自 然 言 語 処 理 65 － 3
（ 1 9 8 8. 3. 2 5 ）

複数のアプリケーションと応答可能な自然言語インターフェース・システム

渡辺 日出雄　丸山 宏
日本アイ・ビー・エム株式会社 東京基礎研究所

本論文では、今までの自然言語インターフェース・システムが抱えてきた問題点であるカバレージの狭さを解決するために考えられた手法である"Full-language Approach"及びこの手法を用いて構築した自然言語インターフェース・システムについて説明する。今日まで構築されてきた自然言語インターフェース・システムは、それが受付ける事のできる文の集合(即ちSub-language)を大きくしようとして来たが、結局十分なカバレージを得ていない。それに対して、本論文では、あらゆる文を予め既知であるパターンに収束させるという手法(Full-language Approach)を用いた。これは、ある二つの文の間の意味的な近似度を対応する単語間の距離と構造的な類似度から計算するというものである。ここで、単語間の意味的近似度は、各単語に与えられた夫々のビットがある属性を表すビット・ベクター(属性ベクトル)の交差する角度により得られ、構造的近似度は、構造的に重ならない部分の大きさに基づいて計算される。また、この"Full-language Approach"の自然な応用として、一つのインターフェース・システムが複数のアプリケーションと応答可能な事により、インターフェースの概念的カバレージをも向上できる事を示す。

A Natural Language Interface System Which Can Respond To Many Applications

Hideo WATANABE and Hiroshi MARUYAMA
Tokyo Research Laboratory, IBM Japan, Ltd.
5-19, Samban-cho, Tiyoda-ku, Tokyo 102, Japan.

We propose a "Full-language Approach" which solves the narrow coverage of previous natural language interface (NLI) systems, and present a NLI system adopting this approach. So far NLI systems have been trying to extend the set of sentences (the sub-language) to which they can respond, but their coverage is still not sufficient. In contrast to existing approaches, we developed a method which always maps a sentence to the semantically closest pattern among several already stored patterns. To determine the closest pattern, we employ as a measure for the similarity between two sentences the sum of the structural similarity and the similarity between corresponding words. The similarity between two words is given by the crossing angle of two bit vectors, each bit of which represents a particular aspect. Structural similarity is given by the size of the non-overlapping parts. This approach allows a NLI to communicate naturally with many different applications. Because of that, the conceptual coverage of a NLI can increase.

０．はじめに

これまで、数々の自然言語インターフェース・システムが構築されてきたが[1][2]、一般に広くユーザーの間に用いられてはいない。
この理由としては、
　１）意味的カバレージの狭さ（応答可能な文の少なさ）
　２）カスタマイズ作業の複雑さ
などが考えられる。[3]
本論文では、この意味的カバレージの問題を解決するために考案した"Full-language Approach"と、それを元に作った自然言語インターフェース・システムについて説明する。まず、第１章では今までの自然言語インターフェース・システムの問題点と我々の手法である"Full-language Approach"について述べ、２章ではその基礎となる近似度の計算手法について述べる。さらに、３，４章では我々の自然言語インターフェース・システムの構成等について述べ、５章でその実行例を示す。そして、最後に６章に於て結論を述べる。

１．自然言語インターフェースのカバレージ

自然言語インターフェースのカバレージには、語彙的・構文的・意味的・概念的などのレベルがある。
語彙的カバレージは、辞書の見出しを増やせば、上げる事が可能である。
しかし、構文的・意味的カバレージは、[4]において示したように、それ単独で上げる事はできなく、対象となるアプリケーションに依存する。何故ならば、構文的カバレージとアプリケーションの要求するデータ構造の複雑さのレベルとの間には、トレードオフの関係があるからである。一般に、複雑なデータ構造を要求するアプリケーションを扱う場合には、インターフェースの構文的･意味的カバレージを犠牲にしなければならない。
さらに、ユーザーの立場に立てば、この語彙的・構文的カバレージが高いだけでは、十分ではない。どれだけ多くの問題解決が図れるかと言う概念的カバレージの問題がある。ユーザーにとっては、一つのインターフェースを通して多くのアプリケーションと対話できるのが望ましい形態である。今までの自然言語インターフェースシステムには、この様な観点が欠けていたと言える。

