

フィードバックを利用した文書の選択に基づくレコード抽出手法

張建偉[†], 黒川沙弓[†], 石川佳治[†], 北川博之^{†,††}

近年大量の文書データからのレコード抽出の研究が行われている。レコード抽出には次の課題が存在する。1) 大量の文書データを情報抽出の対象とした場合に多大な処理コストがかかる。2) 抽出されたレコードは、必ずしもユーザが興味あるトピックと合致しないこともある。本稿ではこれに対し、ユーザの意図に合った情報を効率よく抽出するためのレコード抽出手法を提案する。本手法では、効率的な抽出のために、情報を含んでいる可能性の高い文書群を特定する。また、抽出結果に対しユーザからのフィードバックを受けることで、内容の関連が深い情報を抽出することを目指す。

Record Extraction Based on User Feedback and Document Selection

Jianwei Zhang[†], Sayumi Kurokawa[†], Yoshiharu Ishikawa[†], Hiroyuki Kitagawa^{†,††}

In recent years the research of record extraction from large document data is becoming popular. However there still exist some problems in the record extraction. 1) In the case that large document data are used for the target of information extraction, the process of information extraction usually becomes very expensive. 2) It is also possible that extracted records may not pertain to the user's interest on the aspect of the topic. To address these problems, in this paper we propose a method to efficiently extract those records whose topics agree with the user's interest. To improve the efficiency of information extraction systems, our method identifies documents from which useful records may likely be extracted. We also make use of user feedback on extraction results to find topic-related documents and records.

1 はじめに

近年様々な情報発信手段の発達により、電子化されたテキストデータが急激に増加している。このようなテキストデータには有用な情報がよく含まれている。しかし、これらの情報は構造化されていず、計算機は容易に扱うことができない。そのため、大量かつ異種性の高い文書データからの情報抽出 (information extraction) の研究が近年重要視されている。これら

の研究はウェブページやテキストデータにおける文書構造やテキストのパターンなどを分析することで、情報抽出の実現を図る。そのうち、ブートストラッピング型と呼ばれる情報抽出手法が注目を浴びている [1, 2]。これは抽出パターンと抽出レコードを交互に繰り返し学習することにより、少数のシードから大量のレコードを抽出する手法である。直感的には、レコードとは1ないし複数の属性からなるデータで、各属性には同種のデータが含まれるようなものをいう。本稿では、このような構造化されたレコード構造の情報を抽出しようとする試みに着目する。抽出されたレコードの集合は一種のデータベースと考えられ、既存のデータベースとの統合との活用など、様々な形で応用することが可能となる。

これまで提案されている手法には二つの問題点がある。一つ目の問題は、テキストデータは一般的に膨

[†]筑波大学大学院
システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science,
Graduate School of Systems and Information Engineering,
University of Tsukuba
[†]名古屋大学情報連携基盤センター
Information Technology Center,
Nagoya University
^{††}筑波大学計算科学研究センター
Center for Computational Sciences,
University of Tsukuba

大の量があり、このような大量のデータを情報抽出の対象とした場合に多大な処理コストがかかるということである。[3]によれば、「最先端の情報抽出システムは一つの平均サイズの新聞記事を処理するために9秒以上を必要とする。結果として、135Kの文書アーカイブに対し15日間を必要とする。」とされている。二つ目の問題は、抽出されたレコードは、必ずしもユーザが興味あるトピックと合致しないことである。例えば、IT関係の会社と本部場所のペアを抽出しようとする場合、単なるパターンマッチングでは例えば自動車関係の情報が抽出される可能性もある。

本研究では、これに対し、ユーザの意図に合った情報を効率よく抽出するためのレコード抽出手法を提案する。一般的に、大量のデータの中で一部の文書のみが情報抽出タスクに関連している。そこで、本手法では効率的な抽出のために、情報を含んでいる可能性の高い文書群を特定し、選択された文書のみをレコード抽出の対象とする。本研究では、レコード抽出の対象とした文書が索引付けられ、ブール検索が可能であることを想定している。そこで本手法では、文書を検索するためのブール検索クエリを学習する。これらのクエリでユーザが興味あるレコードを含む可能性の高い文書を検索することを目指す。トピックに適合する文書を見つけ、そこからトピックの正しいレコードを抽出するために、本手法はユーザからのフィードバックを利用する。システムは抽出されたレコードに対しユーザからの指摘をもらい、より適切なクエリの生成を試みる。

