TR-I-0311

言語表現間意味距離計算プログラム説明書
User's Guide: Semantic Distance Calculator

隅田　英一郎　　林　明徳*
Eiichiro SUMITA　　Akinori HAYASHI

1993.3

概要

ATRでは、シソーラスを用いて言語表現間の意味的距離を計算する手法を提案している。我々は、同手法を、用例を使った自然言語処理（機能語などの対訳決定や構造的曖昧性解消）に応用した。従来手法より高い精度が実験により確認できた。この結果は、シソーラスが、単語と違って細か過ぎず、意味マーカと違って粗すぎず、適切な意味の粒度を与えているからだと考えられる。

このマニュアルでは上述の言語表現間意味距離計算プログラムのユーザーズ・ガイドであり、言語表現間意味距離計算の概要、プログラムのインストール方法、必要な知識（シソーラスと用例）のフォーマット、プログラムの実行方法について説明する。

ATR自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

---

*株式会社東洋情報システム

目　　次

はじめに

はじめに

言語表現間意味距離計算プログラムは、言語表現間の意味距離を計算するものです。類似の用例を用例データベースから検索することにより、翻訳と係り先の曖昧性解消ができます。このパッケージが提供するのは、以下のものです。

- ●言語表現間意味距離計算を使った検索システム
- ●小規模な用例データベース
- ●小規模なシソーラス
- ●プログラムの実行に必要なユーザー定義ファイルのサンプル

翻訳用と係り受けの曖昧性解消用の２種類の検索関数を提供します。このパッケージをインストールするだけで、実際に、これらの検索関数を実行してみることができますが、実用的なシステムを構築するためには、各ユーザーが、ここで提供されている知識を参考にして、独自の用例データベース・シソーラス・ユーザー定義ファイルを作る必要があります。

このマニュアルは、Lispの基本的な知識を持った読者を対象としており、プログラムのインストールの方法と、翻訳と係り受けの曖昧性解消での利用方法について説明しています。本書の構成は次のとおりです。

1　言語表現間意味距離計算プログラム概要

このパッケージを利用するのに、最低限必要と思われる範囲で、言語表現間意味距離計算について説明してあります。

2　インストール

インストールに必要な環境や具体的なインストールの方法について説明してあります。

3　シソーラス・用例データベース

実行に必要なシソーラス・用例データベースの構築方法とユーザー定義ファイルの書き方について説明してあります。

4　実行方法

ロードや、検索関数の実行方法について説明してあります。

なお、以下の説明では、英日方向の翻訳を例にとって説明していますが、本システムを使って、日英方向でも同様のことができます。

1　言語表現間意味距離計算プログラム概要

1　言語表現間意味距離計算プログラム概要

ここでは、このパッケージを使うに当たって、最低限必要な知識についてだけ簡単に説明します。１．１では、言語表現間意味距離計算について、１．２では、類似の用例を検索するための類似度→距離を計算する方法について説明します。なお、言語表現間意味距離計算に関して詳しく知りたい方は、下記の参考文献を参考にしてください。

（参考文献）

- Sumita, E. and Iida, H. : "Example-Based Transfer of Japanese Adnominal Particles into English", IEICE TRANS. INF. & SYST., VOL. E75-D, No. 4, pp.585-594, (1992).

- Sumita, E. and Iida, H. : "Example-Based NLP Techniques - A Case Study of Machine Translation -", Statistically-Based Natural Language Processing Techniques-Papers from the 1992 Workshop, Technical Report W/92-01, AAAI Press, (1992).

- 隅田 英一郎，土井 伸一，飯田 仁，山端 潔: "用例に基づいて英語前置詞句の係り先決定を行う英日翻訳システム，"情報処理学会第46回全国大会5B-2, (1993).

１．１　言語表現間意味距離計算プログラムとその応用

言語表現間意味距離計算プログラムは、

- ●言語表現間意味距離計算を使った検索システム
- ●用例データベース
- ●シソーラス

の３つの要素で構成されます。

言語表現間意味距離計算プログラムでは、シソーラス上の距離（「１．２　距離計算の方法」参照）を利用して入力と各用例の類似度（以下この類似度のことを入力と用例の距離と呼びます。距離が近いほど類似度は高いことになります）を計算し、用例データベースから類似の用例を検索します。そして、その用例の入力との距離、翻訳パターン※を用例とともに返します。次ページ図１に言語表現間意味距離計算プログラムの流れを示します。

