

# 全文検索エンジン Sennaについて



2006/11/06

@国立情報学研究所

(有)未来検索ブラジル  
末永 匡



# Sennaとは

- オープンソース(LGPL)のライセンスで公開されている、全文検索ライブラリ
  - すべてのソースコードが公開
  - 無償
  - 商用利用可能
- その3つの特徴は...





高 速



# 高 精 度



# 組込型



# Sennaの三大特徴

- **高速**
  - **高精度**
  - **組込型**
- 
- **この3つの特徴について説明**



# 高 速

について



# 転置インデックスによる高速検索

- Sennaでは転置インデックスを採用
  - ある単語が、どの文書にあるかを保持
  - (例) 書籍巻末にある用語索引

単語	文書番号列
国立	1, 4, 10
情報	10, 11, 12
学	10, 14, 16
研究所	2, 4, 10

- 高速な検索が可能



# 完全転置インデックス

- Sennaは完全転置インデックスを採用
  - 単語の出現位置も保持

単語	(文書番号, 出現位置)列
国立	(1, 10), (10, 3), (12, 5)
情報	(10, 4), (11, 3), (12, 7)
学	(10, 5), (14, 3), (16, 4)
研究所	(2, 5), (4, 6), (10, 6)

- 高速なフレーズ検索が可能



# 位置情報を用いたフレーズ検索(1)

- 例えば、以下のような文書があると仮定
- **文書番号10**
  - 「ここは国立情報学研究所です」
- **文書番号12**
  - 「国立のタウン情報」

単語	(文書番号, 出現位置)列
国立	(1, 10), (10, 3), (12, 1)
情報	(10, 4), (11, 3), (12, 4)
...	...



# 位置情報を用いたフレーズ検索(2)

- (例) 「国立情報」 で検索
  - 位置情報なし：
    - 例えば「国立のタウン情報」が検索される
    - すべての検索結果を検査する → 遅い
  - 位置情報あり：インデックスのみで検索可能

単語	(文書番号, 出現位置)列
国立	(1, 10), (10, 3), (12, 1)
情報	(10, 4), (11, 3), (12, 4)
...	...



# 高速化の工夫

- **バッファ機構**
  - インデックスの一部をメモリにキャッシュ
  - I/O負荷の減少
- **参照ロックが不要**
  - 高い並列性
  - RCU (Read Copy Update)により実現
- **自動再配置**
  - インデックスの不要な領域を再配置
  - 更新・削除による性能劣化を防ぐ



# 高速のおさらい

- Sennaは完全転置インデックスを採用
  - 高速な検索
  - 高速なフレーズ検索
- Sennaは速度に気を遣った実装がされている



# 高 精 度

について



# 検索ノイズ・検索漏れ

- 検索エンジンでよくある不満...
  - 適合する文書が見つげにくい
- 検索ノイズ
  - (例) 「インド」 → 「インドネシア」  
「京都」 → 「東京都」  
「先生」 → 「この先生きのこるため」
- 検索漏れ
  - (例) 「打ち合わせ資料」 → × 「打合せ資料」
- 精度を表す尺度が必要



# 適合率・再現率とは

- 適合率・再現率
  - 検索エンジンの精度をあらわす尺度
- 適合率
  - 「検索ノイズ」に対する尺度
- 再現率
  - 「検索漏れ」に対する尺度



# 適合率

- 適合率とは
  - 検索された文書の中で、検索したかった文書(=適合する文書)の割合
  - 簡単にいうと、1-ノイズ率
  - 高適合率 = ノイズが少ない
  - (例) 「インド」で検索して40件の文書がhit
    - 「インド」に関する文書 30件
    - 「インドネシア」に関する文書 10件
    - 適合率 =  $30 / 40 = 0.75$