１．１．Sub-Language Approach vs. Full-Language Approach

今までの自然言語インターフェース・システムでは、まず最初に意味モデルを定義し、その上で理解可能である自然言語のサブセットを規定していた。この方法では、意味的にはそのサブセット内の文に近くてもそのサブセットに入っていないため受付けられない文が頻出してしまう。この様なシステムでは、このSub-Languageをどれだけ大きくする事ができるかと言う事が問題となる。そこで、我々はこれを"Sub-Language Approach"と呼ぶことにする。
これに対して、我々は、あるアプリケーションにとって理解可能な幾つかの標準的な文(標準表現と呼ぶ)を用意して、入力文とそれらとの意味的な距離を計算する事により、適切な標準表現を選択する事ができる手法を考案した。これによれば、ユーザーからの全ての入力に対して、意味的に一番近い標準表現が選択される事から、必ずなんらかの応答をする事ができる。これを、"Full-Language Approach"と呼ぶ。

１．２．Single Application vs. Multiple Applications

概念的なカバレージはアプリケーションによって決定されてしまう。即ち、概念的カバレージを上げるためにはアプリケーションが豊富な機能を持つ事が必要となる。そのためには、一つの自然言語インターフェースが複数のアプリケーションと応答可能であれば良い。
これまでの自然言語インターフェースシステムは、単一のアプリケーションとしか対話ができなかった。別の見方をすれば、自然言語インターフェースはある一つのアプリケーションの一部となっていた。
これに対して、我々のシステムは、"Full-language Approach"を用いる事により同時に幾つかのアプリケーションとの対話が可能である。これは、夫々のアプリケーション毎に理解可能な幾つかの標準的な文(標準表現)を用意して、入力文とそれらとの意味的な距離を計算する事により、適切なアプリケーションを選択する、と言うものである。
これにより、自然言語インターフェースの概念的カバレージを上げる事ができる。

２．近似度計算

１．１．及び１．２．で述べた事を可能にするには、入力と幾つかの標準表現との間で、一番意味的に近い標準表現を見付けだす事ができなければ成らない。
この入力と標準表現との意味的近似度の計算は、次の２つの観点から行なわれる。

１）構造的近似度SS
２）その構造内での対応する単語間の意味的近似度SWi

構造的近似度SSを求めるに際して、入力及び標準表現をどのような内部構造として表すかと言う事が問題となる。現在句構造表現が広く一般に用いられているが、これは各種の表層上の表現の揺れに対する構造の変化が大きい。それに対して、概念依存構造は、修飾句－非修飾句の関係が構造を作っているため、表層の表現の揺れに対する構造の揺れが句構造表現に対して少ないと言う特長を持っている。構造的近似度を計算するに際しては、表層の表現の揺れに対する構造の揺れが少ない方が良い。この事から、我々のシステムに於ては、内部表現として概念依存構造を用いている。
構造的近似度は、入力と標準表現を表す依存構造を重ねあわせ、その時の重ならなかったノードの数に比例した距離で与えられる。

また、二つの単語間の近似度は、各単語に各種の属性を０又は１で表したビット・ベクター(属性ベクトル)を与え、その二つの属性ベクトル間の角度を距離と見做す。[5]

$$SW(v1,v2) = 1/\cos\theta - 1$$

図１に属性ベクトルの例を、図２、３にそれぞれ『要求』『使用』に近い単語の例を示す。

最終的な二つの依存構造間の意味的近似度Sは、

$$S = SS + sum(SWi)(i=1,n)$$

で表される。(ここで、nは二つの構造間での対応する単語の対の数)

