社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－DBS－133　(14)
2004－ＦＩ－ 75　(14)
2004／5／14

複数書誌データベース統合における重複エントリーの高速検出法

相澤 彰子　　大山 敬三　　高須 淳宏　　安達 淳
{aizawa, oyama, takasu, adachi}@nii.ac.jp

国立情報学研究所

互いに独立に構築された複数の大規模データベースの統合では、重複するレコードを検出し統合する「レコード照合」が重要な役割を果たす。本稿では、レコードの重複登録候補を高速かつ精度よく数え上げるための実践的アプローチを提案し、実際に書誌データベースに適用した結果を報告する。

A Fast Method for Duplicated Entries Detection in Bibliographic Databases

Akiko Aizawa　　Keizo Oyama　　Atsuhiro Takasu　　Jun Adachi

National Institute of Informatics

Record linkage refers to a technique for detecting and eliminating duplicate records, and is crucial in integrating multiple databases that have been independently generated. In this paper, we propose a fast and efficient method for enumerating duplication candidates, and report some experimental results using real-world bibliographic databases.

1　はじめに

異なる情報源の間で共通するレコードを同定する「レコード照合（record linkage）」問題は歴史が古く、すでに 1959 年には Newcombe らにより、計算機による 自動化に関する論文が発表されている [1]。レコード照合問題は、データベース・クリーニング技術の１つとして位置づけることができるが [2]、近年では、データウェアハウス構築や Web マイニングといった話題の盛り上がりを受けて、機械学習や情報検索分野における研究成果の適用も試みられている [3]。

レコード照合問題を考える上で特に注意が必要なのは、まず第一に、レコード照合の機能は単独のモジュールとして存在するのではなく、値の整合性チェックやデータベース間でのスキーマ変換等を含む、より広範なデータベース技術との連携ではじめて実現される点である。第二に、レコード照合問題ではデータベースの品質を高い水準に保つことが最優先されており、領域に依存しない手法の開発が基本となる機械学習や情報検索の場合と異なり、品質を保証するためには人手による領域知識の実装や判定処理までが適用される点である。すなわちレコード照合 問題の本質は、単なる学習や知識獲得アルゴリズムの適用なのではなく、現実世界のデータを前提とした総合的なシステム設計である。

本稿では以下、**2.** でレコード照合システムの一般的な構成を概観したのち、**3.** で照合候補の数え上げ法に焦点をあてて、N グラムの一致とブロッキングルールの獲得に基づく手法を提案する。**4.** では実際の書誌データベースへの適用結果を紹介し、**5.** でまとめと今後の方向を述べる。

2　背景

2.1　一般的なレコード照合システムの構成

レコード照合における技術的なポイントは、(1) 大規模なデータを扱うため高い効率性が求められるこ

と、および (2) データベース品質維持のため高い信頼性が求められること、の２点である。しかしながら両者はトレードオフ関係にあり、すべてのレコード対を単純に比較するだけでは (1)(2) を同時に満足することはできない。このためレコード照合システムでは、候補選別、自動判定、人手による判定などを含む多段階の処理が必要となる。

レコード照合システムの標準的な処理の流れを図 1 に示す。以下に、それぞれの処理機能を簡単にまとめる [4]。

(1) セグメンテーション（segmentation）
テキストを解析してレコード毎の属性値を抽出し、データベースにロードする。（例：“1999.8” を “year⇒1999, month⇒8” など）

(2) 正規化（normalization）
辞書や変換ルールを用いて、入力時の表記の揺れや用語の不統一を修正する。（例：“1999 年” と “’98” など）

(3) 選別（selection）
効率のよい方法を用いて、互いに重複する可能性があるレコード対の候補を数え上げる。

(4) 比較（comparison）
候補レコード対の照合スコアを計算し、照合可（match）/ 要判定（possible match）/ 照合不可（non-match）のいずれかを判定する。