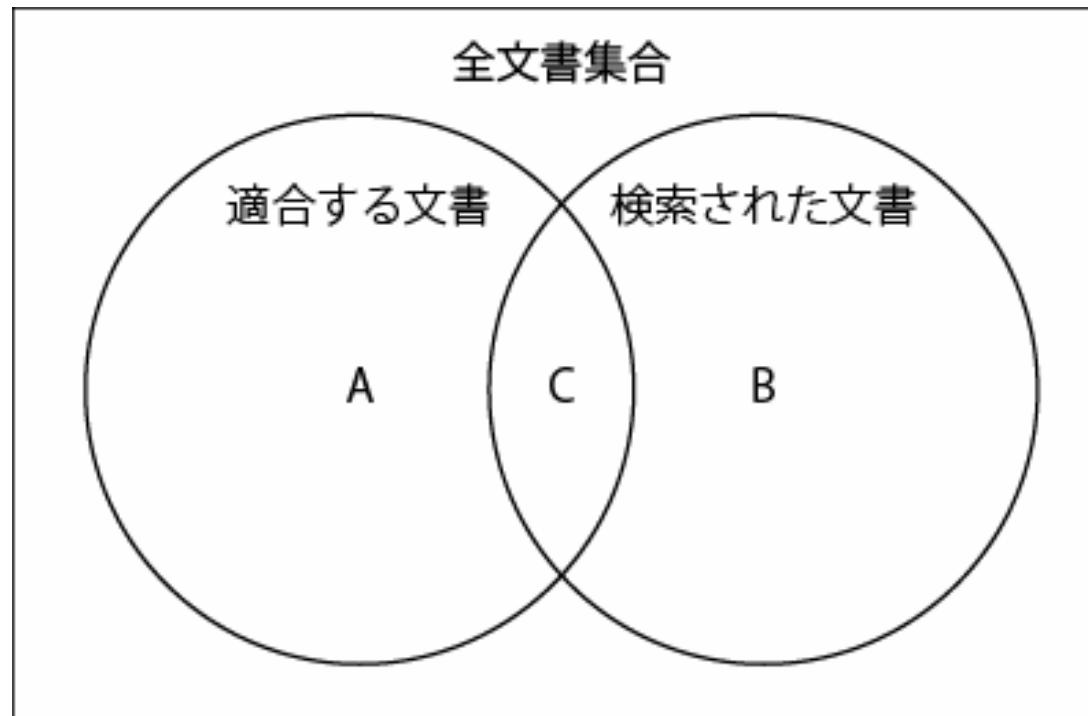
# 再現率

- 再現率とは
  - 適合する文書の中で、実際に検索された文書の割合
  - 簡単にいうと、1-検索漏れ率
  - 高再現率 = 検索漏れが少ない
  - (例) 「打ち合わせ資料」で検索して30件hit
    - 「打ち合わせ資料」を含み検索された文書 30件
    - 「打合せ資料」を含むが、「打ち合わせ資料」ではないため、検索されなかった文書 10件
    - 再現率 =  $30 / 40 = 0.75$



# 適合率・再現率

- 適合率 :  $C / B$
- 再現率 :  $C / A$
- どちらも高いほうがよい
  - 検索ノイズも検索漏れも少ないほうがよい



# 適合率・再現率のトレードオフ

- 適合率のみを高めたい
  - 適合しない可能性が少しでもある文書を、検索結果から除外
  - 再現率の低下 (= 検索漏れの発生)
- 再現率のみを高めたい
  - 適合する可能性が少しでもある文書を、検索結果として採用
  - 適合率の低下 (= 検索ノイズの発生)
- 両者はトレードオフ関係



# 適合率・再現率へのニーズ

- 高い適合率が求められるケース
  - 大規模な文書検索
  - Webサーチエンジン

ノイズが多くて欲しい情報にたどり着けない

- 高い再現率が求められるケース
  - 特許文書検索

調査漏れがあると特許侵害リスクを負う



# 転置インデックスの形式と精度

- 転置インデックス（おさらい）
  - ある単語が、どの文書にあるかを保持
  - (例) 書籍巻末にある用語索引
- 単語の選び方で精度が変わる
  - 形態素
    - 形態素：言語の中で意味を持つ最小単位
    - 適合率が高まる
  - N-gram
    - N-gram：任意のN文字幅の部分文字列
    - 再現率が高まる



