert, \lvert y \rvert\}$

- ただし，$m(x, y)$ は文字列xとyのn-gramのうち，一致するものの数を表す

条件式の適用例

- 検索文字列 x = “methyl sulphone” とする
  - 文字tri-gramで類似度を計算するとき，$|x| = 17$
- コサイン類似度の閾値 $\alpha = 0.7$ とする
- 必要条件を使って，探索すべき文字列のサイズの範囲を求める

$$\left\lceil 0.8^2 \times 17 \right\rceil \leq |y| \leq \left\lfloor 17 / 0.8^2 \right\rfloor \Leftrightarrow 9 \leq |y| \leq 34$$

- このサイズの範囲内で，trigramのオーバーラップの必要十分条件を満たすものをさがす
  - 例えば，$|y| = 16$ のとき，$\mathrm{m}(x, y)$ は12以上必要
    - y = “methyl sulfone“なら，$\mathrm{m}(x, y) = 13$ なのでOK

コサイン類似度の場合の導出

- コサイン類似度を $\mathrm{m}(x, y)$ で表す

$$\mathrm{cosine}(x, y) = \frac{1}{\sqrt{|x|}} \frac{1}{\sqrt{|y|}} m(x, y)$$

- コサイン類似度の閾値の条件を適用すると

$$\alpha\sqrt{|x||y|} \le m(x, y) \le \min\{|x|, |y|\}$$ （必要十分条件）

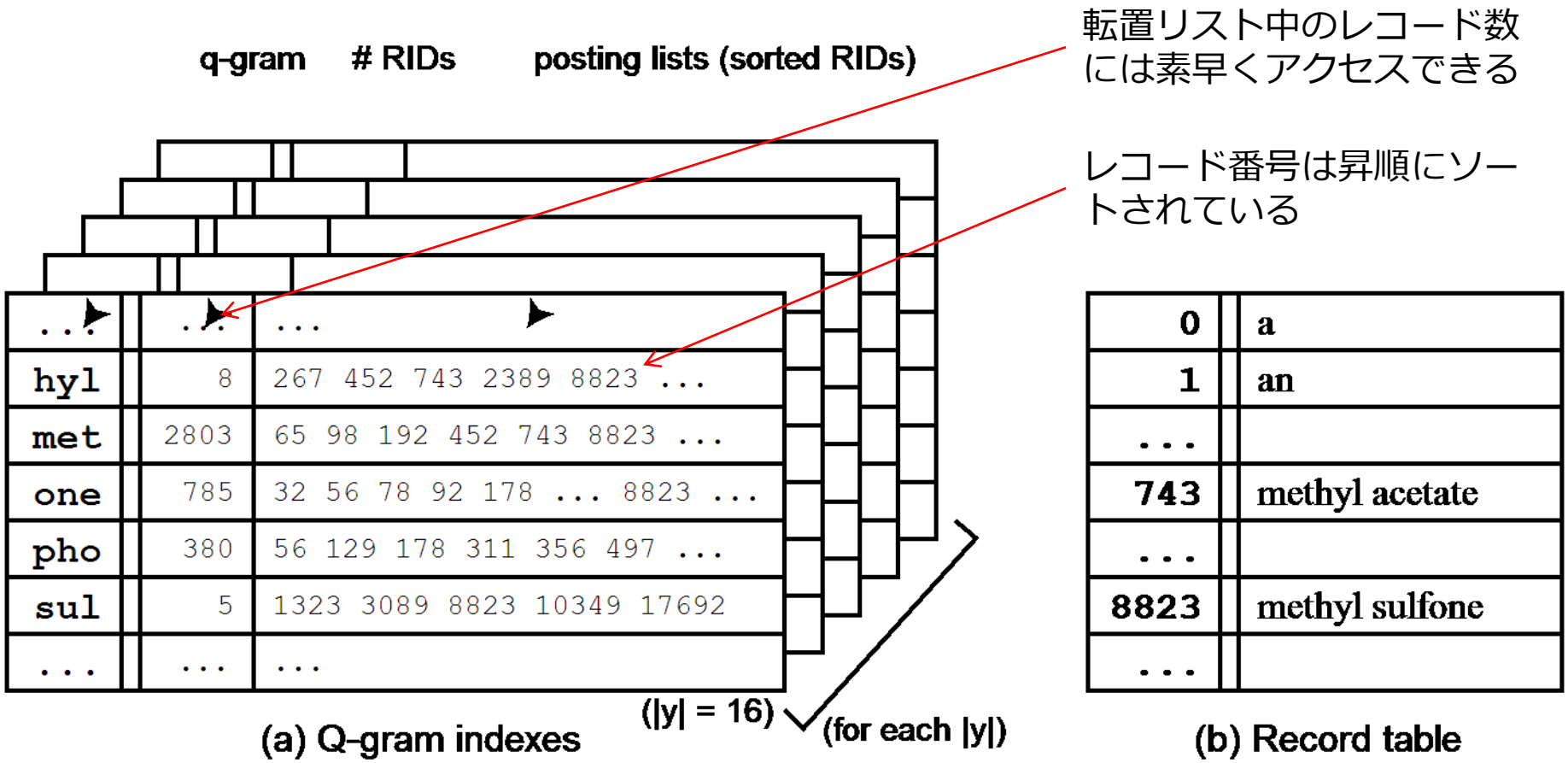
- |y|<=|x|のとき， $\mathrm{m}(x, y)$ を無視すると