(5) 検証（verification）
分類結果を受けて、必要に応じて人手による追加判定を行い、最終的な照合結果を出力する。

(6) 調整（tuning）
照合結果に基づきシステムの性能を評価し、選別や分類等におけるパラメタを調整する。また、正規化のための辞書やルールを拡張する。

上記のうちで、レコード照合問題の中心的な検討課題となるのは、(2) の候補選択および (3) の候補比較である。これらは両者とも、いかにしてレコード間の照合スコア（類似度）を求めるか、という問題に帰着させて考えることができる。前者では、精度は低くてよいが取りこぼしがなく、大量のレコードを現実的な速度で処理できるような高速な手法が求められる。後者では、コストが高くても信頼性の高い手法が求められる。後者については、従来の機械学習が有効に適用できると考えられることから、以下、本稿では前者に焦点をあてて検討を行う。

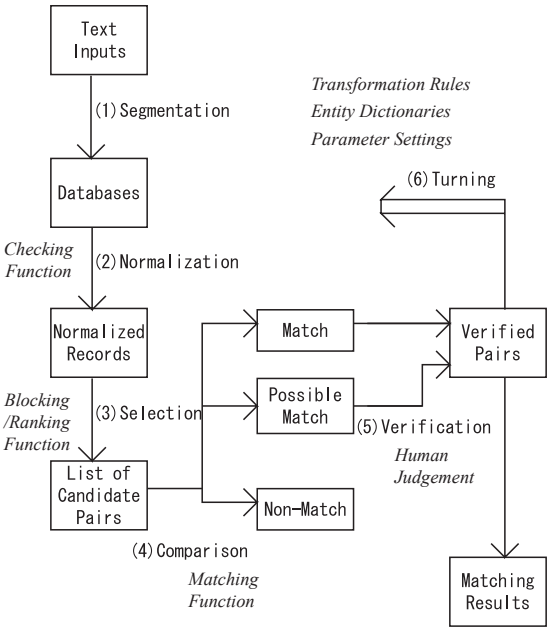

図 1: レコード照合システムにおける処理の流れ [4]

2.2 代表的な候補選択法

候補選択の目的は、データーベースの登録レコードを小さなブロックに分割することで、同定候補を絞り込み、コストの高い照合関数の適用回数を削減することである。たとえば、氏名、生年月日、郵便番号等が同じレコードどうしを比較するといった操作が例としてあげられる。このことから候補選択は「ブロッキング」と呼ばれる場合も多い。

ブロッキング法のための代表的な方法として、(i) ソートに基づく方法、および (ii) 類似度ランキングに基づく方法、の２つがある。

ソートに基づく方法は、候補選択の伝統的な方法として研究されてきた。最も単純な方法である standard blocking [5] では、たとえば「苗字の先頭４文字」など、キーとなる属性（あるいは属性の組みあわせ）を１つ決めて、同じキーの値を持つレコードどうしを照合の候補とする。複数のキーに対してこのような操作を行う場合を multiple pass blocking と呼ぶ。また、sorted neighbourhood method (SNM) [6] では、あらかじめ定めたキーの値にしたがってすべてのレコードをソートし、そのリスト上で固定長ウィンドウの範囲内にある近接レコード群を照合候補とする。SNM にはあらかじめグループ化したレコード群の中でソートを適用する clustering SNM、複数のキーでソートを繰り返す multi-pass SNM など、幾つかのバリエーションが存在する。また、リストを走査する

際に、最新の代表的なレコードだけを特別なキューに入れて効率化をはかる priority queue method [7] も提案されている。

一方、近年になると主に情報検索分野からからのアプローチとして、類似度ランキングに基づく候補選択法が提案されるようになった。その代表例として、bigram indexing [8] と canopy clustering [9][10] をあげることができる。Bigram indexing は、キーの値を文字単位バイグラムの集合に変換し、ある一定の閾値以上でバイグラムが一致するレコードを候補とする方法である。閾値を設定することで、効率とあいまい性のトレードオフを可能にしている。一方、canopy clustering は、情報検索システムで広く用いられている tf-idf を類似度尺度として用いる方法である。レコードをランダムに順次選択しながら、tf-idf による距離が一定値以下であるようなレコード群をまとめて、canopy cluster と呼ばれるグループを構成する。最後に、同一の canopy cluster 内のレコードどうしを照合候補とする。