# 形態素ベース

- 形態素単位でのインデックス
  - 形態素に一致しない語が検索されない
  - 高適合率
  - Googleも形態素ベース

単語	文書番号列
インド	1, 10, 12
インドネシア	5, 10, 11
京都	1, 2, 3, 6, 10
東京都	1, 2, 4, 5, 11
先生	10



# N-gramベース

- N文字の部分文字列単位でインデックス
  - N文字の部分文字列が一致すれば検索される
  - 高再現率

部分文字列	文書番号列
イン	1, 3, 5, 9
ンド	1, 3, 5, 9
ドネ	3, 9
ネシ	3, 9
シア	3, 9



# 形態素ベース

- Sennaは形態素ベースの転置インデックスを採用
  - 形態素解析：文書から形態素を取り出すこと
  - 形態素解析はMeCabを利用  
<http://mecab.sourceforge.jp/>
- 高い適合率
  - 「京都」で「東京都」が検索されない
- ほとんどの場合検索漏れはないが...



# 堰東京都祇園囃子って知ってる？

- 堰東京都祇園囃子
- 「せきひがし きょうと ぎおんばやし」  
って読みます



# 堰東京都祇園囃子とは・・・

- 直山半田銀山が隆盛の頃の芸能・風流が伝播された正調京都祇園囃子の流れをくむ「堰東京都祇園囃子」



寛文九年(1769)半田銀山神社大祭



<http://www.db.fks.ed.jp/txt/10004.001/html/00009.html>より引用



# 形態素解析の失敗

- 「堰東京都祇園囃子」について  
形態素解析を行うと・・・
- 「堰」「東京都」「祇園」「囃子」  
という形態素が取り出される場合がある
- 「京都」で検索しても  
「堰東京都祇園囃子」が検索されない！



# Googleにも弱点があった！

- 福島県のイベントリストのページ  
[http://www.pref.fukushima.jp/kanako/jnl\\_lst/jnl1028.html](http://www.pref.fukushima.jp/kanako/jnl_lst/jnl1028.html)
- Googleでは、「京都」で検索しても上記のページは検索されない！



# 適合率と再現率の両立

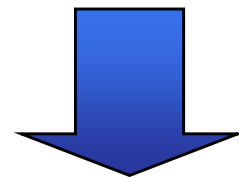
- おさらい
- Sennaは形態素ベース
  - 高い適合率
- 適合率と再現率はトレードオフの関係
  - 再現率が下がる
- わずかながら検索漏れが生じてしまう
  - 堰東京都祇園囃子

Sennaにも検索漏れがあるの...?



# Sennaの工夫

- 適合率と再現率を両立する独自方式
  - 形態素の部分一致検索が可能なデータ構造
  - 「京都」で検索して「東京都」を含む文書が検索可能
  - 「堰」「東京都」「祇園」「囃子」と形態素解析されても、「京都」で検索可能



適合率が落ちてしまうのでは...?



# Sennaの部分一致検索

- 適合率が落ちるとなぜうれしくないか？
  - 検索結果にノイズが多くて  
欲しい検索結果が見つからない
- Sennaの考え方
  - 検索漏れが多い場合のみに限って  
部分一致検索をすればよい
  - 部分一致の検索結果が目立たなければよい
- それぞれに対応した処理が選択可能



# 部分一致検索(1)

- Sennaでは、検索結果がn件以下だった場合のみ部分一致検索を行うことが可能
  - 検索結果が少ない場合、検索漏れの存在が疑われる
  - 部分一致検索の実行によって、検索漏れの救出
  - (例) 「インド」で検索して0件ヒットだから、「インドネシア」を含む文書も検索する

