7P−8

簡易型自然言語インタフェース
− キーワード抽出型と事例ベース型の協調 − *

島津秀雄　有田正剛　高島洋典 †

日本電気（株）C&C情報研究所 ‡

1　はじめに

一般の人が、PC-VAN等のパソコン通信を介してオンライン・データベースを使う/使える機会が増えるにつれて、オンライン・データベースに対する「真に」実用的な自然言語インタフェース実現が望まれている。そのような領域依存の自然言語インタフェースを作成する手法の1つとして事例ベース型の解析手法 (Phrase-based Parsing, PBP) がある [4][2][3]。PBPでは、多くの「言語パターン-意味表現」の対がPBPの知識ベース中に格納保持されている。問い合わせ文が与えられると、PBPの知識ベースの中からマッチするパターンを見つけ、そのパターンと対の意味表現を入力文の意味表現として出力する。アプリケーション開発者は、対象領域で出現しそうなすべての言語パターンとその意味表現を前もって定義しておかなくてはならない。しかも、これらのパターン・概念対は、対象領域に非常に依存しているので、新しい自然言語インタフェースを作成するたびに新たに定義しなくてはならないのが、PBPの問題である。

一方、我々は、キーワード抽出型自然言語解析手法 (Keyword-based Parsing, KBP) を開発してきた [1]。KBPは、テーブル型のデータベースが対象であり、問い合わせ文が与えられると、キーワードのみを抽出して、それらからSQL表現を意味表現として生成する手法であり、簡単な構造の問い合わせ文を非常に堅固に解釈することが出来る。しかし、キーワードのみから意味表現を構築するので解釈を誤ることもある。

本稿では、自然言語インタフェース構築環境SCLIBE (Self-Contained Natural Language Interface Building Environment) について述べる。SCLIBEは、キーワード抽出型の解析手法と事例ベース型の解析手法が互いに相補的な関係にあることに注目して構築したモデルであり、KBPのモジュールとPBPのモジュールが、協調して問い合わせ解析を行なう。キーワード抽出型解析がその解析を失敗すると、アプリケーション設計者は、その失敗事例を解析し、それらの失敗を修復する知識を対話的に定義して、PBPの知識ベースに加えていく。その結果、一方のモジュールの失敗を他方のモジュールが修復するように動作する。両者が協調することで、どちらか一方だけを使うより、良い性能が得られる。

2　キーワード抽出型解析モジュール KBP

テーブル1は、KBPが対象とするテーブル型データベースの例である。

タイトル	著者	出版社	日付	値段	発行部数
Dynamic Memory	Schank	Cam P	1983	$35	200
Society of Mind	Minsky	S&S	1985	$40	250

テーブル1：KBPの対象データベースの例[1]

対象のデータベース中の個々の属性名や属性値にそれを参照する言い回しがインデックスとして括りつけられている。例えば、テーブル1の「タイトル」属性に対するインデックスは、「本」「書名」「名前」「と名づけられた」等である。このような、項目名に対するインデックスのことを**項目名インデックス**と呼ぶ。一方、項目値に対するインデックスは、**項目値インデックス**と呼ぶ。KBPが問い合わせ文からSQL表現を生成する基本アルゴリズムは、以下の通りである。

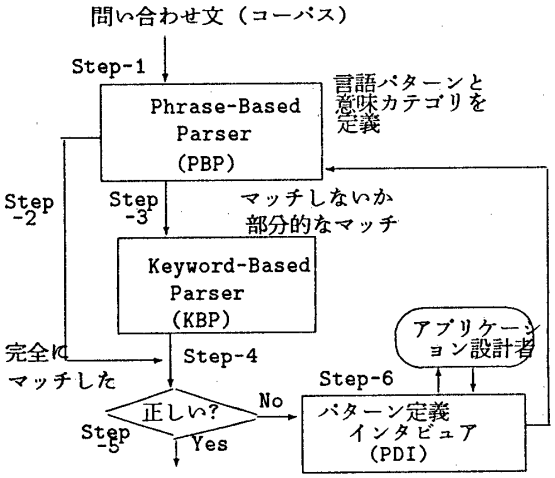

図-1: SCLIBEの処理の流れ

1. KBPは、解釈すべき利用者の問い合わせ文から、項目名インデックスと項目値インデックスのみを取り出す。問い合わせ文の他の部分については、すべて読み捨てる。
2. 項目名インデックスが抽出された時は、それが参照する項目名を、SELECT句の要素として保持する。
3. 項目値インデックスが抽出された時は、それに対応する項目名がその値をとらねばならないという制約の形 (項目名 = 項目値) にして、WHERE句の条件として保持する。
4. 入力文中のすべてのインデックスが処理されたら、保持されているSELECT句とWHERE句の要素をそれぞれ合わせて、SELECT-FROM-WHEREスケルトンに代入する。