3 提案手法

3.1 基本的な考え方

さて上記では、照合候補を数え上げるための代表的な２つの方法を紹介したが、これらの方法は、大規模なデータベースへの実際の適用において、必ずしも処理効率および性能の双方を満足するものではない。まず、(i) のソートに基づく方法では、照合もれをいかに減らすかが課題となり、キーの値や組み合わせ等を経験的・実践的に工夫することが必要となる。一方、(ii) のランキングによる方法では、レコード毎に候補の抽出を行うため見落とし率は比較的低いが、多数のブロックが独立に生成されることになり、重なりの少ないブロックを一括して生成する (i) の場合と比較して、大きく効率が落ちてしまうのが難点である。

上記を背景として本稿では、以下の２つを特徴とするレコード照合方法を提案する。

(1) 単語 N グラムに基づく簡易ブロッキング
(2) ブロッキングルールの自動獲得と適用

提案手法ではまず、(1) により、同定候補となるペアを高速に数え上げ、(2) では、(1) の結果に基づき有効なブロッキングルールを選択して全レコードに適用する。以下、提案手法の概略について述べる。

3.2 単語 N グラムに基づく簡易ブロッキング

関連する従来研究 [8] では、属性値のバイグラムの一致数に基づき照合候補のランキングを行うが、この方法では、レコード毎にバイグラムの inverted file を参照し、OR 演算を行うといった操作が必要となり、特に大規模なデータベースで高頻度バイグラムを扱う場合に、計算コストが大きな問題になる。

そこで本稿では、サフィックスアレイを用いて高速に簡易ブロッキングを行う方法を提案する。提案手法は、バイグラムのかわりに最長一致の任意長 N グラムに注目してレコードのグループ化を行うもので、ブロックを一括して数え上げることから高速な処理が実現できる。具体的にはまず、すべてのレコードをサフィックスアレイの形で読み込み、次に、サフィックスアレイを順に走査して連続した N 単語が一致するレコードをブロックにまとめる（図 2)。次に、各ブロックごとに、(i) 最長一致 N グラムの長さ、(ii) 一致する単語数、を含む表 1 のような情報を求め、一定条件を満足するブロック内のレコード組を照合候補として出力する。ここで、一致単語数は、ブロック内のレコード間の編集距離に基づき順序を考慮して求めたものである（表 1)。

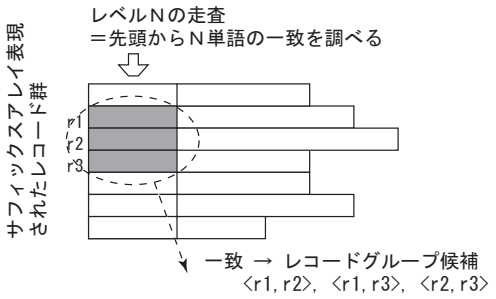

図 2: サフィックスアレイの走査とブロックの抽出

上記の手順に要する計算コストは、基本的にサフィックスアレイ構造を作成するためのポインタのソート時間である。現在の実装では、OS 組み込みのソート関数の効率が支配的な要因となっている。

3.3 ブロッキングルールの自動獲得と適用

本稿では、レコードをブロックにまとめるための共通パターンを「ブロッキングルール」と呼ぶ。まず理解を容易にするため、perl のパターン記法を用いて表されたブロッキングルールの例を表 2 に示す。この例は、次節の実験において 2 つの書誌データベース間での登録書誌の照合を行うために実際に用いた

表 1: 簡易ブロッキングによるブロックの出力例

○平井昌夫／漢字書写能力を高めるにはどうすればよいか／国文学解釈と鑑賞／17,7,1952／30-36／1952
○吉竹勝／余りある文章題の解決能力を高めるにはどうすればよいか／日本数学教育会誌／49,2／33-35／1967
○加藤一典／sds-ポリアクリルアミドゲル電気泳動で示された正常乳と乳房炎乳の異なるタンパク質パターン／the journal of veterinary medical science／51,6／1275-1278／1989
○加藤一典／sds-ポリアクリルアミドゲル電気泳動で示された正常乳と乳房炎乳の異なるタンパク質パターン〔英文〕／日本獣医学雑誌／51,6,1989／1275-1278／1989