調査	1100000000001001000000
調整	1000000001000000100(
調味	1100100000010000000
調味料	0000100010001000010(
調印	1000000000100010000
調書	1000100000010001000(
調節	1000000001000000010(
調教	1000000000010000000
調音	1000000000010001000(
調練	1000000000010010000001
調律	1000000001000000010000
調伏	1100000000010000000100(
調停	1100000000010000000100(
天	0000100100000100100000001
天つち	0000100100000100100000001
天の川	0000100100000100100000001
天上	1000100101000000000001001
天下	0000111100100000000010001
天下り	1100000001000000000100(
天日	0000100100000110100000(
式場	0000111100100000000010(
式典	1000000000010000010000
式服	0000100010001000100000(

図１　属性ベクトルの一部

w	φ(w,要求)
懇願	0.083
申請	0.083
懇請	0.173
懇望	0.173
求め	0.173
要請	0.173
要望	0.173
哀願	0.181
求婚	0.181
矯正	0.289
委託	0.289
教示	0.289

図２　「要求」に近い１３語

w	φ(w,使用)
運用	0.076
利用	0.142
逆用	0.166
援用	0.166
収用	0.166
愛用	0.166
酷使	0.263
駆使	0.263
混用	0.263
活用	0.263
共用	0.263
愛玩	0.263

図３　「使用」に近い１３語

３．システム構成

図４にシステム構成を示す。

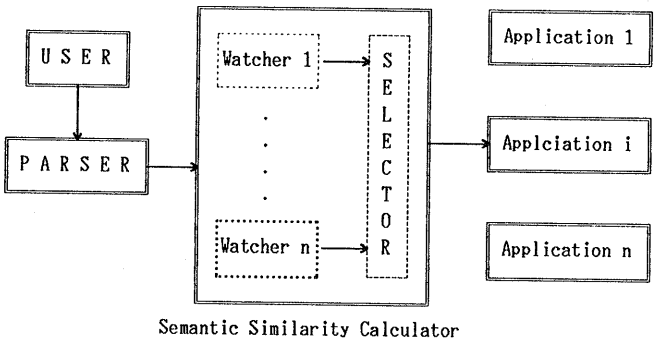

図４．システム構成

全体の処理の流れは、
１）ユーザーからの入力は、パーザーにより概念依存構造に変換される。（このパーザーは、日英機械翻訳用のパーザー[6]を用いている。）この概念依存構造の各単語を表すノードには、その単語の意味を表すビット・ベクターが付けられる。
２）意味的類似度比較器がその入力と一番意味的に近い標準表現を持ったアプリケーションを選択する。WATCHERとは、各アプリケーション毎にあり、そのアプリケーションが持つ幾つかの標準表現と入力との意味的近似度を計算するプロセスである。標準表現は文そのものではなく、アプリケーションが理解可能な表現の断片(標準パターンと呼ぶ)の内包的組合せから得られる。
３）SELECTORは、それぞれのWATCHERから得られた結果を集め、それらを確信度の順番に並べユーザーに提示する。ユーザーはそれらのなかから、ユーザーの意図に一番近いアプリケーションを選択できる。
４）選択されたアプリケーションに対しては、ユーザーの入力の中から有効なデータが取り出されて渡される。

４．意味解析

我々のシステムにおいては，意味解析は，パーザー出力である依存構造木から，それに最も意味的に近い標準表現を構築するプロセスである。前章で述べたように，標準表現は標準パターンの内包的な組み合せで表わすこともできるから，すべての可能な標準表現を求めてそれらと入力文との間の距離を計算するのは現実的でない。我々の意味解析は依存構造木の各部分木に標準パターンをマッチさせて行くことによって行なわれる。マッチした標準パターンの正しい組み合せのうち，距離の最も近い数個の標準表現が，入力文の解釈としてユーザーに提示される。

４．１．ルールの形

ルールは標準パターンである条件部と実行部からなる。標準パターンは依存構造木の形をしており、入力文の依存構造の部分木とマッチする。標準パターンがマッチすると、実行部が起動される。実行部はＬｉｓｐのプログラムであり、通常、現在注目しているノードに解釈を追加するためのコードが書かれる。ルールの適用の際には、そのルールの標準パターンがどのくらい良く入力の依存構造にマッチするかどうかが２章で述べた方法にしたがって計算され、その結果から、解釈のもっともらしさが求められる。
ルールの条件部である標準パターンは基本的には自然言語のある表現に対応する依存構造であるが、マッチしたストリングを取り出したり、ルールの適用条件をより細かくコントロールしたり、解釈を組み合わせたりすることができるように拡張されている。

