データベース連携による文書情報源からのレコード抽出

張　建偉† 　石川　佳治†† 　北川　博之†,†††

† 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
〒 305–8573 茨城県つくば市天王台 1–1–1
†† 名古屋大学情報連携基盤センター
〒 464–8601 愛知県名古屋市千種区不老町
††† 筑波大学計算科学研究センター
〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †zjw@kde.cs.tsukuba.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp, †††kitagawa@cs.tsukuba.ac.jp

あらまし　大量のテキスト情報源から有用な情報を取得するための情報抽出は近年重要視されている．構造化されていないテキストから抽出した情報は一種のデータベースと考えられる．一方，テキストデータベースの統合はデータ工学における重要な研究課題の一つである．本稿では，既存のデータベースと統合することにより，文書情報源から精度の高い情報を効率よく抽出し，ユーザの意図に合った新たな知識を発見する手法を提案する．

キーワード　情報抽出，レコード抽出，ブートストラップ，文書選択，テキスト DB，情報統合，テキスト結合

Record Extraction from Document Resources Cooperating with Databases

Jianwei ZHANG†, Yoshiharu ISHIKAWA††, and Hiroyuki KITAGAWA†,†††

† Department of Computer Science, Graduate School of Systems and Information Engineering
University of Tsukuba
Tennohdai 1–1–1, Tsukuba, Ibaraki, 305–8573, Japan
†† Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464–8601, Japan
††† Center for Computational Sciences, University of Tsukuba
Tennohdai 1–1–1, Tsukuba, Ibaraki, 305–8573, Japan
E-mail: †zjw@kde.cs.tsukuba.ac.jp, ††ishikawa@itc.nagoya-u.ac.jp, †††kitagawa@cs.tsukuba.ac.jp

Abstract　In recent years, the research of information extraction that acquires useful information from large document data is becoming more and more important. Extracted information from unstructured texts can be considered a kind of database. Besides, the integration of heterogeneous databases, sometimes called data integration is one of the important research topics in the field of data engineering. In this paper, we propose an approach to find the knowledge that agrees with a user's interest by efficiently extracting records with high accuracy and integrating them with existing database.

Key words　information extraction, record extraction, bootstrap, document selection, text DB, data integration, text join

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．大量のテキストデータから有用な情報を自動的に，或いは半自動的に抽出する**情報抽出** (information extraction) [1] の技術がますます重要となっている．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 [2]〜[5]．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．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る．この意味で，適切

な抽出システムがあれば，テキスト文書群を仮想的なデータベースと見なせる．

一方，テキストデータベースの統合に関する研究が盛んに進められている[6]〜[8], [10]．独立に構築された複数のデータベース間での重複レコードを見つけ出す処理は，データ統合に不可欠な技術である．複数のデータベースの情報には，異なるスキーマや表記のゆれなどが存在する．このような場合，標準的なリレーショナルデータベース上の結合演算を近似的な文字列照合機能で拡張することにより，曖昧さを許した**テキスト結合** (text join) の照合処理が求められる．

これらの研究の動向を踏まえ，本研究ではテキスト結合の技術を用いて，既存データベースとの連携により，レコード抽出処理を向上させるための実現手法を提案する．

2. 関 連 研 究

2.1 情 報 抽 出

テキストからの情報抽出にはさまざまなアプローチがある．特に大量のテキストデータを対象とした，人手作業を減らすブートストラッピング型と呼ばれる情報抽出手法が提案されている．

ブートストラッピング型のレコード抽出法の一つに **DIPRE** (Dual Iterative Pattern Relation Extraction) [2] がある．もともとはHTML 文書群からレコード集合を抽出するための手法として提案された，比較的単純な手法である．抽出パターンの生成では，レコードのオカレンスの出現コンテクストだけでなく，レコードが発見されたURL のパターンも分析する点に特徴がある．レコードが抽出されたURL を一般化することにより，レコードを含むと想定されるURL のパターンを生成することで，抽出対象とするHTML 文書の絞込みを可能とする．