本論文では、まず、2章で関連研究について述べ、3章においてシステムの概要を説明する。次に、4章においてレコード抽出手法を説明する。そして、5章において文書選択を行う手法について述べる。その後、6章において利用可能なクエリ生成手法について検討する。最後に、7章においてまとめと今後の課題を述べる。

2 関連研究

情報抽出手法の一種として、抽出パターンを予め作成する手法[4]や正解付き文書から帰納学習により抽

出パターンを生成する手法[5]があった。人手作業が必要とするこれらの手法に対して、ブートストラッピング型の手法が提案されている[1, 2, 6]。

DIPRE (Dual Iterative Pattern Relation Extraction) [1] はレコード集合を抽出するためのアプローチであり、HTML文書からの情報抽出のために開発されたものである。例えば、いくつかの本の著者とタイトルのペアがユーザにより与えられると、DIPREは収集されたHTML文書集合の中から与えられたレコードを見つけるためのパターン集合を抽出し、今度はそれらのパターンを用いて新たなレコードの抽出を図る。このようなプロセスを繰り返すことで、HTML文書集合から多くのレコード集合を抽出しようと試みる。特にウェブ環境では、関連するレコードがHTML文書に一定の文脈で繰り返して現れる傾向があるため、この手法は単純であるがうまく機能するといわれる。[6]では、DIPREの繰り返し処理回数を減らすためのレコード抽出手法を提案している。この手法は1回のレコード抽出処理を実行した後、文書集合全体からレコードが抽出できなくなるまで抽出可能なレコード数を予測する。すでに抽出されたレコードの割合(カバー率)を計算し、閾値に達成したら繰り返し処理を終了する。Snowball[2]はプレーンテキスト文書からレコードを抽出する手法である。この手法はDIPREよりも柔軟なパターンを定義し、より厳密なパターンとレコードの評価基準も提案している。パターンには、ベクトルや固有表現(組織名、地名など)も利用している。文字列がきっちり一致するかどうかを判断するのではなく、ベクトルであらわしたパターンと入力文との類似度を計算することで、ある程度の記述の異なりを許容する。パターンの評価は、抽出されたレコード数(カバレッジ)やレコード同士に矛盾があるか(選択性)を考慮し、パターンの確信度を計算する。パターンの確信度を用いてレコードの確信度を計算し、不確かなレコードは今後の抽出に利用されない。後述のように、本研究ではDIPREのパターンを改良し、プレーンテキスト文書に適用する。パターンにはSnowballで利用した固有表現も利用する。

QXtract[3] は情報抽出システムの効率を上げるための文書検索手法を提案している。この手法は機械学習や情報検索の技術によりサンプル文書集合からクエリを学習する。生成されたクエリでレコードを抽出できるような文書を検索することを図っている。しかし、彼らの手法は訓練データの作成においてレコードを抽出できるか否かしか考えていない。本研究は、文書のトピックの正しさも考慮し、ユーザの意図と合致したレコードを抽出することを目指している。また、QXtractの研究では、検索用のクエリは一回しか学習されない。学習の段階が終了すると、クエリは不変である。本研究は、レコード抽出と文書選択を交互に行い、レコード抽出結果とそれに対しユーザからのフィードバックにより、次のレコード抽出用の文書を選択する。クエリはこのような繰り返し処理の中で動的に変化しうる。これにより、より適当な文書とレコードを発見することを期待できる。