$$\alpha^2 |x| \le |y|$$ （必要条件）

- |y|>|x|のとき， $\mathrm{m}(x, y)$ を無視すると

$$|y| \le |x| / \alpha^2$$ （必要条件）

データ構造



- 単純なハッシュデータベースでもSQLでも実装可能

類似文字列検索のアルゴリズム

```
1  X ← string_to_set(x);
2  n ← min_size(|X|, α);
3  N ← max_size(|X|, α);
4  Y ← list();
5  for l ← n to N do
6  |   τ ← min_overlap(|X|, l, α);
7  |   R ← overlap_join(X, D[l], τ);
8  |   foreach r ∈ R do
9  |   |   append r to Y;
10 |   end
11 end
12 return Y;
```

```
1  M ← map();
2  R ← list();
3  foreach q ∈ X do
4  |   foreach i ∈ get(d, q) do
5  |   |   M[i] ← M[i] + 1;
6  |   |   if τ ≤ M[i] then
7  |   |   |   append i to R;
8  |   |   end
9  |   end
10 end
11 return R;
```

xをn-gramで表現 → X

表から探索する文字列のサイズの範囲を計算

最低限必要なオーバーラップの数 τ を計算

τ-オーバーラップ問題：
「集合Xの要素と少なくてもτ個一致する要素を持つアイテムを，データベースからすべて見つけ出す」

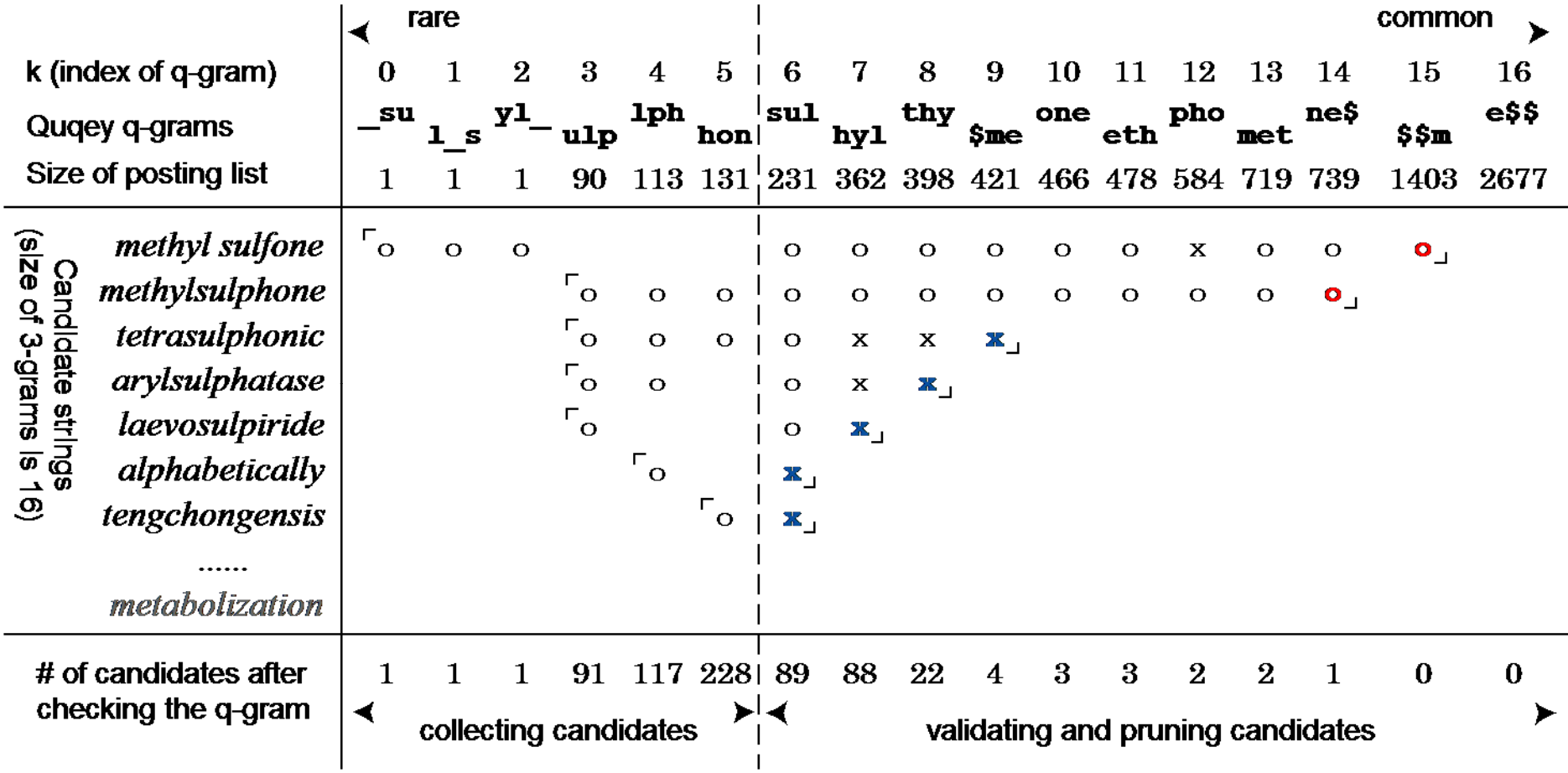
τ-オーバーラップ問題の解を求めるナイーブな実装

データベースdからqを含む文字列IDの転置リストを取り出して，全ての要素をスキャンする

候補の生成と枝刈り

- 候補の生成 [Chaudhuri et al., 06]
  - クエリ文字列のn-gramの数を $|x|$，最低でも必要なオーバーラップの数を τ とする．$(|x| - \tau + 1)$ 個のn-gramで検索したときに，一度も見いだされない文字列IDは，τ 個のオーバーラップを達成できない
    - 最初に $(|x| - \tau + 1)$ 回検索して候補を生成すればよい
- 候補の枝刈り
  - クエリ文字列のn-gramの数を $|x|$とする．k 個のn-gramで検索したときに，オーバーラップの数が θ だった文字列があるとき，その文字列が獲得出来る最大のオーバーラップ数は $(\theta + |x| - k)$
    - この数が τ に満たないときは，枝刈りしてよい