Snowball [3] は DIPRE を拡張したアプローチである．DIPRE とは異なり，主として構造化されていないテキストからのレコード抽出を対象としている．特徴の一つは，固有表現（named entity）の抽出システムを利用することである．固有表現を用いた抽出パターンを生成することにより，精度の高い抽出処理を実現する．

QXtract [5] は，これらのレコード抽出法を内部で利用する，メタなレコード抽出法である．特徴は，レコード抽出における処理コストの削減に着目した点である．訓練データをもとに，レコードが抽出できる文書群とできない文書群の特徴を，文書分類手法をもとに抽出する点である．この結果を利用して，レコードが抽出できる可能性が高い文書群を中心に抽出処理を行うことで，比較的少ない処理時間で多くのレコードの抽出を目指している．

本研究では，レコード抽出処理手法として，基本的にはブートストラッピング型の処理に基づいているが，既存データベースの情報を生かして，抽出処理を向上させる点が特徴となる．

2.2 テキストデータベースの統合

異なる情報源の情報を統合するデータ統合の研究はますます重要となっている．特に複数のテキストデータベースが独立に作成された場合，入力ミス，略語，フォーマットの違いなどのことにより，同じエンティティに複数の表記が存在することがある．表記が異なるが，実際は同じエンティティに参照しているレコードを発見するために，曖昧さを考慮したテキストの結合処理技術 [6], [7] などが開発されている．WHIRL [8](Word-based Heterogeneous Information Representation Language) では，テキストの類似性に基づく結合処理を導入した言語とその処理方式を提案している．情報検索の分野で文書間類似度の尺度として用いられる tf-idf 手法 [9] をレコード類似度の計算に適用し，近似的な文字列照合機能を実現している．Gravano らはサンプリングに基づきテキスト結合の効率を上げる結合処理手法を提案している [10]．基本的なアイデアは，サンプリングにより重要ではないトーケンを削除し，結合を行うレコードのペア数を減らすことで，マッチするレコードを効率よく見つける．特徴の一つは，処理がリレーショナルデータベースにおけるSQL 問合せで実現されていることである．RelDC [11](Relationship-based Data Cleaning) は，グラフで表現したエンティティ間の関係を利用し，同じエンティティを発見する手法を提案している．例えば，同じ著者を認識する場合, 著者名のテキスト類似度を計算するだけではなく, 著者と論文の関係も分析して, 同一エンティティの識別問題を解決する．本研究は，文書からなるテキストデータベースとリレーショナルデータベースとの統合を考え，以上の研究と着目点が異なる．

3. システムの概要

3.1 研究の目的

リレーショナルデータベースと大量のプレーンテキスト文書からなるテキストリポジトリがあるとする．リレーショナルデータベースには構造的な情報が保存される．適切な情報抽出システムが構築されれば，テキストリポジトリから構造化されたデータが抽出できると考えられる．ユーザの立場から捉えると，テキストリポジトリも一種の仮想的なデータベースと見なせる．本研究では，従来のリレーショナルデータベースと文書からなるテキストデータベースの統合の問題を考える．

例として，以下のシナリオを考える．ユーザは手元に Company(name, category) というスキーマを持つリレーション（図 1）を保持している．一方，ユーザはテキストリポジトリ中に会社名とその所在地に関する情報があることを知っているとする．そこでユーザは，テキストリポジトリ上に仮想的なリレーション Location(cname, clocation) を作成し，Location のシードレコードとして，図 2 のようなサンプルを入力する．情報抽出システムを用いて，テキストリポジトリから，会社と住所のレコードを追加したいとする．ただし，ユーザはあるカテゴリ（例えば IT 分野）の会社にのみ興味をもっていると設定し，それ以外の会社については所在地の情報を抽出する必要はないとする．

IT 会社の社名と所在地の情報を求めたい場合，ユーザは要求を以下のようなSQL 風の問合せで記述することを想定する．

```
SELECT C.name, L.cname, L.clocation,
FROM Company C OUTER JOIN Location L
WHERE C.category = "IT" AND C.name ~ L.cname
```