3 システムの概要

図1をもとに、想定するシステムの構成について述べる。

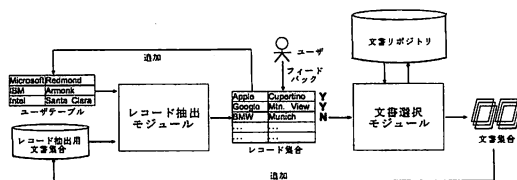


図1: システムの構成

Fig. 1: System Architecture

まず、ユーザが例示レコードを提供する。例えば、あるユーザがIT関係の会社と本部の場所の興味を持っている場合、いくつかのサンプルデータを与えることが考えられる。ユーザが与えた例示レコードは、ユーザテーブル (user table) に収められる。システムによるその後のレコードの追加も、このテーブルが対象となる。

レコード抽出用文書集合 (record extraction document set) は文書リポジトリの部分集合であり、ユーザが興味あるレコードを含む可能性が高い文書からなる。文書リポジトリとは、大量の文書データの全体のことを指す。本研究ではプレーンテキスト文書の利用を想定しており、例えばニュース記事の集合が例として考えられる。なお、文書リポジトリに対しては索引が付けられており、ブール検索により検索できると想定する。システムは文書リポジトリ全体ではなく、そのうちのごく一部であるレコード抽出用文書集合をレコード抽出の対象とする。初期のレコード抽出用文書集合は例示レコードを用いて文書リポジトリから検索する。システムにより文書リポジトリから選ばれた文書はレコード抽出用文書集合に追加する。

レコード抽出モジュール (record extraction module) は、レコード抽出用文書集合にある文書からのレコード抽出を行う。本研究では、この部分にブートストラッピング型の抽出手法の利用を想定している。ユーザから与えられた例示レコードを抽出するためのパターンを学習し、パターンをもとにレコード抽出用文書集合から新たなレコードの抽出を図る。このようにパターンの学習とレコードの抽出を繰り返し行うことでユーザテーブルの内容を増やす。詳細については4章で述べる。

次に新たなレコードに対し、ユーザがフィードバック (feedback) をかける。レコード抽出モジュールの出力として、数多くのレコードが抽出されることが考えられる。ユーザがすべてのレコードを判断するのは非現実的である。そのため、ユーザに見せるレコードを選択し、ユーザの手間を減らす必要がある。ユーザのレコードに対しての判断結果をもとに文書を分類するため、数多くの文書に出現したレコードをフィードバックしてもらうことは有利と考えられる。したがって、一つの手法として、新たなレコードを出現した文書数の降順でソートし、上位 k 件をユーザに見てもらう。ユーザは提示されたレコードに対し、レコードのトピックが正しいかどうかを判断する。例えば、IT関係の会社と本部場所の情報が求められている場合、ほかの業種の情報が不適合であるため、適切なフィードバックを行う。

文書選択モジュール (document selection module) は、ユーザのフィードバックの情報を受け取り、文書リポジトリから今後のレコード抽出用の文書を選ぶ。まず、ユーザが判断したレコードをもとに、レコードの出現した文書を見つけ出す。次に、フィードバックの結果をもとに、これらの文書を分類する。その後、分類の情報を用いてレコード抽出用の文書を検索するためのクエリを学習する。生成されたクエリで文書リポジトリから文書を検索し、レコード抽出用文書集合に追加する。レコード抽出用の文書が動的に追加され、有用な文書のみをレコード抽出モジュールに与える点が本手法の特徴となっている。

システムはレコード抽出と文書選択の処理を繰り返すことで、レコード抽出の効率の向上を図っている。

4 レコード抽出手法

本研究では、レコード抽出にブートストラッピング型の手法を利用する。DIPRE[1] のパターンを改良し、プレーンテキスト文書に適用する。パターンには Snowball[2] で利用した固有表現も利用する。

処理のステップは以下ようになる (図 2)。

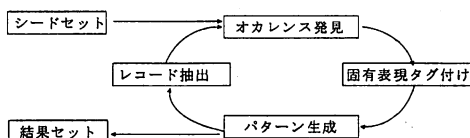


図 2: レコード抽出処理の流れ

Fig. 2: Flow of Record Extraction

1. シードとなるレコード集合 (図 1 のユーザテーブルの例) が与えられる。
2. レコード抽出用文書集合から、シードレコード集合に対応するレコードのオカレンス (occurrence) の集合を見つける。オカレンスは

