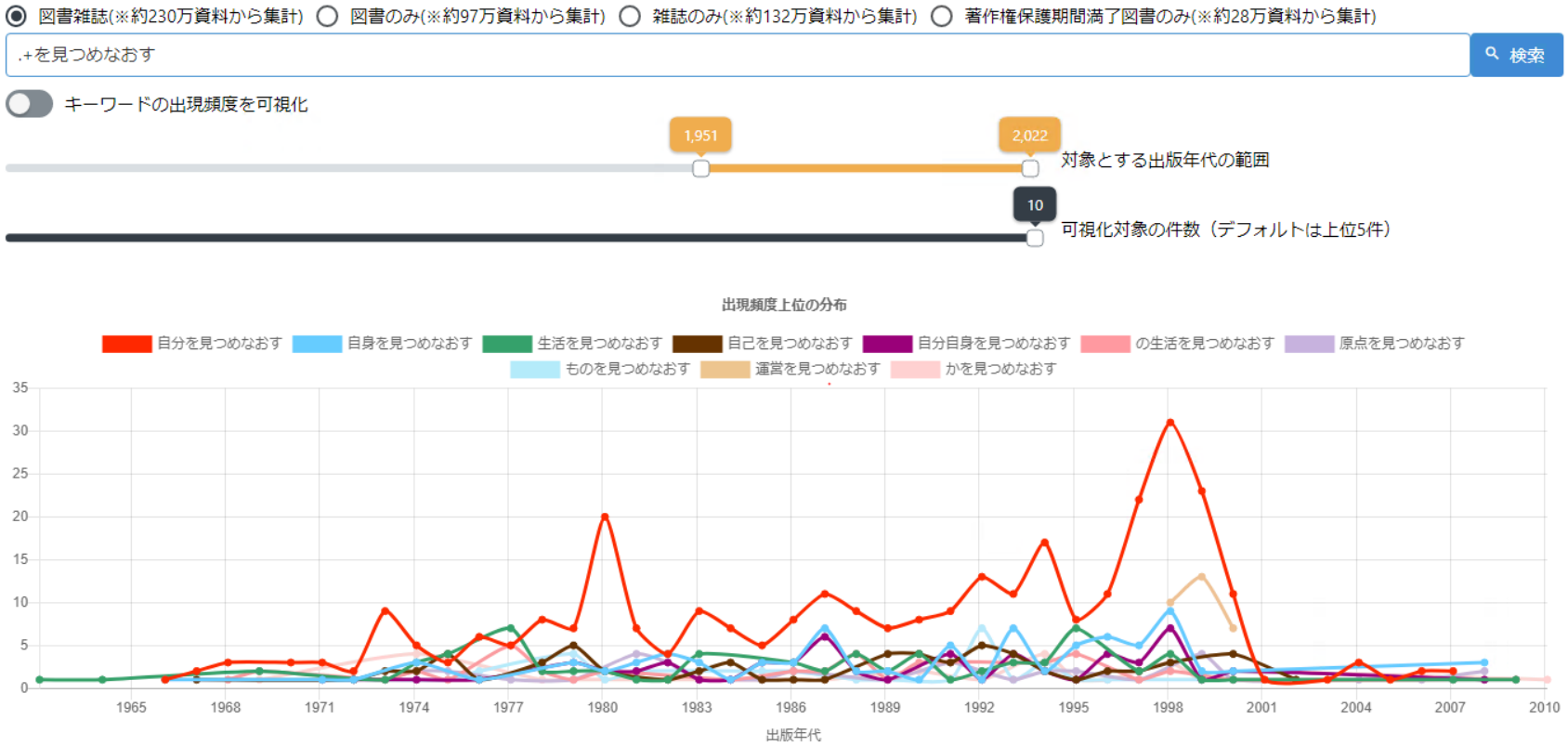
日本語資料の全文テキストデータ分析ツール NDL Ngram Viewerの開発について

国立国会図書館　青池 亨



Ngram Viewerとは何か

- 書籍の全文テキストデータを利用して，特定の単語やフレーズの頻度を出版年代に沿って検索・可視化できるサービス
- 2010 年に公開された“Google Books Ngram Viewer”が端緒
- 他に、Bookworm(Hathi Trust), Gallicagram(Gallica)等がある