問合せの結果は例えば図 3 のようになる．

“∼”という比較は，曖昧なマッチングを表す構文であり，類

Company

name	category
Apple	IT
BMW	AUTO
Banc One	BANK
Canon	IT
Citibank	BANK
⋮	⋮

図 1　既存データベース

Location

cname	clocation
Canon	Tokyo
IBM	New York
Xerox	Stamford

図 2　仮想的なデータベース（ユーザサンプル）

name	cname	clocation
Apple	Apple	Cupertino
Apple	Apple Corporation	Cupertino
Canon	Canon	Tokyo
Canon	Canon Corp.	Japan
⋮	⋮	⋮
	AMD	Sunnyvale
	⋮	⋮

図 3　問合せ結果

似度のスコアが求められ，その値の順で問合せ結果がランク付けされて返されると想定する．類似度のスコアについては tf-idf 方式のベクトル空間モデルの利用が考えられる．ただし，idf については，テキストリポジトリの索引情報の利用やテキストリポジトリ中の文書のサンプリングで求める．閾値がユーザによって与えられるものとし，類似度が閾値を超えるペアが結果になる．

外結合 (OUTER JOIN) は，双方のリレーションにおいて，相手側にマッチする情報が必ずしも存在しない場合でも結合対象となる．特にこれは，テキストリポジトリ側から抽出される新規のレコードを結果に取り込みたいときに有効となる．例えば，テキストリポジトリ中から，ある IT 会社のレコード (AMD, Sunnyvale) が抽出されたとしよう．しかし，“AMD” にマッチするレコードが Company リレーション側になければ，OUTER JOIN なしの場合はこのようなレコードが問合せの結果に含まれなくなってしまう．OUTER JOIN を用いることで，レコード抽出処理で発見した新たなレコードも追加的に取得することができる．

3.2　想定するシステム構成

図 4 に想定するシステムの構成を示す．

あるカテゴリの会社の情報を求めたい場合，テキストリポジトリ全体を抽出処理対象とするのは以下の理由で不適切である．

（1）　大量の文書を対象とする場合，情報抽出処理には非常にコストを要する．

（2）　文書を区別無く抽出対象とする場合，そこからトピックに適合しないレコードが抽出されてしまうことがある．

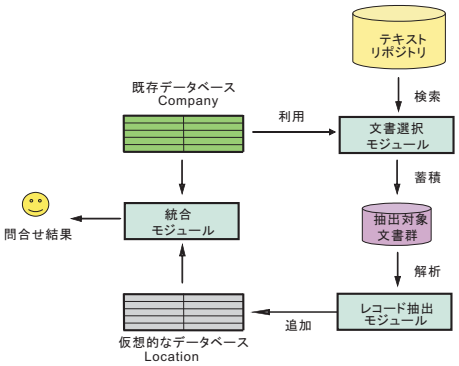

図 4　システム構成

そこで，**文書選択モジュール**は，テキストリポジトリから，抽出に役立つと考えられる一部の文書を取り出し，一時的に蓄積する役割を果たす．本研究では，既存データベースの情報を利用して文書を選択する手法を提案する．これは 4 節で述べる．

初期の時点に Location リレーションはサンプルレコードのみからなり，レコードはほとんど含まれていない状況である．そこで，いかに Location リレーションを拡張するかがポイントになる．本研究では，既存データベースと連携することで，ブートストラッピング型の情報抽出手法を用いて，少数のシードからはじめ，テキストリポジトリから大量のレコードを取得する方式を提案する．**レコード抽出モジュール**はこのような役割を果たす．具体的な処理を 5 節で述べる．

抽出したレコードと既存データベースのレコードがまったく独立に作成されているため，属性に表記のゆれが存在することがある．このような場合に，二つのデータベースは統一的な識別子を持たず，レコードの属性の間に厳密な対応関係が存在しない．通常のリレーショナルデータベース上の結合ではなく，テキスト属性間の曖昧なマッチング処理が求められている．6 節では，既存のデータベースと抽出システムで作られたデータベースの情報を統合する**統合モジュール**を述べる．