(*company, address, prefix, tag1, middle, tag2, suffix*)

という形式で表される。 *company* は抽出対象のレコードの会社の名前であり、 *address* は会社の本部の場所である。例えば図 1 のユーザテーブル例の場合、 *company* が Microsoft に、 *address* が Redmond に相当することになる。 *prefix* と *suffix* はそれぞれ、最初および最後に出てくる文字列や記号などのパターンである。 *middle* は属性間の区切りのパターンに相当し、図 1 の場合は *middle* が会社と場所 (あるいは場所と会社) を区切るパターンを保持することになる。 *tag1* と *tag2* は固有表現タグであり、この例の場合は、 ORGANIZATION や LOCATION の値をとる。

3. 発見されたオカレンスの集合をもとにパターン集合を生成する。パターンは

(*prefix, tag1, middle, tag2, suffix*)

の形式を持つ。パターン生成においては、まず、オカレンス集合を同じ *tag1* と *middle* と *tag2* を持つオカレンスごとにグループ化する。含まれるオカレンスの数が 1 件しかないグループは削除し、残りの各グループについてパターン生成を試みる。

パターン生成においては、グループ内のすべてのオカレンスについて、 *prefix* の最長接尾辞、 *suffix* の最長接頭辞を抽出し、それぞれをパターンの *prefix, suffix* とする。

4. 追加されたパターン集合が得られると、これをもとにレコード抽出用文書集合を再度スキャンし、新たなレコードの抽出を試みる。そして、抽出されたレコードをシードレコード集合に追加する。このような処理を繰り返すことで、逐次的にレコードを抽出する。

DIPRE や Snowball では、静的な文書リポジトリを対象としていた。本研究においては、レコード抽出用文書集合が新たに追加されることを想定している。

そのため、レコード抽出モジュールは、新たな文書からレコードのオカレンスの抽出を試みると、新しいパターンで既存の文書を探索しなおす必要がある。

なお、レコード抽出モジュールは、各レコードとそのレコードを抽出した文書の対応関係を表す情報も記録しておく。例えば、図3のようなテーブルが提供されることが考えられる。

Record	Document
(Apple, Cupertino)	1
(Apple, Cupertino)	2
(Apple, Cupertino)	4
(Google, Mtn. View)	1
(Google, Mtn. View)	4
(BMW, Munich)	1
(BMW, Munich)	3
⋮	⋮
⋮	⋮

図 3: 抽出結果の例

Fig. 3: Example of Extraction Result

5 文書選択手法

文書選択モジュールはレコード抽出用の文書集合を文書リポジトリから選ぶ。情報抽出のタスクに対し、三種類の文書が考えられる。

- D₁: レコードが抽出できる、しかもトピックが適合である。
- D₂: レコードが抽出できるが、トピックが不適合である。
- D₃: レコードが抽出できない。

本研究では、D₁ の文書が抽出に有用な文書であると考え、以下ではこのような文書を適合文書 (relevant document) と呼ぶ。D₂ と D₃ の文書はユーザの興味またはレコードの抽出にふさわしくないと考えられ、非適合文書 (non-relevant document) と呼ぶ。

前述のように文書リポジトリは索引付けられているため、適切なクエリがあれば、文書リポジトリから

適合文書を検索することが可能である。そこで、クエリをどのように生成するかが重要なポイントとなる。本研究では、ユーザからのフィードバックをもらい、クエリ生成用の訓練文書を自動的に見つけ出し (5.1 節)、分類する (5.2 節)。次に分類された文書をもとに情報検索や機械学習の手法を用いてクエリを生成する (5.3 節)。その後、これらのクエリで適合文書を検索する (5.4 節)。図4に文書選択モジュールの概要を示す。

