自 然 言 語 処 理 100－8
（1994．3．18）

知識ベースを利用した自然言語処理システム

阿折義三

日本DEC研究開発センター

自然言語処理（NLP）のボトルネックは解析対象文の曖昧性であり，従来は格文法や意味属性などで解決しようとしてきたが，現実の自然言語を人間並みに正確に解析できるまでに至っていない。この限界は自然言語が人間の所有する知識を前提として生成されるというところに根ざしており，根本的には人間並みの知識を利用しなければ満足な結果は得られないと考えられる。最近のアプローチとして例文ベースの利用が話題となっているが，例文ベースは慣用的または例外的表現には効果が予想されるものの，事前に準備した例文ベースだけでは現実の自然言語の多様性には対応しきれない。むしろ，一般化された知識の方が少ない知識量で多様な言語現象に対応できる。筆者は4年にわたって文法知識や例文知識などの言語知識に加えて，一般常識，専門分野知識，文脈知識などの多様な知識を概念のレベルに一般化して利用するための各種要素技術を開発するとともに，これらの多様な知識をすべて利用可能とする統合方式を開発してきた。本年度はこれらの技術をNLPシステムにインテグレートし，機械翻訳に適用して効果が認められたので，ここに報告する。

Knowledge Based Natural Language Processing System

Yoshizo Aori

Research and Development Center DEC-Japan

The bottle neck of Natural Language Processing(NLP) is ambiguity of natural language to be analyzed, and people have been trying to solve this problem by case slot grammer, semantic attribute, and so on, but they have not been successul enough compared to capability of a human. This restriction comes from the fact that natural language is produced by a human based on the big assumptions that is equal to the total knowledge stored in his/her brain. So, it is expected that satisfactory result is not obtained without using big knowledge base as a human has. Recently, many people are interested in example based NLP to solve this bottle neck. Example base is effective for idiomatic expressions and exceptional expressions, but a prepared set of example base cannot cover real world of language phenomena, because variation of language phenomena is almost unlimited. On the other hand, small volume of generalized knowledge can be applicable for large variations of language phenomena. In the past 4 years, I have been developing component technologies for applying generalized knowledge of common sense, domain knowledge, and context knowledge, in addition to language knowledge such as grammer rule or example base. Also, I have been developing integration technologies that allow us to integrate all the type of knowledge. This year, I have completed to integrate all these technologies into a NLP system, and I'd like to report that I could get very good results by applying the system to Machine Translation.

1 はじめに

言語知識だけでなく常識，専門分野知識，文脈知識など一般知識をNLPへ利用する方法については研究報告は少なく要素技術としてもまだ十分確立してはいない段階であるが，実はAIやデータベースの分野での伝統的な簡単な知識表現がそのために利用できる。筆者は一般知識の利用のための要素技術を洗煉する前に，まず上記あらゆるタイプの知識を統合的に利用可能とする全体システムの枠組みを作ることが人間並みの自然言語理解という最終ゴールへの近道と考えた。一旦，知識を利用する全体システムができ上がれば，あとは要素技術やシステム技術の改善の繰り返しによって，比較的短期間にNLP能力が見違えるほど向上し，人間に近づけると確信しているためである。筆者はこの信念に基づき4年間知識ベースによる自然言語処理システムの研究開発を続けてきたが，ようやく機械翻訳に適用して効果が確認できたのでここに報告する。本報告では，代表的な知識利用のための要素技術を説明しながらも，全体システムとしての統合の問題に重点をおいて話を進める。

2 実在の自然言語の分析からの出発

一見すばらしいアイデアも実際に適用してみると，期待ほどの効果がなく裏切られるケースが多いものである。筆者はある程度まとまった量の実在の自然言語をサンプルとして選定し，それを正しく解析するには何が必要かというアプローチを取った。まず，この文集合を人間シミュレーションにより繰り返し解析してみることにより本当に必要な要素機能を抽出する。そしてその機能を実現し，実際にその文の解析に適用し効果を評価／検証する。結果が満足いかなければ，その機能や実現法を改善する。このような "RESULT DRIVEN" アプローチは品質改善活動の世界では鉄則であるが，現実問題から離反しないためには研究開発でも見習うべき方法である。具体例として筆者による下記英語例の文解析の人間シミュレーション結果を示す。

> The DECnet/SNA Data Transfer Facility Software is a DECnet/SNA access routine that connects a VMS/DTF server node and its clients on a DECnet network to IBM MVS and VM client systems on an SNA network.