適合率を阻害せずに再現率を向上



## 部分一致検索(2)

- 検索結果ごとに、スコアという値を計算
  - 文書がどれだけ適合しているかを表す
- 通常、検索結果はスコア順で並べる
- Sennaでは、部分一致による検索結果のスコアを相対的に下げることが可能
  - 検索されるが、検索結果の後のほうに表示
  - ユーザは検索結果の最初のほうしか見ない

主観的な適合率を保ちつつ再現率を向上



# Sennaでの高精度検索

- おさらい
- Sennaは形態素ベース
  - 高い適合率
- 適合率と再現率はトレードオフの関係
  - 再現率が下がる
- わずかながら検索漏れが生じてしまう
  - 堰東京都祇園囃子
- 検索漏れを部分一致検索でカバー
  - 適合率を維持しつつ再現率を向上



# 文書単位での検索の問題点(1)

- 「渋谷 ラーメン」で検索
  - おそらく「渋谷のラーメン情報」を知りたい
- 例えば、こんなWebページは検索されて欲しくない!!!!



# 文書単位での検索の問題点(2)

- 3月17日  
今日渋谷に行った。...
- 3月19日  
今日は高円寺でラーメン食べた。...
- 確かに同じページに「渋谷」も「ラーメン」も出現している文書
- 渋谷のラーメン情報はない・・・



# 段落・パラグラフ情報の保持

- 段落単位での検索が可能
  - 段落：文章中の任意の一部
  - (例) ブログのエントリごとの検索
- スコアの重み付けが可能
  - パラグラフ：段落中の任意の一部
  - パラグラフ単位で重み付けしたスコアを保持
  - (例) ブログのエントリごとの見出しに  
単語が出現していたら高いスコアを付与

高精度な検索の実現



# 高精度のおさらい

- Sennaは精度重視の設計
  - 形態素ベースの転置インデックス
  - 適合率を重視しつつ、再現率も高める
- Sennaは段落検索・スコアの重み付けが可能
  - 内容がいくつかに分割できる文書に対しても適合率の高い検索が可能



# 組込型

について



# 組込型

- 文書を保存しない
  - 文書の内容そのものは保持せず、転置インデックスのみを作成する
  - 他の文書管理システムと統合して利用
    - MySQL
    - PostgreSQL
    - etc...