4.　文 書 選 択

本研究では，テキストリポジトリがプレーンテキスト文書集合からなることを想定しており，例えばニュース記事の集合が例として考えられる．なお，テキストリポジトリに対しては索引が付けられており，ブール問合せに応じて検索結果が返されると想定する．一般的にテキストリポジトリのうちのごく一部のみがレコード抽出に関連する．本節では，既存データベースの情報を元に検索質問を生成し，レコード抽出用の文書集合の選択手法を述べる．ユーザが興味あるトピックに適合する文書を選び，そこからトピックに合致するレコードを抽出することが目標である．

検索質問の作成について二つの方式を検討する．

(a)　Company リレーションにある IT カテゴリの会社名を検索質問とする．具体的に IT カテゴリの会社名の論理和を検索条件とし，N 件の文書を取得する．これで選ばれた文書に既存データベースの情報が含まれる可能性が高いため，Company リレーションの IT 会社とマッチするレコードの抽出に有利とな

ると考えられるが，テキストリポジトリから Company リレーションに存在しない新たなレコード発見については，漏れが多い可能性がある．

(b) Company リレーションとテキストリポジトリ上の仮想的なリレーション Location にユーザが入力したサンプルの情報を利用して，着目するカテゴリのトピックを表す特徴単語を識別し，それらの単語を検索質問とする．トピックの特徴単語で検索された文書には既存データベースにない会社の情報が含まれている可能性があり，新規レコードの抽出に有利であると考えられる．

次に文書を選択するアプローチ (b) について具体的に述べる．処理のステップは以下のようになる．

（1） **適合文書と非適合文書の選択:** 初期にユーザサンプル（図 2）を用いてテキストリポジトリから文書を検索する．これらの文書に対して，レコード抽出の処理（詳細は 5 節で述べる）を行い，レコードの抽出を試みる．次に Company リレーション中の IT 会社のレコードとのマッチングを行い（マッチング処理は 6 節で述べる），類似しているものがあれば抽出されたレコードを「正しいレコード」と考える．文書集合を抽出した正しいレコード数の降順でソートし，上位 k 件を適合文書とする．また，テキストリポジトリからランダムに適合文書集合と異なる文書を k 件選び，非適合文書集合とする．

（2） **検索質問の学習:** 適合文書集合と非適合文書集合が選ばれた後，**Okapi** [12] の手法を利用して，順位付けの単語リストを作る．Okapi は与えられた適合文書中に出現する各単語について下記のスコアを計算する．

$$w(t) = \frac{r_t/(R - r_t)}{(n_t - r_t)/(N - n_t - R + r_t)} \qquad (1)$$

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，あまり非適合文書に出現しないほど低くなるという性質がある．トピックに適合するレコードに出現した単語（IT 会社の会社名や所在地の単語）やトピックを表す特徴単語（例えば，Computer, workstation, 80386 など）が上位にランク付けされる．スコアの高い n 個の単語の論理和を検索質問として，テキストリポジトリから N 個の文書を取得する．これで取得した文書は着目しているトピックに適合する文書であり，そこから抽出するレコードがトピックに合致する可能性が高いと考えられる．

5. レコード抽出

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 [2] 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 [3] で利用した固有表現も利用する．

処理のステップは以下のようになる（図 5）．

（1） シードとなるレコード集合（図 2 の例）が与えられる．

（2） 選択された文書集合から，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

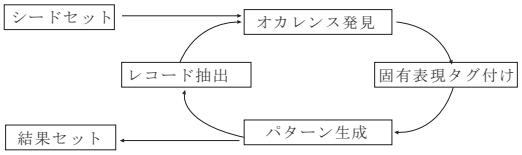

図 5 レコード抽出処理の流れ

$$(company, location, prefix, tag1, middle, tag2, suffix)$$

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，$location$ は会社の所在地である．例えば図 2 のユーザサンプル例の場合，$company$ が Canon に，$location$ が Tokyo 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現する属性（会社や場所）の前および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 2 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．

（5） 抽出したレコードをソートする．一般的に抽出のために用いたパターン数が多いほど，抽出したレコードのノイズが少ないと考えられる．また，トピックを限定した文書集合中に出現回数が多いほど，抽出したレコードの質が信頼できる上，限定されたトピックに関連する可能性が高い．そこで，レコードの抽出を用いたパターン数とレコードの出現した文書数の降順でレコードをソートし，上位の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．