† 一致部分を下線で示す。

もので、このルールの適用により、たとえば DB2 で ' 雑誌 ID' が 'NA10539261'、' 巻号' が'0034(0009)' であるものと、DB4 で' 雑誌名' が ' 情報処理学会論文誌'、' 巻' が '34' で ' 号' が'9' である登録書誌が、同一ブロックにまとめられることになる。

表 2: ブロッキングルールの例（１）

適用条件	<src>DB2<\/src><ncid>AN10539261<\/ncid>
対応パターン	<voln>[^\d]*(\d+)\([^\d]*(\d+)\)<\/voln>
ブロック ID	"$1"."_"."$2"
適用条件	<src>DB4<\/src><jnrl>情報処理学会論文誌<\/jnrl>
対応パターン	<voln>[^\d]*(\d+)<\/vol><no>[^\d]*(\d+)<\/no>
ブロック ID	"$1"."_"."$2"

ブロッキングルールの自動獲得では、まず、上記の「対応パターン」にあたる部分（すなわち perl のパターン式で記述された部分）の対応を、あらかじめ人手により対応表に登録しておく。次に、前出の簡易ブロッキング法により信頼性高く照合可能と判定されるレコード対を自動抽出する。最後に、抽出したレコード対の中に、登録済の対応パターンに合致するものが一定数以上含まれれば、残りの共通部分（すなわち、'DB2', 'AN10539261', および 'DB4', ' 情報処理学会論文誌' にあたる部分）を取り出して適用条件とし、新しい１つのブロッキングルールとして登録する。

ここで、「対応パターン」にあたる部分は必ずしも人手で登録しておく必要はなく、合致パターンがない場合には、機械的にふられるブロック ID 番号を用いて直接、表 3 の例に示す形のブロキングルールを採用することも可能である。

次節の実験では、人手によるパターン生成のコストを勘案して、対応パターンを用いるルール獲得法（表 2）と用いない獲得法（表 3）の両者を相補的に用いてレコード照合を行っている。

表 3: ブロッキングルールの例（２）

適用条件	<src>DB2<\/src><ncid>AN10539261<\/ncid><voln>0034(0009)<\/voln>
ブロック ID	b.0001
適用条件	<src>DB4<\/src><jnrl>情報処理学会論文誌<\/jnrl><vol>34<\/vol><no>9<\/no>
ブロック ID	b.0001

4 実装と評価

4.1 対象とする書誌データベース

実験では、複数の書誌データベース間での重複書誌の同定を目的として、提案手法によるブロッキング処理を行った。対象となるデータベースは国立情報学研究所および協力機関がサービスとして提供する学術文献データベースの一部で、表 4 に示す４つである。（各データベースの紹介はここでは省略する。）

表 4: 実験対象とするデータベースと登録件数

データベース	登録書誌件数
DB1	623,996
DB2	1,345,570
DB3	5,977,839
DB4	15,464,052

実験ではこれらのデータベースを組にした合計６通りのデータセット（DB1-DB2, DB1-DB3, DB1-DB4, DB2-DB3, DB2-DB4, DB3-DB4）について、本稿で提案する手法により照合候補となる書誌ペアを抽出した[1]。 得られた書誌ペアについては、サービス運用 [11][12] の一環として、実験後に人手による判定作業が行われ、照合可・不可のいずれか（すなわち候補として抽出したペアが同一の書誌を指しているかどうか）が確定している。照合候補の自動

[1]運用の都合上、DB1 の一部（43,618 件分）について先に DB2 との同定・統合作業を行ったため、DB1-DB2, DB1-DB4 については、DB1 の件数はそれを除いた 587,308 件となっている。

抽出に用いた属性は、各登録書誌について、「第一著者名、題目、掲載雑誌名、巻号、年」の５つである。一方、人手判定ではより多くの情報を参照しており、例えば会議の共催や雑誌名の変遷等、様々な条件を考慮して総合的に判定が下されている。

ここで、表 4 の４つのデータベースは長い時間をかけて構築された大規模なもので、各々のデータベース内での重複登録はないものの、属性値の表記が必ずしも統一されているわけではない。また、データベース間では掲載雑誌名、巻号表記、副題や JIS 水準外の外字の扱い方等がまちまちで、同一書誌を指している場合でも、全属性値が完全に一致することは、むしろまれである。