候補の生成と枝刈りの例



- 転置リストを文字列IDのサイズの昇順に並べかえて，候補の数を出来るだけ減らす

効率のよいアルゴリズム (CPMerge)

```
1  sort X by ascending order of get_size(d, X[k]);
2  M ← map();
3  for k ← 0 to (|X| − τ) do
4  |   foreach i ∈ get(d, X[k]) do
5  |   |   M[i] ← M[i] + 1;
6  |   end
7  end
8  R ← list();
9  for k ← (|X| − τ + 1) to (|X| − 1) do
10 |   foreach i ∈ M do
11 |   |   if binary_search(get(d, X[k]), i) then
12 |   |   |   M[i] ← M[i] + 1;
13 |   |   end
14 |   |   if τ ≤ M[i] then
15 |   |   |   append i to R;
16 |   |   |   remove i from M;
17 |   |   else if M[i] + (|X| − k − 1) < τ then
18 |   |   |   remove i from M;
19 |   |   end
20 |   end
21 end
22 return R;
```

候補生成フェーズ

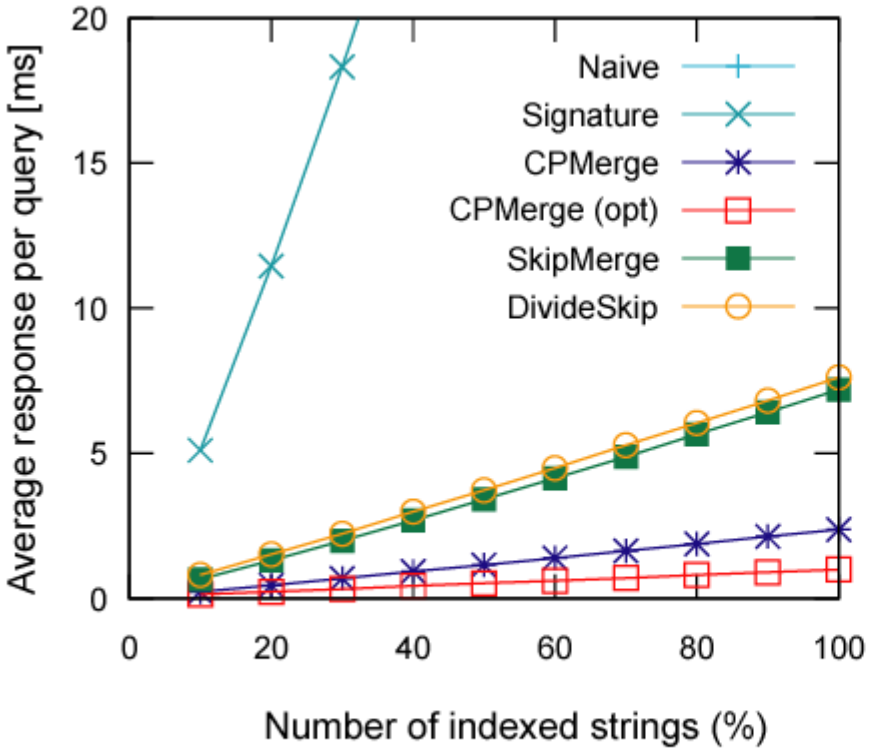
それぞれの候補文字列IDが転置リストに含まれるかどうか，二分探索で調べる

認定処理

枝刈り処理

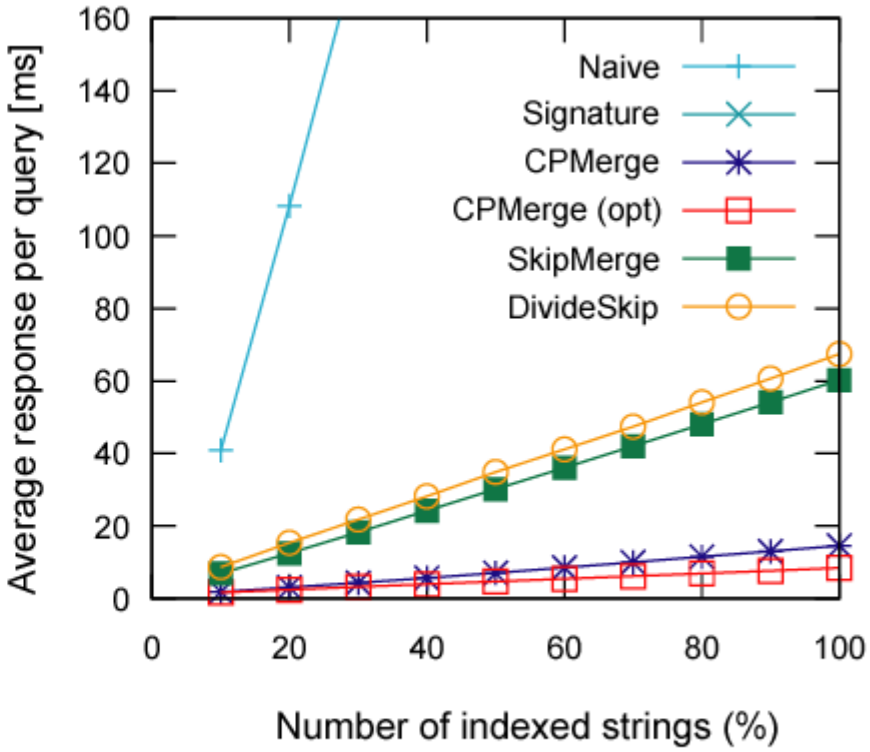
- 転置リスト中のRIDがソートされていることを使うと，map型を使わずに，list型のみで実装可（詳細は省略）

パフォーマンス実験



(a) Google Web1T unigram

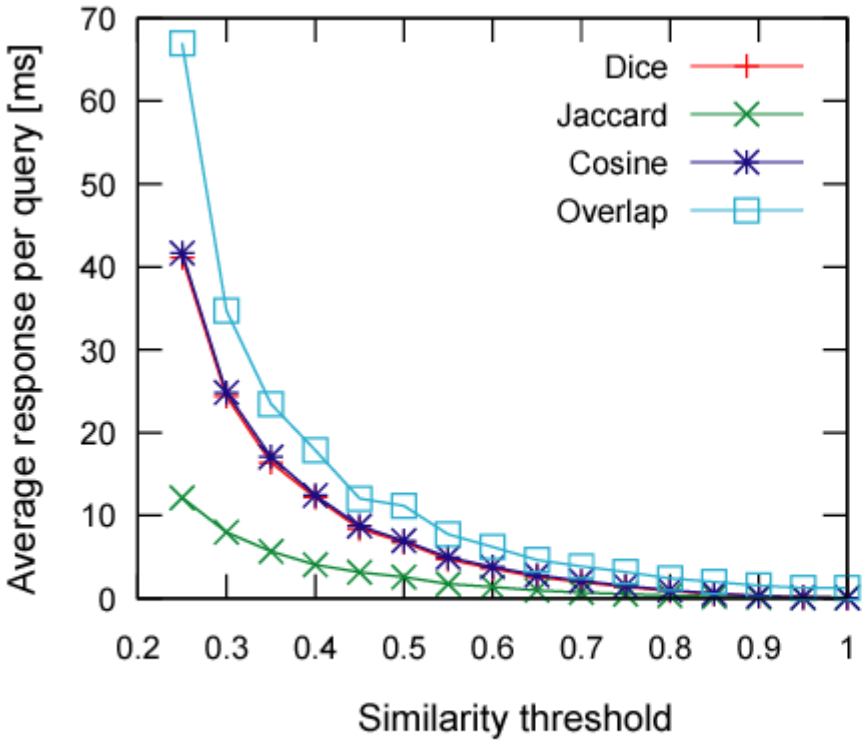
13,588,391文字列 (121MB); tri-gram長平均は10.3; 構築時間557.4秒 (601MB)