標準パターンの例を図５に示す。

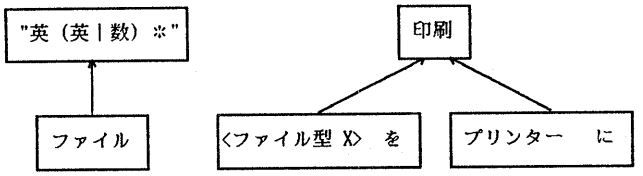

図5 標準パターンの例

標準パターン（a）は『ファイルＡＢＣ』、『文書Ｘ１２』などとマッチする。標準パターン（b）はＸについてファイル型の解釈が得られた時に、『プリンターにＸを印刷する』、『Ｘの印刷装置への出力』などにマッチする。

標準パターンに書かれる依存構造のノードのタイプとしては以下のものが許される。

ａ．例示： 単語 ｜<単語 X>
　その単語と似た属性ベクトルの単語を持つノードとマッチする。
ｂ．正規パターン： <"パターン" X>
　その正規パターンに適合する単語を持つノードにマッチする。
ｃ．解釈： <解釈型 X>
　その型の解釈が既に得られているノードにマッチする。

変数Xには、例示または正規パターンの場合はマッチしたノードの単語が束縛され、型の場合にはマッチした解釈に束縛される。束縛された変数はルールの実行部で新たな解釈を生成するために用いられる。

付属語についてもマッチングの条件が書けるようになっている。付属語についての条件は、

ａ．例示： 格マーカー （が、を、に、から、 など）
　その格マーカーと意味的に矛盾しない付属語とマッチする
ｂ．文字列： "文字列"
　その文字列が付属語の中に含まれていればマッチする

のいずれかが許される。

４．２ ルールの適用

ルールのマッチングはそのルールの標準パターンのノードのすべてが入力依存構造中のノードに対応したときに成功する。マッチングのコストは各ノードのマッチングのコストの和に入力依存構造中の余ったノードにペナルティを掛けて加えたものである。ノードのマッチングのコストは次のように計算される。

ａ．ノードが例示のとき ．．． コストは単語間の距離
ｂ．ノードが正規パターンのとき ．．． マッチすればコスト＝０
　そうでなければ、条件部全体のマッチングは失敗
ｃ．ノードが型のとき ．．． コストはマッチした解釈のコスト

ある入力ノードの解釈は一般には複数のルールの適用によって得られる。例えば図６(a)という入力に対して、図６(b)と図６(c)という二つのルールがあったとしよう。

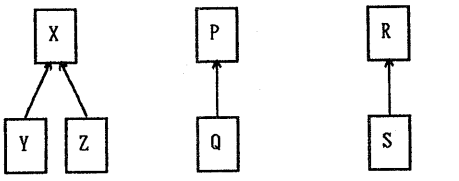

図６

この場合、ルール適用の可能性は、

1. X - Y <==> P - Q　　(ルール１を X - Y に適用)
2. X - Z <==> P - Q　　(ルール１を X - Z に適用)
3. X - Y <==> R - S　　(ルール２を X - Y に適用)
4. X - Z <==> R - S　　(ルール２を X - Z に適用)

の４とおりである。ノードＸにつけられる解釈は、これらのルール適用の組み合わせである。ノードＹまたはＺのどちらか一方がマッチしないままの組み合わせも許されるから（その場合、ペナルティが加算される）すべての組み合わせは、

{1}, {2}, {3}, {4}, {1,2}, {1,4}, {3,2}, {3,4}

の８とおりである（根を除く入力ノード、すなわちここではＹ，Ｚは、同時に二つのルール適用をうけてはならない。したがって、{1,3}, {2,4}, {1,3,4} ...などの組み合わせは除外される）。