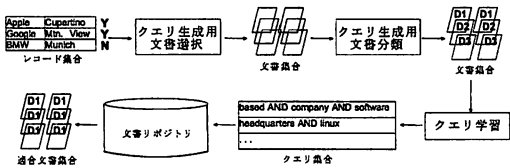


図 4: 文書選択の流れ

Fig. 4: Flow of Document Selection

5.1 クエリ生成用文書の選択

レコード抽出モジュールの出力としてのレコード集合の中から、出現した文書数の多いレコードの上位 k 件をユーザに提示し、フィードバックをもらう。例えば、図3のレコード (Apple, Cupertino), (Google, Mtn. View), (BMW, Munich) がユーザに提示され、それぞれにトピックが適合 (Y), 適合 (Y), 不適合 (N) と判断されている (図5)。

Record	Feedback
(Apple, Cupertino)	Y
(Google, Mtn. View)	Y
(BMW, Munich)	N

図 5: フィードバック結果

Fig. 5: Feedback Result

これらのレコードをもとに抽出した文書 1, 2, 3, 4 を見つけ出すことになる。各文書に対し、そこから抽出したレコードおよびレコードに対するフィードバックの結果が図6のようになる。これらの文書は

レコードを抽出できる文書である。また、レコード抽出用文書集合からレコードを抽出できなかった文書を特定し、その中から一定数の文書をランダムに取り出す。

Document	Record	Feedback
1	(Apple, Cupertino)	Y
1	(Google, Mtn. View)	Y
1	(BMW, Munich)	N
2	(Apple, Cupertino)	Y
3	(BMW, Munich)	N
4	(Apple, Cupertino)	Y
4	(Google, Mtn. View)	Y

図 6: 選択した文書の例

Fig. 6: Example of Selected Documents

5.2 クエリ生成用文書の分類

各文書のトピックが適合するかどうかを判断するために、その文書から抽出したレコードのフィードバック結果を用いている。基本的なアイデアとしては、ある文書から抽出したレコードのうちでトピックが適合とユーザに判断されたレコードの割合が高ければ、その文書のトピックが適合と考える。そこで、適合文書集合 D_1 は

$$D_1 = \{d_i \mid |R(d_i)| > 1 \wedge \frac{|R_{positive}(d_i)|}{|R(d_i)|} > \tau_1\} \quad (1)$$

のような文書集合である。なお、 $|R(d_i)|$ は文書 d_i から抽出したレコードの数であり、 $|R_{positive}(d_i)|$ はそのうちトピックが正しいとユーザに判断されるレコードの数である。 τ_1 は予め与えられた閾値である。

同様に、非適合文書集合 D_2 と D_3 は下記の式で定義される。

$$D_2 = \{d_i \mid |R(d_i)| > 1 \wedge \frac{|R_{positive}(d_i)|}{|R(d_i)|} < \tau_2\} \quad (2)$$

$$D_3 = \{d_i \mid |R(d_i)| = 0\} \quad (3)$$

図 6 の例に対し、 $\tau_1 = 0.6$ 、 $\tau_2 = 0.2$ の場合、文書 1 について、3 個のレコードが抽出でき、2 個のレコードのトピックが正しいと判断されるため、文書 1 を適合文書 D_1 に帰属する。同様に、文書 2 と文書 4

を適合文書 D_1 に、文書 3 を非適合文書 D_2 に帰属する。レコードを抽出できなかった文書も見つけ出し、非適合文書 D_3 に帰属する。

以上のように、クエリ生成用の文書は適合文書 D_1 、非適合文書 D_2 、非適合文書 D_3 三種類の文書からなる。

5.3 クエリ生成

前節のように自動的に分類された適合文書と非適合文書を訓練文書データとし、クエリを学習する。学習のために、適合文書と非適合文書内の単語を特徴として扱う。この節では、訓練文書データからクエリの生成手法について述べる。5.3.1 節において、情報検索の手法を説明する。5.3.2 節においては、機械学習の手法について述べる。