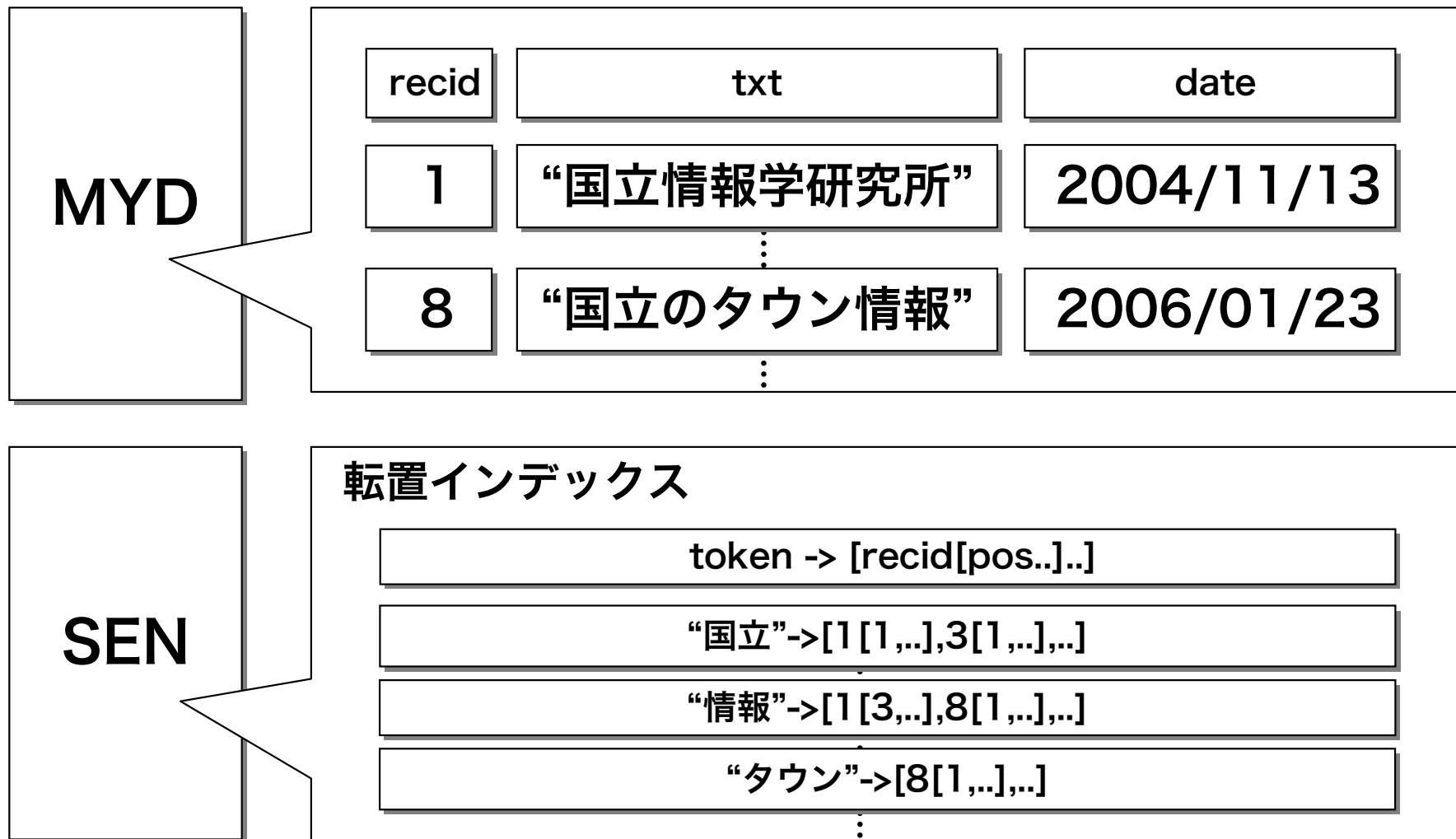
# MySQLバイインディング

- MySQLが文書を管理
  - SQLを用いた問い合わせ  
(例) `SELECT * FROM table1  
WHERE MATCH(col1)  
AGAINST('クエリ');`
  - SQLの高い記述力
    - 他カラムでの絞り込み・並び替え・グループ化
    - 複雑な問い合わせが可能
    - (例) あるユーザが、閲覧権限のある文書のみに対して全文検索を行い、その結果を日付順に並べたい



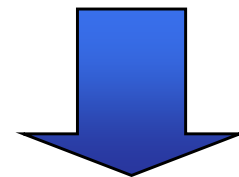
# MySQLバインディングの構造

- オリジナル全文検索インデックスを置換



# オリジナルの全文検索の問題点

- 日本語など、分かち書きしない言語が検索できない
- フレーズ検索が遅い
- インデックス更新が極端に遅い
- 全文検索と、他のインデックスとを組み合わせると検索できない



Sennaで解決



# MySQLとSennaの比較表

実験データ : Wikipedia 英語版 458,713レコード 1088MB

	MySQL オリジナル	MySQL+ Senna
インデックスサイズ	109 MB	1028 MB
フレーズ検索 (‘united states’)	44.91 sec	0.40 sec
既存のレコードに インデックス付与	1,474 sec	1,808 sec
インデックス付与後に レコードを追加	28,182 sec	1,839 sec
order by 主キー	20.33 sec	0.89 sec
where 全文検索 and 主キー > 20万	6.55 sec	0.32 sec