この文は選定したドキュメントの中では最も複雑な構造をしたものの1つである。人間シミュレーションを繰り返した結果，人間が文理解に使っている能力のうち，

1) connect NP1 to NP2 のような大局的文型を優先的に認識できる
2) Network, Server, Clientなどに付随する修飾を一般化し， Network, Server, Client の概念のレベルで推論できる
2) Networkなどの構成要素関係や，IBM とVM や MVS との所有関係のような概念間の関係知識が利用できる

が最も重要な能力であることがわかった。この文の解析のためには，名詞句の句構造を正しく決定することが決定的に重要な問題であり，そのためには，本システムとしてもこれらの能力は必須機能であると考え，これらの能力実現のため全力を投入した。一方，この文は例文ベースの限界も示唆していると考えられる。すなわち，名詞句のバリエーションはあまりに多く，例文ベースとしてすべてをカバーできそうにない。ちなみに3種類の市販の機械翻訳システムで，この文を翻訳してみたが，正しく句構造を決定できるものは1つもなかった。したがって，この文の句構造を正しく決定できることが現状の技術レベルを越える1つの目標となった。その他の文例を2つ上げておく。

> The DTF access routine requires two major pieces of software.

The VMS/DTF server software kit is installed on the VMS node chosen to be the VMS/DTF server node.

"two major pieces of software" は "2枚の主なソフトウエア" と訳すわけには いかない。"2種類の主なソフトウエア" とでも訳したいところだ。"chosen to be ..." は 動詞単語訳と文法規則だけで訳すとうまくいかない。 まとめて "...として選ばれた" と訳したい。このような場合，例文ベース の考えでまるごと対訳として定義した方が手っ取り早い。このような例から例文ベース利用機能もインテグレートする必要があると考えた。

1つの単語で訳語を複数使い分けなければいけないケース はサンプルの範囲では ほとんどなかった。今後，サンプルのサイズと種類を拡大して評価しなければ 正確な判断はできないが，DECのユーザ・ドキュメント という非常に 大きい範囲であっても訳語はかなり一意に決まるようにみえた。 したがって，構築する知識ベースをDECユーザ・ドキュメント解析用に限定するならば，大規模な意味属性体系の定義をしなくてもいけるのではないかとの期待を 持たせた。その結果，本研究開発の第一段階では大規模意味属性体系の実現にはあまり力を入れないこととした。ちなみに，市販のMTシステムの訳語はDEC ユーザ・ドキュメントに対してはまったく不満足なものであった。 したがって，いずれ問題になるテーマと覚悟はしている。

3 知識ベースに基づくNLPシステムの要素技術体系

常識，専門分野知識，文脈知識などの大規模知識ベースを利用したNLPシステム を構築するには，以下の要素技術全体を有機的に連係して動作させる必要がある。

言語知識定義:

形態素解析規則，文法／文型規則，ワード列構造変換規則(機械翻訳の場合)

形態素生成規則(機械翻訳の場合)，一般単語知識 （各単語には品詞， 意味属性，シソーラス関係，文型規則，慣用句，訳候補，日本語活用形 などを含む。）

意味属性(意味素性)体系定義，意味制約定義:

制約単一化機能: 意味属性に基づく単一化の制限，一般知識に基づく単一化の制限など

概念階層関係定義:

常識／分野知識定義:

ヒューリステイック知識，構成要素関係／所有関係などの概念間関係知識など

文脈知識処理:

文脈表現機能，文脈登録機能，文脈評価機能（一種の制約単一化の指定方法）

大規模知識アクセス機能:

単語による主メモリ内知識ベースのサーチ，オンデマンド・ロード必要性検出，

ファイル・アクセス，ファイルからの読み込みレコードのメモリ内知識構造への展開 など

テキスト現象と知識を結び付ける機能:

テキスト現象の一般化ルール，テキスト現象と知識の連想

推論エンジン:

形態素解析，文法解析，意味解析，ワード列変換／形態素生成(機械翻訳の場合)など

競合解消機能:

知識定義への確信度付与機能，確信度計算機能，形態素／句構造／訳候補などの競合解消

知識デバッグ機能:

辞書作成機能: 電子辞書からの自動変換，辞書ファイル作成機能など

一見してわかるとおり非常に大規模な体系になる。したがって，これをいかにシステムとしてまとめるかはそれ自身１つの重大な技術テーマである。筆者は参照モデルとしてAIにおけるエキスパート・システム構築ツールの考え方が有効と考え，これをNLPに適用することとした。つまり "NLP用知識ベース" と "知識ベースと完全に分離されたNLP向け推論エンジン" というアプローチで全体システムがまとめられると考えた。もうひとつのシステム化のポイントは

形態素解析 -> 文法解析 -> 意味解析 -> [生成/変換]

というような実行要素をシーケンシャルな実行フェーズとして考えるかどうかという問題がある。筆者による人間シミュレーションの結果では人間はこのようなフェーズを踏んでいるとは思えない。むしろ先行して進められる部分はどんどん先へ進むことによって，できるだけ早期にテキスト解釈の候補を絞り込んでいると観察した。一方，プログラミング上もこの様なフェーズに分けると開発量が膨大になると予想された。そこで，上記実行要素をフェーズに分けることなく，どんどん先行して処理を進める方式を目標とした。そして，それは上記実行要素をすべて１つの推論エンジンで実行することにより実現可能となった。

4 主要要素技術の実現法

筆者のアプローチは個別要素技術にあまり深入りすることなく，まずシステムが全体としてバランスよく，受け入れ可能な性能で，実際の自然言語解析の問題を解決することを優先することであった。実際にアプリケーションに適用して成果が具体的に見えることにより，NLPの全体像が理解でき，本当のボトルネックとなる要素技術がわかると期待したことによる。しかしながら，本報告では全体像を理解いただくために，まず代表的要素技術の実現法を具体例を用いて紹介しようと思う。

4.1 知識表現

使えそうなものは一通り準備した。結果として言えることはAIの伝統的知識表現がNLPに有効に使えるということである。具体例を以下に示す。

概念階層表現:

コンピュータ，ワークステーション，DECstationの間の概念階層定義例:

```
($ch '$n nil '&computer nil nil 0.9)
($ch '&computer nil '&workstation nil nil 0.9)
($d '&workstation nil "DECstation" nil nil 0.9)
```

その他階層表現:

"クライアント" は "ネットワーク" の構成要素である。
($def_hi '&client '$comp_of '&network 0.9)

"サーバ" は "ネットワーク" の要素である。
($def_hi '&server '$comp_of '&network 0.9)

"MVS" は "IBM" のものである。
($def_hi "MVS" '$owned_by "IBM" 0.9)

"VM" は "IBM" のものである。
($def_hi "VM" '$owned_by "IBM" 0.9)

3段論法的知識:

形態素解析ルール:

区切り記号＋アルファベット -> アルファベットは部分ワードの始まり

```
($rule ($if ($seq '$delimiter '$alpha)) ($then ($string '$part_of_word)))
```

部分ワード+アルファベット -> 部分ワード

```
($rule ($if ($seq '$part_of_word '$alpha)) ($then ($string '$part_of_word)))
```

注: $seq関数はテキスト現象のパターンを指定する。

文法／文型解析ルール:

動詞＋目的語 -> 動詞句

($rule ($if ($seq '$verb '$obj)) ($then ($phrase '$vp) .....))