5.3.1 Okapi

Okapi[7] は適合文書集合と非適合文書集合を利用したクエリ拡張 (query expansion) の標準的手法として知られている。本研究はこの手法をクエリの生成に適用する。Okapi の手法では、与えられた適合文書の中に出現する各単語について下記のスコアの計算を行う。

$$w_t = \left(\frac{r_t}{R} - \frac{n_t - r_t}{N - R}\right) \log \frac{r_t / (R - r_t)}{(n_t - r_t) / (N - n_t - R + r_t)} \quad (4)$$

ここで、 r_t は単語 t を含む既知の適合文書数、 n_t は単語 t を含む文書数、 R は既知の適合文書数、 N は訓練文書データ内の文書数である。直感的に、単語 t のスコアは多くの適合文書に出現するほど高くなり、多くの非適合文書に出現するほど低くなるという性質がある。

本研究では、文書からレコードを抽出できるかどうかと、文書のトピックが正しいかどうか、という二つの観点から二つの単語リストを作る。レコードを抽出できるかどうかを学習するために、適合文書 D_1 を正例とし、非適合文書 D_3 を負例とし、適合文書 D_1 にある単語について式 (4) でスコアを計算し、一つの順位付けの単語リストを生成する。例えば、{based, spokesperson, ...} のような単語リストが得られる。

トピックが正しいかどうかを学習するために、適合文書 D_1 を正例とし、非適合文書 D_2 を負例とし、もう一つの単語リストを作る。例えば、{software, linux, ...} のような単語リストが生成される。それで、二つの単語リストからそれぞれスコアの高い単語を上位 n 個選び、組み合わせでクエリを生成する。前述の例に対して、{(based AND software), (based AND linux), (spokesperson AND software), (spokesperson AND linux), ...} のようなクエリ集合が生成される。

5.3.2 Slipper

Slipper[9] はルール学習システムの一つである。2クラスの訓練データから、分類のルールを学習できる。“bag of words”の形式で表した2クラスの文書を入力とした場合、文書分類のルールが出力される。例えば、“based AND company $\rightarrow$ relevant”のようなルールが帰納できる。これは、文書に based と company の単語が含まれていれば、文書が適合だということを意味している。ルールの精度と呼ばれる、ルールで正しく判断される文書の割合もシステムの出力である。生成されるルールの精度の降順でルールを並べ替え、上位に順位付けられるルールをクエリに変更する。例えば、上記のルールの例に対し、“based AND company”のクエリが生成される。5.3.1 節と同様に、適合文書 D_1 と非適合文書 D_3 を訓練データとして、レコードを抽出できるかどうかを学習し、適合文書 D_1 と非適合文書 D_2 を訓練データとして、トピックが適合かどうかを学習する。このように二つのルール集合を生成し、組み合わせでクエリを生成する。

5.4 レコード抽出用文書の検索

クエリが生成された後、クエリでレコード抽出用文書を文書リポジトリから検索する。文書リポジトリは索引付けられており、前節で生成されたクエリに対応できる。検索してくるレコード抽出用文書の総数と各クエリで検索する文書数はシステムのパラメータであり、適宜設定する必要がある。検索した文書はレコード抽出用文書集合に追加される。

6 議論

他の二つのクエリの生成手法について検討する。

6.1 適合性フィードバック

5.3 節で述べた手法で生成されたクエリは索引語を AND で結合した論理式の形式である。論理式を満たす索引付けられた文書を検索の対象とする。これは検索処理にブール検索モデルの利用を仮定している。一方では、文書リポジトリ内の文書をベクトルで表現する場合、適合性フィードバック (relevance feedback) [8] のアプローチを用いて、ベクトルで表現されるクエリを生成することも考えられる。ベクトルの各要素は索引語の重みである。

本来の適合性フィードバックは、ユーザと情報検索システムが対話することにより、クエリをユーザの意向の方向に修正する。一つのアプローチとしては、*Rocchio* の式と呼ばれるものがある。

$$q' = \alpha q + \frac{\beta}{|R|} \sum_{d_i \in R} d_i - \frac{\gamma}{|N|} \sum_{d_i \in N} d_i \quad (5)$$