例えば、「S&Sで出版された本を示せ」という文が与えられると、KBPは、例文の中から「本」「出版された」「S&S」だけをキーワードとして抽出する。「本」は「タイトル」への項目名インデックス、「出版された」は「出版社」への項目名インデックス、「S&S」は「出版社」への項目値インデックス、である。その結果、

SELECT タイトル, 出版社　**FROM** テーブル1
WHERE 出版社 = S&S;

というSQL式が意味表現として生成される。

3　SCLIBEの処理の流れ

図-1は、SCLIBE内の処理の流れを示している。SCLIBEは、アプリケーション設計者が前もって収集しておいた対象領域の問い合わせ文のコーパスから、問い合わせ文を1つずつ取り出して解釈し、アプリケーション設計者にその解釈が正しいかどうかを確認していく。まず、PBPが与えられた入力文の処理を試みる (Step-1)。もし、PBPが、パターン・概念対データベースの中から入力文に完全にマッチする言語パターンを見つければ、それに対応する意味概念が入力文の解釈として生成される (Step-2)。もし、PBPが入力文にマッチする言語パターンを見つけられなかったら、その文は、KBPに渡される。もし、PBPがパターン・概念対データベースの中から、入力文に部分的にマッチする言語パターンを見つけたときに

---

* Natural Language Interface Building Environment

† Hideo SHIMAZU, Seigo ARITA and Yosuke TAKASHIMA

‡ C&C Information Technology Reseach Labs., NEC Corp.

[1]「値段」と「発行部数」の値は、正しくない。

は、PBPは、入力文中でマッチした部分を、対応する意味概念に置き換える。そして、その変形された入力文をKBPに渡す (Step-3)。KBPは、入力文からキーワードだけを抽出し、入力文の意味を構築する。KBPは、その意味表現を生成する (Step-4)。

次に、PBPまたはKBPによって生成された入力文の解釈結果がアプリケーション設計者に提示される。アプリケーション設計者は、その解釈が正しいかどうかを評価する (Step-5)。もし、その解釈が正しければその入力文の定義の作業は終了し、新たな入力文がコーパス集合からから取り出されて、同様な処理がなされる。もし、アプリケーション設計者が、その解釈を正しくないと判断したときには、知識獲得支援機構PDIが励起される。PDIは、開発者にその入力文の正しい解釈を尋ねる。アプリケーション設計者は、PDIを通じて例文の正しい解釈を提示する (Step-6)。新たに加えられた定義は、KBPの知識ベースかPBPの知識ベースに格納され、次にSCLIBEが同じ文に遭遇したときには、SCLIBEは、その文を正しく解釈することに成功する。

4 KBPの失敗とPBPの修復

アプリケーション設計者は、KBPの失敗事例に遭遇すると、その失敗を繰り返さないように修復し、その修復結果をPBPの知識ベースに加える。そうすれば、KBPが失敗したこれらの問い合わせ文をPBPが正しく解釈できるようになる。以下にKBPが解釈生成に失敗し、アプリケーション設計者が修復する典型的な状況を列挙する。

[失敗 - 修復 -1] 慣用句的な表現を含む入力文や空間的表現を含む入力文の場合：
KBPは、入力文に「100より大きい」のような慣用句的な表現を含んでいたり、「AとBの間の」のような空間的表現を含んでいると正しい解釈を生成することができない。この場合、その特定の言い回しとSELECT-FROM-WHEREのSELECT句かWHERE句の要素の間の対応関係を定義してやれば良い。例えば、問い合わせ中に「価格が1万円以上2万円以下」という言い回しがあり、このためにKBPが正しい解釈を生成できなかったとしよう。アプリケーション設計者は、この言い回しに対して具体的な解釈を次のように定義すればよい。

定義 1: パターン列が「価格、1万円、以上、2万円、以下」なら、
(1) SELECT句に「価格」を加え、
(2) WHERE句に、「価格 > 1万円、価格 < 2万円」を条件として加えよ」