このように、表記の統一がされていない場合には、データベース間でのレコード照合は単純には解決できない問題である。具体的な数値をあげると、人手判定で照合と判定される正解ペアのうち全属性値が完全に一致するのは、実験と同じ正規化プログラムの適用後で約 10% である。また、もっとも特定性が高いと思われる第一著者名および題目の組を手がかりに正規化後単純一致でブロックを生成する場合でも、照合候補として抽出されるのは正解数の 30%に過ぎず、抽出される候補の 7%以上は誤対応である。さらに注意が必要なのは、重複登録されている書誌は全書誌の約 20% であり、残りの 80% については「重複登録がない」ことを判断しなければならないことである。運用システムでは照合もれ 0.5%、照合誤り 0.1% を目標値の目安としているが、これらの数値から、正規化やキーの組み合わせを経験的に工夫するだけでは、実用的なレベルに到達するのは困難であることが予想される。

4.2 処理の流れ

上記を背景に実験では、提案手法による高速で効率的な照合候補の抽出を試みた。ブロッキングにおいては、登録書誌情報を「冊子」（１冊にまとめられた論文誌や論文集）単位にまとめることを目標として、以下の２通りの方法を用いて、異なるデータベース間での「雑誌名、巻号、年」の値（以下「冊子」情報）を対応づけるブロッキングルールを抽出した。

(1) 人手により定義した対応ルールを用いる方法
著者および題目が正規化後に完全一致する冊子対応候補から、あらかじめ人手により与えた対応パターンに合致する適用条件を抽出して、ブロッキングルールとして登録する。（表 2 参照）

(2) 人手により定義した対応ルールを用いない方法
まず単語 N グラムに基づく簡易ブロッキングを適用し、一致の度合いが一定値以上の冊子ペアを抽出して、これらの冊子情報をブロッキングルールとして抽出する。（表 3 参照）

ただし運用の都合から、(1)(2) を相補的に組み合わせ、(2) は (1) で同定からもれた書誌だけに対象を限定して適用した。各々の方法の具体的な実現方法については次節以降で述べる。

なお方法 (2) において、tf-idf 等を利用して冊子どうしの対応をとりブロッキングルールを抽出する方法も考えられる。しかしながら、収録方針の違いから一方のデータベースではごく一部の書誌だけが登録されているケース等もあり、対応は必ずしも自明ではないことから今回の実験には含めなかった。

さて、上記によりブロッキングルールを獲得したのちは、以下の手順にしたがって人手判定のための同定候補を抽出した。

(1) 自動獲得したブロッキングルールを用いて書誌情報を冊子ごとのブロックに分割する。
(2) 同一ブロック内に属する書誌対して、編集距離を基本とする照合関数を適用して照合スコアを計算する。書誌が和英両方の登録情報を持つ場合、双方を照合候補としてスコアの高い方を採用する。
(3) スコアの高い順に同定候補となる書誌ペアを求め、その値にしたがって、「照合可」「要判定」「照合不可」のラベルをつける。
(4) 「照合可」および「要判定」にラベルづけされた書誌ペアを人手判定の候補とする。ここで、判定作業の現場では、「照合可」データについて、経験的に一定の条件を満たすと判断した場合には判定を省略し、自動的に同定する場合がある。

(2) の照合関数について、運用システムではサポートベクタマシンによる評価関数を実装しているが、本実験では、人手により経験的に作成した関数を用いた。また (3) の閾値についても本実験では機械学習を適用することはせず、経験的に設定した。

4.3 方法 (1)：完全一致に基づくルール抽出

方法 (1) ではまず、’著者’ および’題目’ の正規化した値が完全一致する書誌ペアを数え上げた[2] 。 ここで、データベースは同時開催の会議等や連載記事等を含むため、前述のように ’著者’ と ’題目’ が一致しても、書誌としては異なるペアが少なからず存在する。そこで、巻号表記のパターンが、あらかじめ人手により登録した「対応パターン」のいずれかを満足する場合に限り、該当する雑誌名を「ブロッキングルール」として登録することで、ルールを選別し、不適切なブロックの生成を防いだ。