→日本語のフレーズによって検索可能なNgram Viewerは2021年時点においては存在しなかった

※Bookwormの一部コレクションは日本語の「単語」単位で検索可能

NDL Ngram Viewerとは何か

- 国立国会図書館がOCRテキスト化事業で作成したテキストデータを活用した日本語版Ngram Viewer
- 表記揺れが多い日本語向けに正規表現検索をサポート
- 著作権保護期間満了図書資料約28万点（次世代デジタルライブラリーと同様）に対象を限定して2022年5月に一般公開

https://lab.ndl.go.jp/ngramviewer/

- 2022年12月21日より、国立国会図書館デジタルコレクションでほぼ全てのデジタル化資料を全文検索可能に

→NDL Ngram Viewerも近々に対象を大幅に拡大予定

NDL Ngram Viewerの対象範囲

バージョン1（2022年5月公開）
- 著作権保護期間が満了した図書資料（約28万点）

バージョン2（2023年1月以降公開予定）
以下の4種類の対象を切り替えて利用可能
- 図書及び雑誌資料全件（約230万点）
- 図書資料全件(約97万点)
- 雑誌資料全件(約132万点)
- 著作権保護期間が満了した図書資料（約28万点）

※対象はいずれも2020年12月時点で国立国会図書館デジタルコレクションから提供していた資料

【参考】国立国会図書館のOCRテキスト化事業(2021年度実施)

https://lab.ndl.go.jp/data_set/ocr/r3_text/

●対象資料

2020年度末時点でデジタル化済の資料：

1870年代から1968年までの図書・1870年代から1990年代までに刊行された雑誌等、**247万点・約2億2300万画像**

●課題

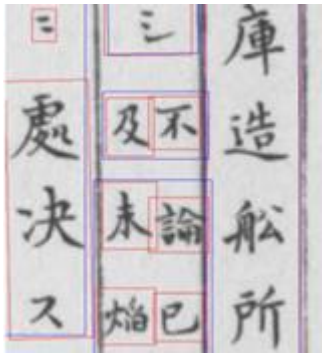
1870年代から1940年代の資料は古い字体（例：学→學）が使用されており、商用OCRサービスそのままでは精度が低い

古い書籍のデジタル化資料は、文字のフォントやサイズ・行間が現代と異なる、汚れ・ノイズ・透けて見える文字がある等のため、OCRの精度が下がる

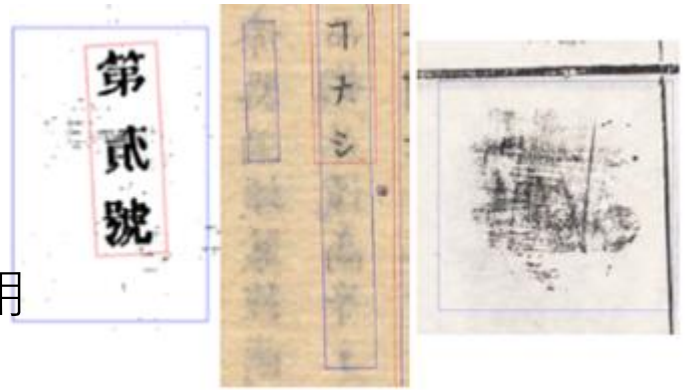
●方法

既存のAI-OCRサービスを機械学習技術により改良し、NDL所蔵資料用に最適化（外部委託）

●結果　OCR性能（F-measure）は対象とする全年代の平均0.96を達成



文字サイズ・行間が現代と異なる資料



汚れ・ノイズ・透けて見える文字がある資料

本論文及び発表のテーマ

NDL Ngram Viewerを開発するにあたり、次の3つの技術的検討が必要であった

1. テキストデータに含まれる文字列の分割方法

2. 分割した文字列の集計方法

3. 集計した情報を検索システムとして提供する方法

膨大な分量のテキストデータを検索可能なNgram Viewerを
「**限られた計算資源を用いて**」「**正確に**」「**高速な**」
方法で実現するための技術的検討、分析及び実装したサービスを説明する