このように比較的簡単な入力に対しても非常に多くの解釈の可能性が出てきてしまうため、ルール適用アルゴリズムは次のようになっている。

１．すべての可能なルール適用を求める。
２．それらのうち、コストの悪いものを、絶対的および相対的な閾値で、
　　切り捨てる。
３．残ったルール適用のすべての組み合わせを生成する。
４．それらのうち、コストの悪いものを、絶対的および相対的な閾値で、
　　切り捨てる。
５．残ったルール適用の組み合わせに対して、各々のルールの実行部を呼び出す。

ルールの実行部は領域に依存したデータ構造を作りだすため、一般には最適化がしにくい。実行部の呼出しを最小限におさえることによって、無駄な解釈の生成を少なくすることができる。

５．実行例

本システムは、ＶＭ／ＣＭＳ上のＬｉｓｐ／ＶＭおよびその上のオブジェクト指向言語ＬＯＯＸ[7]で書かれている。以下にＩＳＰＦ[8]によるメニューシステムとこの自然言語インターフェースを融合したシステムの実行例を示す。

図７は、トップレベルのメニューである。ユーザーは番号を選択して目指すアプリケーションに到達する事もできるが、この図に見られるように自分がしたい事を自然言語で入力する事もできる。この図は、『ＤＯＳの文書が欲しい。』というあいまいな文をユーザーが入力したところを示している。

```
------------------- 適用業務の選択 --------------------------------
 適用業務の番号を入れて下さい。          ユーザーＩＤ = WATANABE
                                         時刻         = 16:55
                                         端末の形式   = 3278KN
   0. ISPFパラメーターの設定
   1. ＣＭＳ機能の実行              (ファイル編集、検索ほか)
   2. ＰＳ／５５ユーティリティ      (コンフィギュレーターほか)
   3. データベース検索
   T. 解説
   X. 終了

 もしくは、やりたいことを日本語で下に入力して下さい。

 ===> ＤＯＳの文書が欲しい。
```

図７

図８に、この場合に使用されるアプリケーション『マニュアル注文』のルールの一部を示す。

```
%%% DOS Manual
((id manual-dos-1)
 (condition (<" (ＤＯＳ｜ｄｏｓ｜ＯＳ｜ｏｓ｜オペレーティングシステム) " x>))
 (action (new-interpretation 'ps55:manual 'manual-name 'dos))
 (cost 2.0))

%%% DOS-JWP Manual
((id manual-dosjwp-1)
 (condition (<" (ＤＯＳ｜ｄｏｓ) " x> "" <" (文書｜ワープロ) " y>))
 (action (new-interpretation 'ps55:manual 'manual-name 'dosjwp))
 (cost 2.0))

%%% ～のマニュアル
((id manual-2)
 (condition (<ps55:manual x> "" マニュアル))
 (action (set-interpretation x)(penalty -2.0))
 (cost 0.1))

%%% 動詞パターン
((id . manual-hosii)
 (condition . (<ps55:manual x> "" 注文))
 (action . (set-interpretation x))
 (cost 0.1))
```

図８ アプリケーション『マニュアル注文』のルールの一部

これらのルールからは、例えば次のような標準表現が得られる。[]はその標準表現を作った標準パターンのルールについているコストの和であり、その標準表現とマッチした時のデフォールトのコストを表している。実際のコストはこれに構造的類似度と対応する単語の類似度を加えたものである。

『ＤＯＳのマニュアルを注文したい。』[cost = 0.2]
『ＤＯＳを注文したい。』[cost = 2.1]　...（マニュアルが省略された表現）
『ＤＯＳ文書のマニュアルを注文したい。』[cost = 0.2]
『ＤＯＳ文書を注文したい。』[cost = 2.1]　...（マニュアルが省略された表現）