ここで、人手で登録する対応パターンを詳細化すればするほど、ブロッキング性能は高くなる。しかしながら、現実問題として、登録されているのべ雑誌数だけでも総計で 10 万以上あり、同じ雑誌でも入力年代により通号表記か巻号表記かが異なるなど複数の規則が対応する場合もある。また、「臨時増刊号」等の不定期刊行物に関する表記には統一した形式が存在しない。上記を勘案して実験では、十分に一般性のある少数の対応パターンだけを選び登録するようにした。

人手で登録した対応パターンの数および得られたブロッキングルールの数を表 5 に示す。

表 5: 方法 (1) で獲得したブロッキングルールの数

データセット名	対応パターン数	ブロッキングルール数
DB1-DB2	4	148
DB1-DB3	6	612
DB1-DB4	8	806
DB2-DB3	10	464
DB2-DB4	9	356
DB3-DB4	12	6,016

4.4 方法 (2)：N グラム一致に基づくルール抽出

方法 (2) ではまず、対象となる 2 つのデータベースに登録されている書誌情報をあわせて、語を単位とするサフィックスアレイの形式で読み込んだ。和文については、著者名を文字単位で、その他のフィールドを形態素解析ツールの適用により単語単位で分かち書きした。英文については、空白文字を境界として単語を切り出した。また各フィールド境界にフィールドの名前がわかる区切り記号を挿入した。

[2]正規化では原則として「,(カンマ)」等の記号類を無視し、英文字の大文字をすべて小文字に変換した。

次に、3.2 で述べた方法にしたがって、サフィックスアレイ上で以下の条件を満足する書誌グループを書き出した。(i) 双方のデータベースのデータを少なくとも 1 ずつつ以上含む、(ii) グループ内の書誌数が 4 以下、(iii) 少なくとも 4 単語以上が連続して一致する。(iv) 全体で 12 単語以上が（順番を考慮して）一致する。ここでの基本的な考え方は、これらの条件を満足する書誌グループは互いに類似しており、照合の候補となるということである。

続いて方法 (2) では、上記で求めた書誌ペアそれぞれについて、方法 (1) と同じ照合関数を用いて照合スコアを計算し、異なる「雑誌名・巻・号」の対応ごとに集計した。そして、スコアの最大値やスコアの合計値等が一定条件を満足する冊子の対応を「ブロッキングルール」として抽出した。各データセット毎に得られたブロッキングルールの数を表 6 に示す。

表 6: 方法 (2) で獲得したブロッキングルールの数

データセット名	ブロッキングルール数
DB1-DB2	428
DB1-DB3	5,552
DB1-DB4	2,189
DB2-DB3	5,172
DB2-DB4	900
DB3-DB4	38,614

4.5 実験結果

システムの運用において、処理時間の上で問題となるのは、(a) 照合候補の数え上げおよび (b) 人手判定に要する時間である。そこで以下ではこの 2 点に焦点をあてて提案手法を評価する。

(a) 照合候補の数え上げに要する時間

方式 (2) で行った簡易ブロッキングにおいて、各データセット毎に抽出した書誌グループの総数、グループ内の書誌を組み合わせて得られる書誌ペアの総数、および書誌グループの抽出に要した処理時間（実測値）を表 7 に示す。処理時間のうち支配的なのは、サフィックスアレイを読み込むためのポインタのソート時間であり、計算機のメモリやソートアルゴリズムに依存する。表 7 の結果は、Sun Fire V880 (900MHz, メモリ 64GByte) で C プログラム組み込みの qsort 関数を単一 CUP・オンメモリで実行した場合の参考値である。

表 7: 方法 (2) における類似書誌グループの抽出結果

データセット名	類似クラスタ数	類似文書ペア	処理時間 (分)
DB1-DB2	922,683	1,168,521	72
DB1-DB3	731,279	994,272	142
DB1-DB4	683,078	311,951	469
DB2-DB3	886,958	1,228,011	198
DB2-DB4	1,721,709	2,295,823	576
DB3-DB4	3,134,668	5,086,244	591

照合候補抽出について、現行の我々の運用システムでは同定もれを減らすため、高速 XML データベースエンジンを用いる方法を併用しているが [11]、書誌レコード 1 件あたりの候補検索に要する時間は 0.2 秒程度であり、DB3-DB4 の同定候補の数え上げをレコード毎に行うと約 2 週間を要することになる。これに対して、方法 (2) における簡易ブロッキングの処理時間は 10 時間程度と、かなりの高速化が実現されている。なお照合候補の抽出では正規化を含めその他多くの処理を適用しているが、これらの処理は基本的にデータに対して線形であり、運用においても深刻なボトルネックとはなっていないためここでは省略する。