1. テキストデータに含まれる文字列の分割方法

- 日本語の場合、英語等と異なり単語間の区切りが自明ではない

→ngramを抽出する前処理として形態素解析による分かち書きが必要

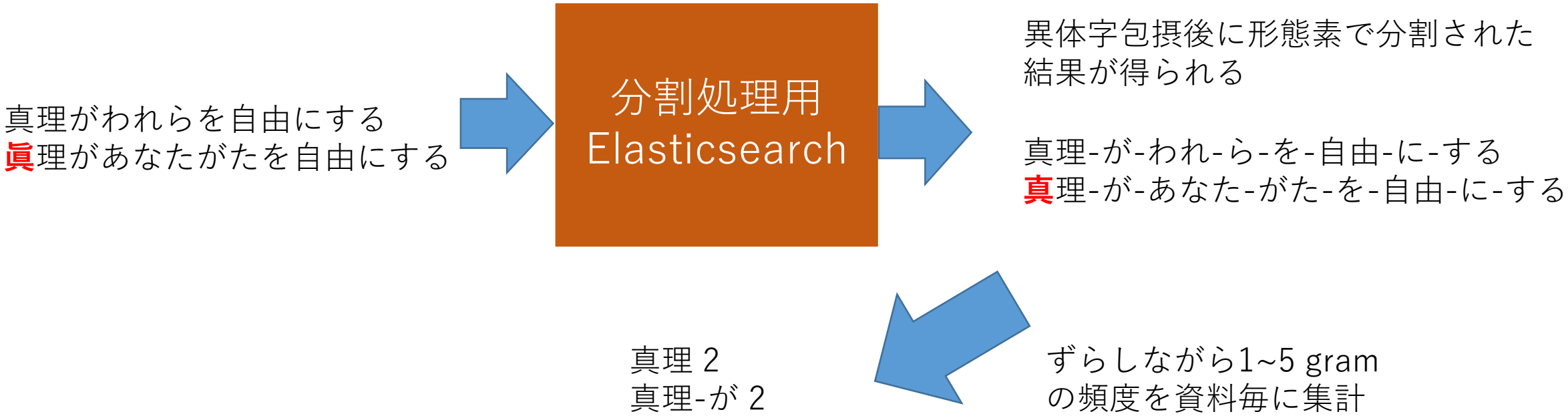
- 古い資料が多く存在するので、旧字体等異体字の包摂処理もしておきたい

- 字体を包摂しながら高速に分かち書きをするため、マッピング表を持たせたElasticsearchにKuromojiプラグイン（Mecab+IPA辞書と互換）を導入