[失敗 - 修復 -2] 問い合わせの構造が、SELECT-FROM-WHEREタイプでないがSQL表現で記述できる場合：
KBPは、すべての問い合わせが、SELECT-FROM-WHEREタイプであると仮定している。従って、問い合わせの内容が、それ以外のSQL構造、例えばSELECT-FROM-GROUP BY-HAVING、のような構造のときは、正しく解釈を生成できない。例えば、「合計数が100冊以上の著者と発行部数をリストせよ」という文は、SELECT-FROM-GROUP BY-HAVINGで実現される。この場合、アプリケーション設計者は、言い回しに対して、新しいSQL文スケルトンを定義する。

[失敗 - 修復 -3] 問い合わせがメタ的な質問の場合：
「このデータベースからどんな情報が得られるのか？」のような対象データベースにとってメタ的な質問をされると、KBPは正しくキーワードを抽出できないので、正しい解釈が生成できない。この場合は、言い回しの意味表現をSQLの式で実現できないので、全く新しい意味カテゴリを生成して、それに応答の手続きを括りつけるしかない。これは、従来の事例ベース型解析手法における個々のパターンの定義方法と同じで、アプリケーション設計者にとっては非常に面倒な作業である。

[失敗 - 修復 -4] KBPが1つの入力文から沢山の解釈を生成してしまう場合：この場合は、上記3つの修復のどれかを試みると解決する。

5 PDIユーザインタフェース

失敗 - 修復の分類のうち、アプリケーション設計者にとっては、修復 -1,2,3 の順にその定義に複雑な手順を要する。あるKBPの失敗に対して、一般には、2つ以上の修復方法が可能であるが、アプリケーション設計者としては、できるだけ簡単な方を選択すべきである。SCLIBEでは、上記3つの定義を定型タスクとして用意し、アプリケーション設計者が上記の定義を簡単に定義できるように支援している。PDIのアプリケーション設計者との対話的定義プロセスは、以下のように続く。

1. PDIは、設計者に、KBPとPBPの両方が解釈を失敗した入力文を示して、その正しい解釈を定義するように依頼する。
2. アプリケーション設計者は、失敗の理由を診断する。
3. アプリケーション設計者は、失敗を修復するための修復パターンを選択する。この時、アプリケーション設計者は、定義すべき言語パターンを、汎化オペレータを使うことによって、汎化／修飾することが可能である [3]。
4. PDIは、失敗した問い合わせ文の解釈を再び試みる。その結果をアプリケーション設計者に提示する。その解釈が正しければ、その問い合わせの処理を終了する。正しくなければ、1に戻る。

6 まとめ

SCLIBEは、テーブル型の構造が多いオンライン・データベースの自然言語インタフェース構築に特に向いている。この分野の要件としては、(1) 利用者の自然言語による問い合わせだけでなく、オンライン・データベースの利用者がしばしば好んで使う非文法的な表現スタイル (キーワード列とかコマンド言語のようなスタイル) の解釈が可能であること、(2) オンライン・データベース利用者が我慢できる程のレスポンスタイムであること、(3) アプリケーション作成者にとって、個別システム構築・メンテナンスが容易であること、がある。SCLIBEは、これらの3つの条件を満足している。KBPモジュールは、キーワードのみから意味を構築するので、非文法表現の解釈に強いし、応答速度は速い。また、SCLIBEでは、解析部分と対話的失敗・修復部分が密結合されているのでアプリケーションの開発、メンテナンスがしやすい。特に (3) においては、アプリケーション設計者は、PBPのパターン・概念対定義としては、KBPがその処理を失敗した言語パターンのみを定義すればよいので、その負担はPBPだけを使ってシステム構築する場合にくらべて大幅に楽になる。このように、従来のフレーズ・ベース解析システム構築の困難さの原因であったパターン・概念対定義の大変さが、大きく改善されたことはSCLIBEにおいて特記すべきことである。

参考文献

[1] 有田、島津、高島「簡易型自然言語インタフェースモデル」人工知能学会全国大会、1991年6月.

[2] Martin, C.E., "Case-based Parsing", In *Inside Case-based Reasoning* edited by R. Schank and C. Riesbeck, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[3] 島津、高島「コーパス解析に基づく事例ベースパーザ」情報処理学会・人工知能研究会、1991年3月.

[4] Wilensky, R. et. al., "UC – A Progress Report", Rep. UCB/CSD 87/303, 1986.