"connect" + 目的語 + "to" + 名詞句 -> 動詞句，目的語 "を" 名詞句 "に接続する"と訳す

($rule ($if ($seq "connect" '$obj "to" '$np))

($then ($phrase '$vp) ($trans '$vp '$obj "を" '$np "に接続する")))

注: $trans は翻訳用関数であり，ワード列を指定の列と値に変換する。

一般的ヒューリステイック知識:

指定された仮説を知識ベースで証明する後向き推論のための ルール表現も準備した。紙面の都合でここでは詳細は省略する。現在のところ有効に使いきれていない。

4.2 テキスト現象の一般化

テキスト現象のバリエーションは修飾や名詞の組み合せによって無限にある。この多様なテキスト現象に知識を適用するためには，テキストを概念のレベルに一般化して知識を利用することが最も重要な技術の１つとなる。テキスト現象の一般化の例を以下に示すが，この例に示す方法は記述量が多く，もっと一般化した記述方法にして，記述量を削減することが課題である。

a large SNA network というテキスト現象をネットワークの概念名 &network に一般化する

($rule ($if ($seq "NETWORK")) ($then ($phrase '&network)))

($rule ($if ($seq '$prop '&network)) ($then ($phrase '&network)))

($rule ($if ($seq '$adj '&network)) ($then ($phrase '&network)))

($rule ($if ($seq '$art '&network)) ($then ($phrase '&network)))

$prop: 固有名詞

$adj: 形容詞

$art: 冠詞

4.3 知識ベースによるテキスト現象の解釈の確信メカニズム

本機能は知識ベースを利用する上で，もう１つの重要な技術である。筆者は$seq関数で指定されるテキスト現象に対する制約評価関数によって，知識ベースで裏づけられる結論（=テキスト現象の解釈）であるかどうかを評価するメカニズムを導入した。具体的には本システムでは

- o 階層関係知識を評価する $memberp
- o 知識ベースをbackward 推論して仮説述語を評価する$predicatep
- o ワード／概念の意味属性を評価する $attr_p

などの評価関数をルールの条件部に書くことによって実現している。例を以下に示す。

($rule ($if ($seq '$np1 "AND" '$np2 '$prep '$np3) ($memberp '$np1 '$np2 '$comp_of '$np3))

($then ($phrase '$np) .....))

上記ルールは，テキスト解析において，たとえば変数 $np1, $np2, $np3 にそれぞれ概念 &client, &server, &networkが単一化された場合，ネットワークに関する構成要素の関係知識が知識ベースにあれば，結論が非常に高い確信度をもって成り立つ。

($rule ($if ($seq '$np1 '$np2 "AND" '$np3) ($memberp '$np2 '$np3 '$comp_of '$np1))

($then ($phrase '$np) .....))

上記ルールは，変数 $np1, $np2, $np3 にそれぞれ"IBM", "MVS", "VM" が単一化された場合，IBMとMVSおよびVMに関する 所有関係知識が知識ベースにあれば，結論が非常に高い確信度をもって成り立つ。

4.4 文脈知識表現

簡単な使用例を以下に示すが，紙面の都合で詳細は別の機会とする。

文脈に応じた訳語の選択関数$sel_contの使用例:

```
($d '$n nil "transfer" nil  (list ($sel_cont "付け替え" '&finance) ($sel_cont "転送" '&network)))
```

文脈登録関数$upd_cont の使用例:

```
($rule ($if ($seq '&topic "is" '&finance)) ($then ($upd_context '$concept '&finance)))
```

文脈評価関数$contextp の使用例:

```
($rule ($if ($seq '$subj "is" "a" "transfer" "issue") ($contextp &network '$comp_of))
        ($then ($phrase '$sent) ($trans '$sent '$subj "は" "転送問題" "です")))
```

4.5 テキストからの知識獲得

テキスト現象を知識の形式に変換し，メモリ上の知識ベースに組み入れる$get_k関数により，分析対象のテキストから知識を獲得し，文脈として利用できる。例を以下に示すが，まだ実際の自然言語解析には使用していない。ドキュメントからの知識獲得の問題にも有効と期待している。

```
($rule ($if ($seq '$subj "has" '$obj) ($attr_p '$subj "&person"))
        ($then ($phrase '$sent) ($get_k '($def_hi '$obj '$owned_by '$subj 0.9))))
($rule ($if ($seq '$subj "has" '$obj) ($attr_p '$subj "&physical"))
        ($then ($phrase '$sent) ($get_k '($def_hi '$obj '$comp_of '$subj 0.9))))
```