$|R|, |N|$ は、それぞれ、適合文書集合 R 、非適合文書集合 N の文書数である。 α, β, γ は定数であり、一般に適合文書が非適合文書よりも重要であるため、 γ は β よりも小さく設定する。

適合文書 D_1 を R として、非適合文書 D_2 と非適合文書 D_3 を N として、式 (5) でクエリを生成し洗練することも考えられる。初回のレコード抽出モジュールを実行した後の時点で、初期クエリ q の各要素は 0 である。 q' は適合文書と非適合文書の情報を用いて生成する。2 回目以後の文書選択では、適合性フィードバックの計算式 (式 (5)) で前回のクエリを修正する。

6.2 SVM

ユーザの評価できるレコード数とそれをもとに見つけ出せる文書数は有限であると考えられる。そこで、数少ない訓練データからの学習に適していると言われている SVM の手法の利用も考えられる。SVM (Support Vector Machine) [10] は 2 クラスの分類を

行う学習手法であり、訓練データより最適分離平面を決める。クエリの生成に利用する場合、訓練データを用いて適合文書に分類できるような単語の最小集合を学習する [11]。結果として、SVMはSlipper (5.3.2節) と同様なルール集合の出力が得られる。

7 まとめと今後の課題

本稿では、ユーザのフィードバックを利用したレコード抽出用の文書を選択することによる効率的なレコード抽出手法を提案した。システムはユーザのフィードバックに応じて訓練データを自動的に生成し、クエリを自動的に学習する。クエリは適合文書の特徴を捉え、適合文書を検索することを期待している。文書リポジトリ全体ではなく、適合と予測される文書のみをレコード抽出に利用することでレコード抽出システムの効率の向上を目指している。

今後の課題としては以下の点が挙げられる。

1. システムの実装と実験：提案したシステムをもとに実装を行い実験を行う。そのためには、手法の詳細化が必要となる。
2. クエリ生成手法の比較：いくつかのクエリの生成手法を提案したが、各手法の効果を比較する必要がある。各手法で生成されるクエリが重なっているかなどについて、検証する必要がある。

謝辞

本研究の一部は、日本学術振興会科学研究費基盤研究 (C)(16500048)、科学研究費補助金特定領域研究 (18049005) 及び科学技術振興機構 CREST「自律連合型基盤システムの構築」による。

参考文献

- [1] S. Brin, Extracting Patterns and Relations from the World Wide Web. *Proc. WebDB*, 1998.

- [2] E. Agichtein and L. Gravano, Snowball: Extracting Relations from Large Plain-Text Collections. *Proc. ACM SIGMOD*, 2001.
- [3] E. Agichtein and L. Gravano, Querying Text Databases for Efficient Information Extraction. *Proc. ICDE*, pp. 113–124, 2003.
- [4] R. Baumgartner, S. Flesca and G. Gottlob, Visual Web Information Extraction with Lixto. *Proc. VLDB*, pp. 119–128, 2001.
- [5] N. Kushmerick, Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, Vol. 118, No. 1-2, pp. 15–68, 2000.
- [6] R. Y. Zhang, L. V.S. Lakshmanan and R. H. Zamar, Extracting Relational Data from HTML Repositories. *SIGKDD Explorations*, 2004.
- [7] S. E. Robertson, Overview of the okapi projects. *Journal of the American Society for Information Science*, Vol. 53, No. 1, pp. 3–7, 1997.
- [8] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.
- [9] W. W. Cohen and Y. Singer, A Simple, Fast, and Effective Rule Learner. *Proc. AAAI/IAAI*, pp. 335–342, 1999.
- [10] T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proc. ECML*, pp. 137–142, 1998.
- [11] L. Gravano, P. Ipeirotis and M. Sahami, QProber: A System for Automatic Classification of Hidden-web Databases. *ACM Trans. Inf. Syst.*, Vol. 21, No. 1, pp. 1–41, 2003.