- 資料単位にまとめたテキストデータをElasticsearch APIに投げて、分かち書き結果を受け取る

→文字包摂と分かち書き処理をElasticsearch内で一括で行うことで高速化

1. テキストデータに含まれる文字列の分割方法



2. 分割した文字列の集計方法

- 資料毎のngramの出現頻度が求まったので、今度は対象資料全部の中に各ngramがそれぞれ何件ずつあるか集計する

- ngramを高速にカウントするためには、集計サーバのメモリ(DRAM)上にデータを保持する必要がある

- 集計サーバ上のメモリは当然に有限※であるので、物理的な制約から出現する全てのngramをメモリに載せることはできない

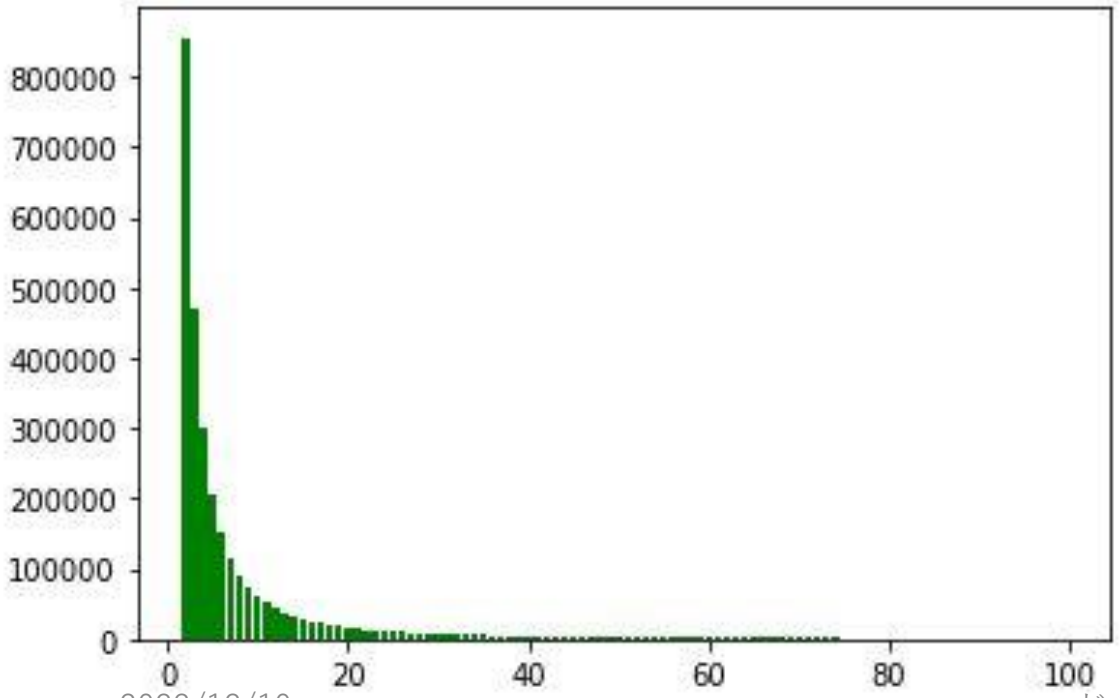
→必要な情報の取捨選択が必要

※今回の開発時に利用した当館内のローカルサーバは、384GBのDRAMを搭載

2. 分割した文字列の集計方法

著作権保護期間満了資料28万資料のテキストデータを対象に、 2回以上出現したngramをランダムサンプルした合計200万ngramの分布を調べた

2回以上の出現回数（横軸）と該当するngramの種類数（縦軸）

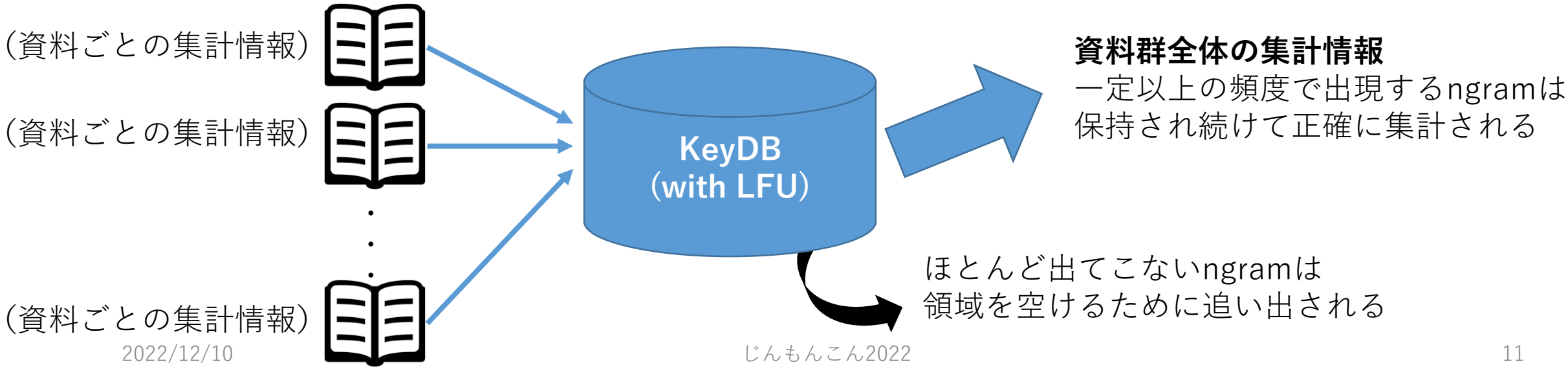


出現回数の閾値を増やしていくと該当するngramの種類数は指数関数的に減少する
出現回数の少ないngramには、OCRの読み取りミス等が含まれる
こうしたngramを効率的に捨てることができれば、現実的なメモリサイズで計算を行える

2. 分割した文字列の集計方法

メモリの上限サイズが固定された状況で、アクセスの少ないマイナーなキーを追い出しながらキャッシュするアルゴリズムとして、 LFU（Least Frequently Used）がある

KeyDB(Redis互換のKey Value Store)はLFUによるキャッシュアルゴリズムをサポートしているため、KeyDB上でngramを集計することとした



2. 分割した文字列の集計方法

一律で閾値を総出現頻度4以上とした場合の集計対象となるngramの種類数は次のとおりとなった

- 著作権保護期間が満了した図書資料（約28万資料，0.3億コマ画像）
  →8.3億ngram
- 図書資料(約97万資料，1.4億コマ画像)→8.9億ngram
- 雑誌資料（約132万資料，0.7億コマ画像）→ 8.5億ngram

3. 集計した情報を検索サービスとして提供する方法

2で集計したKeyDBのダンプファイルをパースし、検索サービス用Elasticsearchに集計結果をインデックスしている

Elasticsearchを利用しているのは、正規表現検索をサポートしたかったから

Elasticsearchの正規表現検索特性

「前方一致検索が高速な反面、後方一致検索は非常に遅い」

→検証時、3,000倍程度応答時間に差があった(前方一致:10msに対し、後方一致:30s)

3. 集計した情報を検索サービスとして提供する方法

【解決策】全てのngramに順序を反転させたフィールドを持たせる
例：「館書図会国立国」「館書図国帝」……
- クエリを見て、後方一致の正規表現であれば、
「**反転したキーワードを前方一致で探す正規表現**」
に変換して反転フィールドに問い合わせる

「.*図書館」→（**前方一致のクエリに変換**）→「館書図.*」→（**検索**）
→検索結果を反転前のキーワードとして返す

Elasticsearchが得意な検索方法へと内部で変換を行い高速な応答を実現

全体を通してのまとめ・考察

- 物理的な制約のあるメモリサイズの中で集計処理を行うために、LFUアルゴリズムを組み込んで集計を行った
- 反転文字列フィールド等を活用することで小さなリソースで正規表現検索と高速な応答を両立したサービスを提供可能と分かった
- 低頻度のngramは集計結果の信頼性が低い。適切な閾値を資料種別ごとに見積もることが今後の課題
- 例えばngramに品詞を付与したり、集計元のNDC等を利用してコレクションを出し分けたりする場合には、管理すべきngramの種類数が増え、今回の方法ではより多くのメモリを必要とするため、集計方法の改良が必要