4.6 大規模知識アクセス機能

知識ベースのアクセススピードはNLP性能上の重大な要因であるため，知識ベースサイズの増大に対してアクセススピートが低下しないようにすることも重要な要素技術である。この問題を解決する１つの方法として，知識ベースの構造を百科事典と同じように見出しごとにグルーピングする方式とした。テキスト解析時に単語を検出した時点で，この単語を見出しとしてISAMファイルをサーチし，その見出しの関連知識をデイスクからオンデマンド・ロードする。この結果，推論エンジンが探索する範囲はテキスト（たとえば１つの文）に現れる単語に関連する知識だけになり知識ベースがどれだけ大きくとも解析時間が一定限度におさまると期待される。また，必要な主メモリも一定量で処理できる。見出しとそれに付随するオンデマンド・ロード単位の例を示す。

```
"THAT"  <--- 見出し
    ($d '$pro_noun    "THAT" "&pro_that" '($subj) '("それ"))
    ($d '$adj "THAT" "&adj_that" '("&katsuyou_type3") '("その"))
    ($d '$conj "THAT" "&conj_that" nil '(" ")) .....
"CONNECT" <-- 見出し
    ($d '$vt nil "CONNECT" '("&sahen" $now) '("接続"))
    ($d '$vi "CONNECT" "&vi_connect" '("&ra_5" $now) '("つなが")) .....
    ($cf $very_high)
    ($rule ($if ($seq "CONNECT" '$obj "TO" '$np))
            ($then ($phrase '$vp) ($trans '$obj "を" '$np "へ接続")))
    ($rule ($if ($seq '$be "CONNECTED" "WITH" '$np))
            ($then ($phrase '$be_p) ($trans '$be_p '$np "と親戚関係" '$be))) .....
```

ロード単位

ロード単位

5 システムの統合化方式

4章で述べた要素技術はすべて推論エンジンによって実行を統合されるが，そのかなめになる技術が確信度と競合解消機能である。文法知識，例文知識，一般常識，専門知識，文脈知識などはすべて知識であり，これらを統合して使うことによってはじめてNLPの能力を人間並みに高めることができる。これらの知識を統合する一つの方法が知識の適用可能性の程度を表わす確信度と，それに基づく競合解消機能であり，筆者が本論文で最も強調したい部分である。競合解消には2つの意味がある。"解釈可能な候補を絞り込むこと"，および"最も適切な解釈を選択すること"である。上記多様なタイプの知識を利用することにより，テキスト現象に適用可能な知識は文法や意味属性などの言語知識だけに基づくNLPに比較し大幅に増加する。その結果，1つの文の解釈可能な句構造の組み合せは爆発的に増加する。したがって，文解析の早期段階から競合解消により解釈可能な候補を絞り込まなければ，実行スピードは許容範囲に収まらない。一方，最も適切な解釈を選択する能力は解析の質そのものである。競合解消は

1) 知識登録時，知識に応じた適切な確信度を付与する。
2) テキスト現象にマッチした知識に対して結論（現象の1つの解釈）の確信度を計算する。
3) 結論集合の中からもっとも確信度の高い結論を選択する。

の3つのステップからなる。

知識への確信度の付与:

<<一般的／多義的な知識には低い確信度，一方，限定的／一意な知識には高い確信度>>という確信度付与規準により，多様な知識のタイプを分類でき，テキスト現象に適用可能な知識のうち最も適切な（=確信度の高い）知識を選択することができるようになる。たとえば，文法レベルの知識は非常に一般性があるが，曖昧性が多い（=多義的）ので最も低い確信度を付与する。一方，例文知識"Time flies like an arrow."のように文全体としてはほぼ一意な意味を持つと考えられる場合，文全体には非常に高い確信度を付与すればよい。

確信度計算:

単純なANDやORの確率計算ではうまくいかない。ポイントは知識に裏づけられた（=高い確信度を持つ）部分句構造解釈を文全体の解釈にどう引き継ぐかという点にある。計算にはひと工夫が必要となる。紙面の都合でここでは詳細は省略する。

競合集合への登録と競合集合からの選択:

本システムにはシステムが暗に実行する選択とアプリケーション開発者が指定する選択がある。選択実行そのもは競合集合の中からもっとも確信度の高いものを選ぶだけであり，技術的問題はない。問題はいつ何を競合集合に登録し，いつ選択を実行するかである。本システムではこの指定機能をアプリケーション開発者に解放している。その理由はどのタイミングで競合解消を実行するかはそれ自身知識であり，人それぞれ異なると考えたからである。具体例を以下に示す。

動詞句を検出したとき，それを'$vp_confという名前の競合集合に登録する。
($rule ($if ($seq '$vp)) ($then ($dcl_conf '$vp_conf '$vp)))
文末を検出した場合，動詞句競合集合'$vp_confから最も確信度の高い動詞句を1つ選択する。選択された動詞句には'$vp_finalと名前を付ける。
($rule ($if ($seq '&eos)) ($then ($exec_conf '$vp_conf '$vp_final 1)))

6. システム評価

評価環境は開発言語がVAX LISP, 使用したハードウエアが30 MIPSのVAXシステム, 使用した作業メモリが10MB（ただしLISPインタプリタおよびシステム関数領域は除く）である。評価には筆者が問題分析に使用したDECのユーザドキュメントから抜粋した1ページを使用した。翻訳スピードは1文当たり0.3秒から3.3秒であり, 実用可能なレベルと判断している。ただし, まだ数倍の向上余地がある。翻訳品質については評価対象からさらに抜粋した下記サンプルで推定願いたいが, 少なくとも句構造の解釈は完璧であったことを付け加えたい。なお, 使用した一般知識は4章で例示したものだけである。

英文	翻訳文
The DECnet/SNA Data Transfer Facility software is a DECnet/SNA access routine that connects a VMS/DTF server node and its clients on a DECnet network to IBM MVS and VM client systems on an SNA network.	(DECnet/SNA データ転送 ファシリテイ ソフトウエア は DECnet ネットワーク の上の VMS/DTF サーバ ノード および その クライアント を SNA ネットワーク の 上の IBM の MVS および VM クライアント システム に 接続 する DECnet/SNA アクセス ルーチン です .)CPU Time: 3.34 sec., Real Time: 3.69 sec.
The purpose of DTF is to extend the DECnet file transfer facilities to the IBM MVS and VM environments.	(DTF の 目的 は DECnet ファイル転送 ファシリテイ を IBM の MVS および VM 環境 へ 拡張 する こと です .)CPU Time: 1.69 sec., Real Time: 1.96 sec.
Therefore, the DTF software does not support file transfers between two DECnet clients.	(したがって , DTF ソフトウエア は 2 つの DECnet クライアント の間の ファイル転送 を サポート しません)CPU Time: 1.14 sec., Real Time: 1.32 sec.
You should use the DECnet file transfer utilities for this purpose.	(あなた は DECnet ファイル転送 ユーテイリテイ を この 目的 のために 使う べき です .)CPU Time: 0.97 sec., Real Time: 1.13 sec.
On VMS clients, DTF supports the DCL interface, the TRANSFER/DTF utility supplied with VMS/DTF, the RMS programming interface, and the VAX DATATRIEVE product.	(VMS クライアント の上で , DTF は DCL インタフェース , VMS/DTF とともに提供される TRANSFER/DTF ユーテイリテイ , RMS プログラミング インタフェース , および VAX DATATRIEVE 製品 を サポート します)CPU Time: 2.01 sec., Real Time: 2.44 sec.

7. おわりに

本研究により, 知識ベースを利用してテキスト現象の解釈の質を大幅に向上できる目処が立った。また, 大規模知識ベースを利用したNLPの実行スピートに関する実現見通しも立った。筆者は現状のマイクロプロセッサの機能で100MIPSのオーダの性能があれば, 本技術の延長で, 改善を繰り返すことにより, 人間に近いNLPができると確信している。まだまだ知識ベース利用に関する要素技術には改善すべき課題も多いが, 今後の最重点課題は大規模知識ベース構築技術である。多数の研究開発者がこの問題にチャレンジすることを期待する。

参考文献

[1] 阿折義三: エキスパートシステム作成支援ツール技術の自然言語処理への適用, 情報処理46回全国大会, pp. 3-193(1993)