図９は、ユーザーからの入力に対して意味的に近い標準表現を持ったアプリケーションを提示したパネルである。それぞれの解釈ごとに、アプリケーションがユーザーの入力をどのように理解したかを示す説明が付随している。ここで、確信度とはコストのことであり、コストの低いものほど確信度は高い。

```
入力文: ＤＯＳの文書が欲しい。
あなたの入力は以下のように解釈されました。

解釈１．　確信度＝4.7 ．　アプリケーション：PS/55 Manual Order
日本語ＤＯＳのマニュアルを注文したい。
解釈２．　確信度＝6.1 ．　アプリケーション：PS/55 Configulator
ＤＯＳ文書プログラムを使いたい。
解釈３．　確信度＝6.6 ．　アプリケーション：PS/55 Manual Order
ＤＯＳ文書プログラムのマニュアルを注文したい。
解釈４．　確信度＝8.999 ．　アプリケーション：CMSFUNC
ファイルの編集

どれですか　======>
数字を入れて下さい。次頁はＰＦ８、前頁はＰＦ７、キャンセルはＰＦ３。
```

図９

図１０は、図９で解釈１を選択した場合の『マニュアル注文』というアプリケーションのパネルである。ユーザーの入力から有効なデータが予め取り出されてパネルのデータ領域に代入されている。

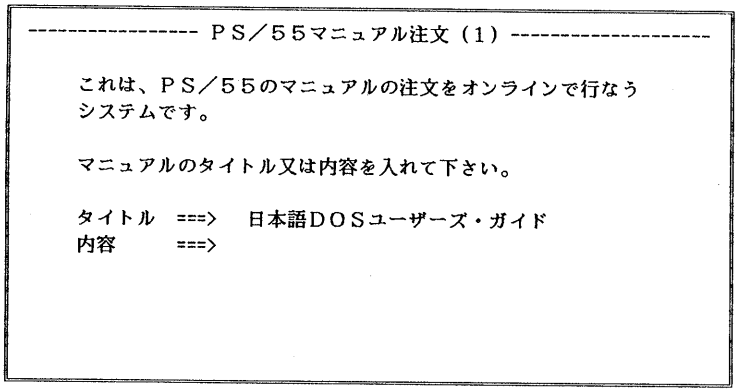

図１０

このシステムによれば、ユーザーはメニューを選択していく事によって目指すアプリケーションに到達する事ができると同時に、自分がしたい事を自然言語で入力する事によっても目指すアプリケーションに到達する事が可能である。

６．おわりに

ユーザーの入力に一番意味的に近いパターンを見付けると言う"Full-language Approach"によって、複数のアプリケーションと応答可能なかなり柔軟な自然言語インターフェース・システムを構築する事ができた。これを用いれば、アプリケーション側の期待するパターンとしては、自然言語に近い柔軟なもので良く、カスタマイズの手間も相当軽減する事が可能である。
現在の問題点は、意味的近似度を計算するに際して内部表現を概念依存構造としているために、意味が同じでも概念依存構造のレベルでは異なる構造である場合に、近似度が正しく計算されない点である。よりよい精度で近似度を求めるためには、より柔軟なマッチングの手法が必要となるであろう。

参考文献

[1]『日本語情報処理』、近代科学社、pp.205~252
[2] David H.D. Warren and Fernando C.N. Pereira,
"An Efficient Easily Adaptable System for Interpreting Natural Language Queries",
AJCL, Vol.8, No.3-4, 1982
[3] Gray G. Hendrix, "Natural-Language interface", AJCL, Vol.8, No.2, 1982
[4] 丸山、渡辺、『既存のメニューシステムと自然言語インターフェースの融合』、日本ソフトウェア科学会第４回大会論文集、1987
[5] 丸山、渡辺、『ビットベクトルによる単語間の意味の近さの近似法』、情報処理学会第３６回全国大会論文集、1988
[6] N.Maruyama, M.Morohashi, E.Sumita, S.Umeda, "A Japanese Sentence Parser",
IBM Journal of R&D, Jan. 1988
[7] 渡辺、『Lisp/VM上のオブジェクト指向言語LOOXにおける多重メソッドの探索法および最適化について』、情報処理学会第３５回全国大会論文集、1987
[8] "Interactive System Productivity Facility",IBM,SC34-2137,1984