6. 結合処理

抽出処理システムで抽出したレコードの集合を一つのデータベースと見なし，既存のデータベースと統合して，ユーザが求めている情報を答える．まず，レコードの各属性を属性ベクトルとする．属性に出現した各単語を属性ベクトルの一次元に対応付ける．属性ベクトルの各要素は tf-idf 法で付けられた各単語の重みである．あるレコード中での出現頻度が高

い (tf) 単語のうち，他のレコードにはあまり出現しないもの (idf) をそのレコードに特徴的な単語として扱うという考え方である．例えば，*Apple Corporation* の例について，よく出現した単語 *Corporation* に低い重み，*Apple* に高い重みが与えられる．テキスト属性の類似度は，対応する属性ベクトルの内積で計算される．例えば，Company データベース（図 1）の i 番目のタプルについては，2 つの属性ベクトルを要素に持つベクトルを $(\boldsymbol{v}_{name,i}, \boldsymbol{v}_{category,i})$ のように，Location データベース（図 2）の j 番目のタプルに対応するタプルについては，$(\boldsymbol{v}_{cname,j}, \boldsymbol{v}_{clocation,j})$ というタプルをそれぞれ対応付ける．各属性ベクトル値は，属性のテキスト値をもとに作成される．よって，“Apple”〜“Apple Corporation”という比較は，具体的に属性ベクトルの類似度

$$sim(\boldsymbol{v}_{name,1}, \boldsymbol{v}_{cname,j})$$

を計算することで算出する．類似度がユーザによって指定された閾値以上であるペアを類似度の降順でソートし，最後の結果として返される．

また，OUTER JOIN の演算で，抽出処理で上位にランク付けられたレコードも結果リレーションに追加される．

7. 予備実験

予備実験として，ブートストラッピング型の抽出システムを実装し，レコードの抽出の効果について検証した．この実験は予備的な実験であり，今回の提案手法に関する実験は今後の論文に反映する．実験に利用するテキストリポジトリは 1986 年から 1992 年までの Wall Street Journal ニュース記事であり，173,039 件の文書からなる．全文検索システム Namazu [13] を用いて，テキストリポジトリにインデックスを作成する．仮想的なリレーションの初期サンプルは 5 個の IT 企業と所在地のペア（図 6）からなる．適合文書と非適合文書をそれぞれ 250 件選び，Okapi の手法で単語リストを学習した単語リストの上位 30 個の論理和を検索条件とし，テキストリポジトリから 5000 件の文書を取得し，抽出処理を行った．5000 件の文書から，10 個のパターンが生成され，2909 個のレコードが抽出された．パターンを図 7 に示し，抽出レコードの上位 50 個を図 8 に示す．上位 50 個のうち，41 個のレコードが IT 関係の企業である．

8. まとめと今後の課題

本稿では，既存データベースと抽出処理で構築した仮想的なデータベースを統合するシナリオについて述べた．

今後の課題としては以下の点が挙げられる．

（1） 手法の詳細化：本稿で示した内容は処理の流れであるため，それらの処理の詳細化を進める．

（2） システムの実装と実験：提案したシステム構成をもとに実装を行い実験を行う．各部分の効果を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)，文部科学省科学研究費補助金特定領域研究 (18049005)，科学技術振興機構 CREST「自律連合型基盤システムの構築」，柏森情報科学振興財団，及び，セコム科学技術振興財団の助成による．

文　献

[1] E. Agichtein and S. Sarawagi, Scalable Information Extraction and Integration. *Proc. KDD*, 2006 (Tutorial).

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

[3] E. Agichtein and L. Gravano, Snowball: Extracting Relations from Large Plain-Text Collections. *Proc. ACM SIGMOD*, 2001.

[4] R. Y. Zhang, L. V.S. Lakshmanan and R. H. Zamar, Extracting Relational Data from HTML Repositories. *SIGKDD Explorations*, 2004.

[5] E. Agichtein and L. Gravano, Querying Text Databases for Efficient Information Extraction. *Proc. ICDE*, pp. 113–124, 2003.