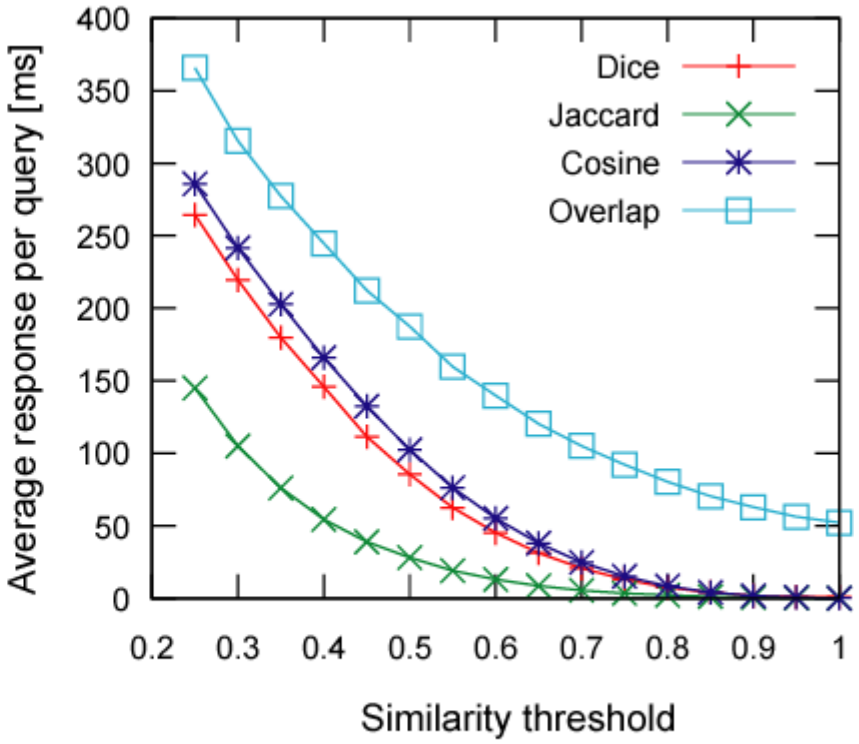
(b) UMLS

5,216,323文字列 (212MB); tri-gram長平均は43.6; 構築時間1215.3秒 (1.1GB)

異なる類似度尺度での比較



(a) Google Web1T unigram



(b) UMLS

- CDB++でハッシュDBを実装．全ての実験環境はDebian GNU/Linux 4.0, Intel Xeon 5140 CPU (2.33GHz) with 4GB main memory

今後の課題

- 重み付き類似度への対応
  - もっと精密な類似度尺度が使えるようになる
  - 単語や用語の分布類似度から似ている語を高速に検索
- 転置リストの圧縮
  - 文字列IDの割り当てを工夫して，転置リストの文字列IDリストを圧縮する
  - うまくいけば，処理速度を速くすることができるかも
- NLPのアプリケーションでの貢献度合いを調べる

固有表現抽出での応用

- 生命医学分野での固有表現抽出
  - BioNLP/NLPBA 2004コーパス
    - 固有表現の種類: protein, DNA, RNA, cell line, cell type
- 学習器
  - Linear-chain CRF (CRFsuiteを使用)
  - 辞書を活用した素性設計 [Sasaki et al., 2008]
- Han-Cheol Choさんと共同で実験

```
Activation      NN      O           O
of              IN      O           O
the             DT      O           O
IL              NN      PROTEIN     B-PROTEIN
2               CD      PROTEIN     I-PROTEIN
precursor       NN      PROTEIN     I-PROTEIN
provides        VVZ     O           O
```

辞書引きを行い，
表現が固有表現辞書中で見つかれば，
辞書素性を発火

類似文字列検索の効果

辞書素性のタイプ	再現率	適合率	F1スコア
完全一致辞書素性	67.42	68.94	68.17
近似文字列一致辞書素性	68.22	70.63	69.40

- 結果
  - 固有表現の認識性能において，若干の改善がみられる
- 今後の課題
  - 複数の単語境界で同じ辞書エントリが類似検索されるとき，どれを素性に加えるべきか（e.g., Semi-CRF？）
  - ラティスを作りながらインクリメンタルに類似文字列検索を行う方法