# デモ

- MySQLバイインディングのデモ



# PostgreSQLバイインディング

- **Ludia**

<http://www.nttdata.co.jp/services/ludia/index.html>

- (株)NTTデータが開発
- SQLを用いた柔軟な問い合わせ  
(例) `SELECT * FROM table1  
WHERE col1 @@ 'クエリ';`



# Pgバイインディングの構造

- インデックス拡張機能を利用

Postgre  
SQL  
データ

recid	txt	date
1	“国立情報学研究所”	2004/11/13
⋮	⋮	⋮
8	“国立のタウン情報”	2006/01/23
⋮	⋮	⋮

SEN

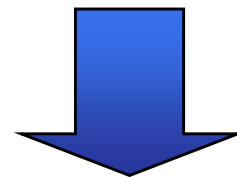
転置インデックス

token -> [recid[pos..].]
“国立”->[1 [1,..],3 [1,..],...]
⋮
“情報”->[1 [3,..],8 [1,..],...]
⋮
“タウン”->[8 [1,..],...]
⋮



# GINの問題点

- PostgreSQL 8.2で全文検索が可能に
  - GINインデックス
- MySQLと同様の欠点
  - 日本語など、分かち書きしない言語が検索できない
  - フレーズ検索が遅い



Sennaで解決



# 組み込み型のおさらい

- おさらい
- Sennaは組み込み型の検索エンジン
  - 文書データは保存しない
- MySQLバインディング  
PostgreSQLバインディング
  - SQLを用いた柔軟な問い合わせ
  - 組み込みの全文検索の欠点を解消



# 以上がSennaの三大特徴です

- **高速**
  - **高精度**
  - **組込型**
- 
- **是非覚えてネ！**
  - **他にも...**



# その他の機能

- UTF-8対応
- 純粋なN-gramインデックス作成機能
  - 高い再現率
- 関連文書検索
  - クエリに指定された文書と、内容が類似する文書を検索
- 近傍検索
  - 指定された複数の単語が、近傍に現れる文書を検索



# 導入実績

- タワーレコード商品検索

<http://search.tower.jp>

**TOWER SEARCH**  
beta

- はてなキーワード検索

<http://search.hatena.ne.jp/keyword?mode=top>



- その他、続々導入中!!!!!!!!!!!!



# 今後の開発予定

- ストレージ機能の開発

ストレージ

recid	txt	date
1	“国立情報学研究所”	2004/11/13
⋮	⋮	⋮
8	“国立のタウン情報”	2006/01/23
⋮	⋮	⋮

SEN

## 転置インデックス

token -> [recid[pos..].]
“国立”->[1 [1,..],3 [1,..],...]
⋮
“情報”->[1 [3,..],8 [1,..],...]
⋮
“タウン”->[8 [1,..],...]
⋮



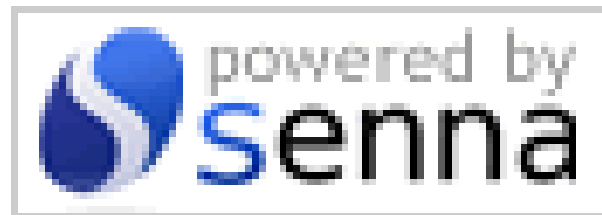
# 今後の開発予定

- ストレージ機能
  - Senna単体での利用
    - デスクトップ検索などが手軽に実装可能



# 発表終了

- 詳細な情報は、  
<http://qwik.jp/senna/>  
をご覧ください
- Sennaのバナーもあります



- 開発者募集中です!!!
- ご清聴ありがとうございました



# 補足説明

- この資料は、ACM SIGMOD日本支部第35回支部大会で発表した際のスライド資料です。



# 改版履歴

- 2006/11/06
  - 発表当日
- 2006/11/14
  - 再現率の説明のスライドにあった誤りを訂正