(b) 人手判定に要する時間

人手判定に要する時間は、判定者の熟練度、判定用ツールの画面デザイン、同定書誌の照合可・不可、書誌の言語等によりばらつきがあるが、概算で 10,000 書誌あたり 30~60 のべ人時間程度である。したがって，効率の上からは、要判定データの数をなるべく低く抑えることが第一の目標となる．表 8 に、方法 (1)（上段）および方法 (2)（下段）を用いて書誌ペアの同定候補を抽出した場合の、「照合可」「要判定」それぞれの書誌ペア数、および人手判定結果による正解数をまとめる。ここで、方法 (2) は方法 (1) における「照合可」「要判定」書誌を取り除いたデータに対して適用しており、両者に重複はない。すなわち両者を合計した値が、本稿の提案手法による最終的な同定候補数である。

比較のため，方法 (2) において、簡易ブロッキングの結果得られた書誌ペアをそのまま照合候補とみなして、照合スコアに基づき「照合可」「要判定」「照合不可」を自動判定する場合の最善の「要判定」データ数を表 9 に示す。最善性能を求めるために、「照合不可を照合可とみなす」 照合誤りの上限値を 1%、

表 8: 方法 (1)(2) による照合候補抽出数のまとめ

データセット名	「照合可」候補数（うち正解数）	「要判定」候補数（うち正解数）
DB1-DB2(1)	71,856 (71,856)	4,561 (4,268)
DB1-DB2(2)	14,195 (14,185)	867 (839)
DB1-DB3(1)	216,155 (216,155)	14,525 (9,744)
DB1-DB3(2)	38,558 (38,530)	4,949 (3,700)
DB1-DB4(1)	299,356 (299,355)	13,863 (13,454)
DB1-DB4(2)	15,083 (14,715)	1,886 (171)
DB2-DB3(1)	155,375 (155,325)	16,180 (12,395)
DB2-DB3(2)	25,475 (24,056)	1,340 (819)
DB2-DB4(1)	179,920 (179,918)	13,134 (12,809)
DB2-DB4(2)	9,292 (8195)	625 (301)
DB3-DB4(1)	1,071,159(1,068,490)	96,473 (83,036)
DB3-DB4(2)	187,592 (185,052)	12,905 (11,492)

「照合可を照合不可とみなす」照合もれの上限値をを 5%と緩く設定し（目標値の 10 倍)、仮に照合スコアの閾値が最適であったとした場合のペア数を計算した。表 8 の下段と表 9 の要判定候補数の違いがブロッキングルール適用による候補数削減の効果である。

なお、運用システムでは XML データベースエンジンを用いる併用方式について、現在人手判定を進めているところであるが、DB2-DB4 のデータをサンプルとして調査した範囲では、本手法による候補の抽出もれは目標値 0.5%以下であることが観察されている。

表 9: ブロッキングを行わない場合の要判定データ数

データセット名	候補書誌ペア数	要判定数 （正解数）
DB1-DB2	1,168,521	151,476 (147,953)
DB1-DB3	994,272	89,801 (48,566)
DB1-DB4	311,951	58,299 (20,890)
DB2-DB3	1,228,011	110,423 (34,736)
DB2-DB4	2,295,823	301,418 (229,099)
DB3-DB4	5,086,244	525,933 (261,847)

5 おわりに

本稿では、複数のデータベース間でのレコード照合のために高速で効率的なブロッキング方法を提案し、実際のデータベースに適用した結果を報告した。

本稿における実験では、「冊子」という物理的にも独立した単位でブロッキングを行ったため、ルール抽出の効果が顕著であったと考えられる。しかしながら、このことは必ずしも提案手法の適用範囲を限

定するものではなく、むしろブロッキングの単位をうまく設計することが作業効率の面から特に重要であり、データ照合問題の要となることを示すものとして強調しておきたい。今後は、引用文献など、より一般的なデータを対象とする方法について検討を進める予定である。