[6] N. Koudas, S. Sarawagi and D. Srivastava, Record Linkage: Similarity Measures and Algorithms. *Proc. ACM SIGMOD*, 2006 (Tutorial).

[7] 相澤，大山，高須，安達，レコード同定問題に関する研究の課題と現状．電子情報通信学会論文誌，Vol. J88-DI, 3, pp.576–589, 2005.

[8] W. W. Cohen, Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity. *Proc. ACM SIGMOD*, 1998.

[9] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[10] L. Gravano, P. G. Ipeirotis, N. Koudas and D. Srivastava, Text Joins in an RDBMS for Web Data Integration. *Proc. WWW*, 2003.

[11] D. V. Kalashnikov and S. Mehrotra, Domain-independent data cleaning via analysis of entity-relationship graph. *ACM TODS*, June 2006.

[12] S. E. Robertson, Overview of the okapi projects. *Journal of the American Society for Information Science*, Vol. 53, No. 1, pp. 3–7, 1997.

[13] Namazu: http://www.namazu.org/

cname	clocation
Xerox	Stamford
Intel	Santa Clara
Apple	Cupertino
Compaq	Houston
Sun	Mountain View

図 6　シードレコード

1	ORG , LOC
2	ORG , based in LOC
3	ORG of LOC
4	LOC 's ORG
5	ORG , a LOC
6	ORG in LOC
7	ORG , of LOC
8	ORG , which is based in LOC
9	ORG is based in LOC
10	ORG , the LOC

図 7　生成したパターン例

rank	cname	clocation	IT topic
1	Dataquest Inc.	San Jose	Y
2	Intel	Santa Clara	Y
3	Sun Microsystems Inc.	Mountain View	Y
4	Intel Corp.	Santa Clara	Y
5	Sun	Mountain View	Y
6	Tandem Computers Inc.	Cupertino	Y
7	IBM	Armonk	Y
8	NEC Corp.	Japan	Y
9	Apple	Cupertino	Y
10	Toshiba Corp.	Japan	Y
11	Motorola	Schaumburg	N
12	Borland International Inc.	Scotts Valley	Y
13	Hewlett-Packard Co.	Palo Alto	Y
14	Cypress Semiconductor Corp.	San Jose	N
15	Mips	Sunnyvale	Y
16	Hewlett-Packard	Palo Alto	Y
17	Ardent Computer Corp.	Sunnyvale	Y
18	Metaphor	Mountain View	N
19	Acer Group	Taiwan	Y
20	Fujitsu Ltd.	Japan	Y
21	Advanced Micro Devices Inc.	Sunnyvale	Y
22	Digital	Maynard	Y
23	Xerox	Stamford	Y
24	Microsoft	Redmond	Y
25	Siemens AG	Germany	Y
26	Fujitsu	Japan	Y
27	Forrester Research Inc.	Cambridge	Y
28	Apple Computer Inc.	Cupertino	Y
29	International Computers Ltd.	Britain	Y
30	Unisys	Blue Bell	Y
31	Matsushita Electric Industrial Co.	Japan	Y
32	Mips Computer Systems Inc.	Sunnyvale	Y
33	Canon Inc.	Japan	Y
34	Commodity Exchange	New York	N
35	Siemens	West Germany	Y
36	Carnegie Mellon University	Pittsburgh	N
37	Sharp Corp.	Japan	Y
38	Dain Bosworth Inc.	Minneapolis	N
39	Robinson-Humphrey Co.	Atlanta	Y
40	Sun Microsystems	Mountain View	Y
41	Silicon Graphics Inc.	Mountain View	Y
42	Software Publishing Corp.	Mountain View	Y
43	Battelle Memorial Institute	Columbus	N
44	Thinking Machines Corp.	Cambridge	Y
45	National Semiconductor Corp.	Santa Clara	N
46	SoundView Financial Group	Stamford	N
47	Advest Inc.	Hartford	Y
48	Lotus	Cambridge	Y
49	Seagate	Scotts Valley	Y
50	MIPS Computer Systems Inc.	Sunnyvale	Y

図 8　抽出したレコード例