レコード照合はデータクリーニングのための基本的な要素技術であり、その照合結果からは、より複雑なデータ統合処理に役立つ資源の構築が期待できる。具体的な例として、今回の照合結果から、人名における一般的な誤記の対応表を作成した結果の一部を表 10 に示す。対応表の作成にあたっては、公開されている異体字辞典[3] を参照して「異体字」を除いた上で、辞書には登録されていないが間違いやすい人名中の漢字（平仮名を除く）の頻度上位２０件を選んだ。このように、大量の照合済レコード対を用いれば、充実した別名・別字対応表を頻度情報とともに自動生成することができ、さらに照合スコアの計算へとフィードバックすることが可能となる。今後は、本稿の実験では経験的に定めた各閾値の調整に機械学習を適用するとともに、自動生成した対応表による照合関数の改善や半構造化データ解析への適用 [13] についても検討を進めて行きたい。

表 10: 間違いやすい人名中の漢字の抽出例

1	己	巳	11	東	藤	21	浩	治
2	地	池	12	佐	左	22	知	和
3	男	夫	13	掘	堀	23	通	道
4	健	建	14	磯	礒	24	巳	已
5	亨	享	15	次	治	25	修	脩
6	管	菅	16	政	正	26	康	泰
7	治	冶	17	俊	敏	27	諭	論
8	宜	宣	18	孝	考	28	昌	晶
9	太	大	19	荻	萩	29	紀	起
10	昭	明	20	理	里	30	州	洲

謝辞

本研究は、国立情報学研究所が事業サービスを開始している文献情報ナビゲータ（CiNii）開発の一環として行いました。研究開発および運用に携わる富士通研究所、富士通株式会社、国立情報学研究所開発事業部前アプリケーション課の小陳氏、大綱氏およびご関係の方々に感謝いたします。

[3] http://homepage1.nifty.com/kotobukijirushi/ddt.html

参考文献

[1] H. B. Newcombe, J. M. Kennedy, S. J. Axford and A. P. James: “Automatic Linkage of Vital Records,” Science, 130 (3381), pp. 954-959 (1959).

[2] E. Rahm and H. H. Do: “Data Cleaning: Problems and Current Approaches,” IEEE Trans. on Data Engineering, 23 (4), pp. 3-13 (2000).

[3] L. Gu, R. Baxter, D. Vickers and C. Rainsford: “Record Linkage: Current Practice and Future Directions,” CMIS Tech. Rep., CSIRO Mathematical and Information Sciences, 03/83 (2003).

[4] 相澤, 高須, 大山, 安達「異種データベース間でのレコード照合に関する研究動向」NII ジャーナル, No.8, pp. 43-51 (2004).

[5] M. A. Jaro: “Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida,” J. of the American Statistical Society, 84 (406), pp. 414-420 (1989).

[6] M. A. Hernandez and S. J. Stolfo: “Real-world Data is Dirty: Data Cleansing and the Merge/Purge Problem,” J. of Data Mining and Knowledge Discovery, 1 (2) (1998).

[7] A. E. Monge and Charles P. Elkan: “An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records,” Proc. of the ACM-SIGMOD Workshop on Research Issues on Knowledge Discovery and Data Mining (1997).

[8] P. Christen and T. Churches: “Febrl – Freely Extensible Biomedical Record Linkage,” Computer Science Technical Reports, TR-CS-02-05, Australian National University (2002).

[9] A. McCallum, K. Nigam and L. H. Ungar: “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching,” Proc. of ACM-KDD2000, pp. 169-178 (2000).

[10] W. W. Cohen and J. Richman: “Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration,” Proc. of ACM-KDD2002, pp. 475-480 (2002).

[11] 大山, 相澤, 後藤田, 小陳, 大綱,「論文情報ナビゲータの構築」情報処理学会研究報告, 情報学基礎研究会, FI75（2004）(掲載予定).

[12] 国立情報学研究所「CiNii Home（NII 引用文献情報ナビゲータ） 」 <available from http://ci.nii.ac.jp/> (2004).

[13] 岡田, 高須, 安達「SVM/HMM による引用文献データの同定」, 情報処理学会研究報告, 情報学基礎研究会, FI74 (2004).