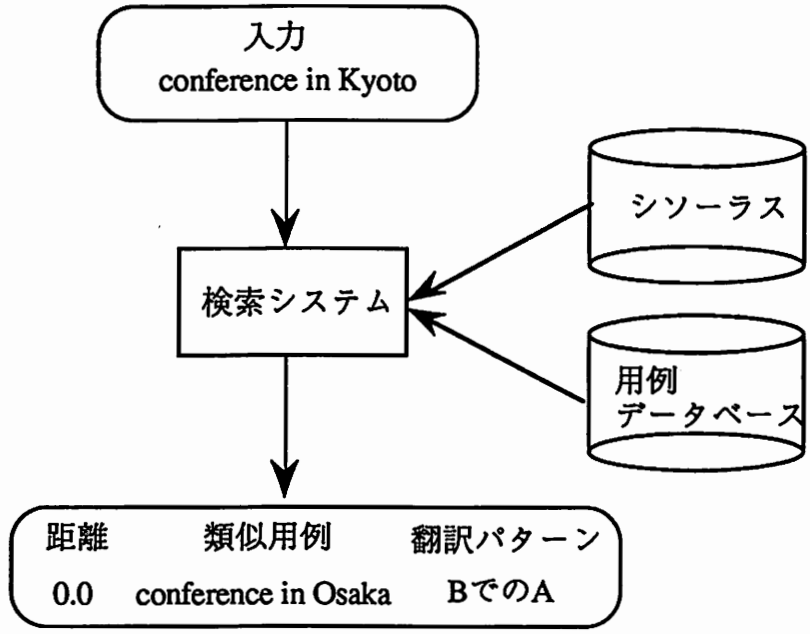

図１　言語表現間意味距離計算プログラムの流れ

※翻訳パターン

翻訳パターンとは、どのような形で用例を翻訳するかを抽象的な形で表現したものです。

(例)	用例		訳		翻訳パターン
・conference in Kyoto		--->	京都での会議		BでのA
	A B		B	A	
・brochure on Hakone		--->	箱根のパンフレット		BのA
	A B		B	A	

言語表現間意味距離計算プログラムを用いることにより、次のようなことが可能になります。

(1) 検索システムの返す翻訳パターンを使って、入力を翻訳する。

言語表現間意味距離計算プログラムでは、検索された類似用例とともに、その翻訳パターンが返されます。したがって、その翻訳パターンと変換辞書（ユーザーが準備する必要があります）を使い、入力を翻訳することができます。

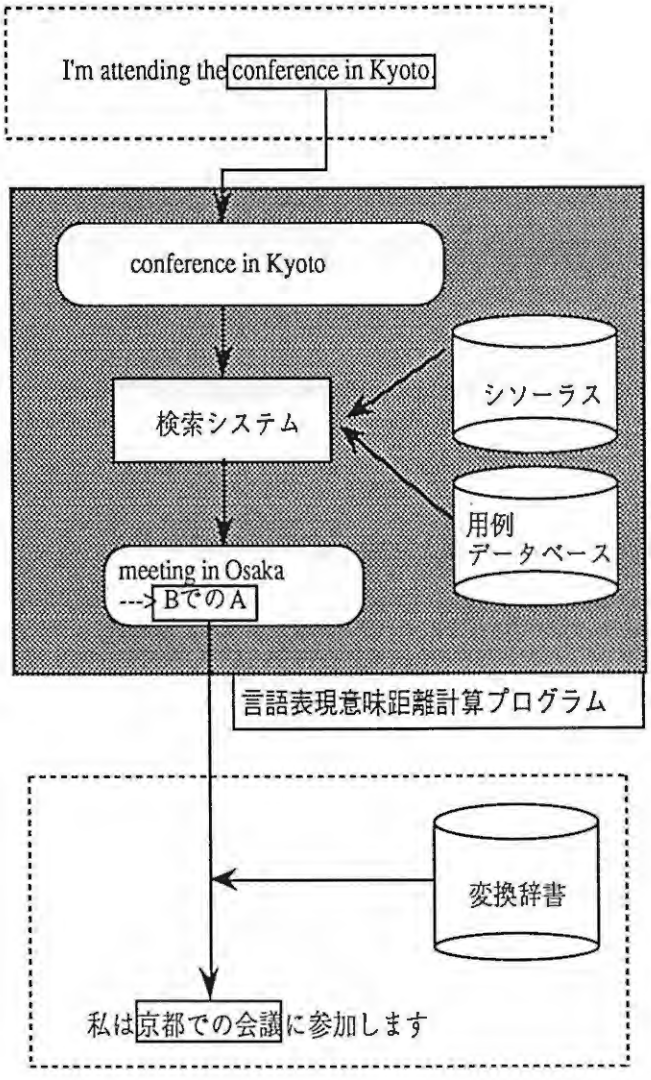

図２　言語表現間意味距離計算プログラムを使った翻訳のイメージ

(2) 係り受けが曖昧な時、検索システムの返す距離を利用して係り先を決定する。

言語表現間意味距離計算プログラムでは、検索された類似用例とともに、入力との距離が返されます。前置詞句の係り受けが曖昧な場合なら、係り先の各候補と前置詞句の組み合わせで検索を実行することにより、各候補と前置詞句の結び付きの強さを距離を使って比較でき、曖昧性の解消に役立てることができます。

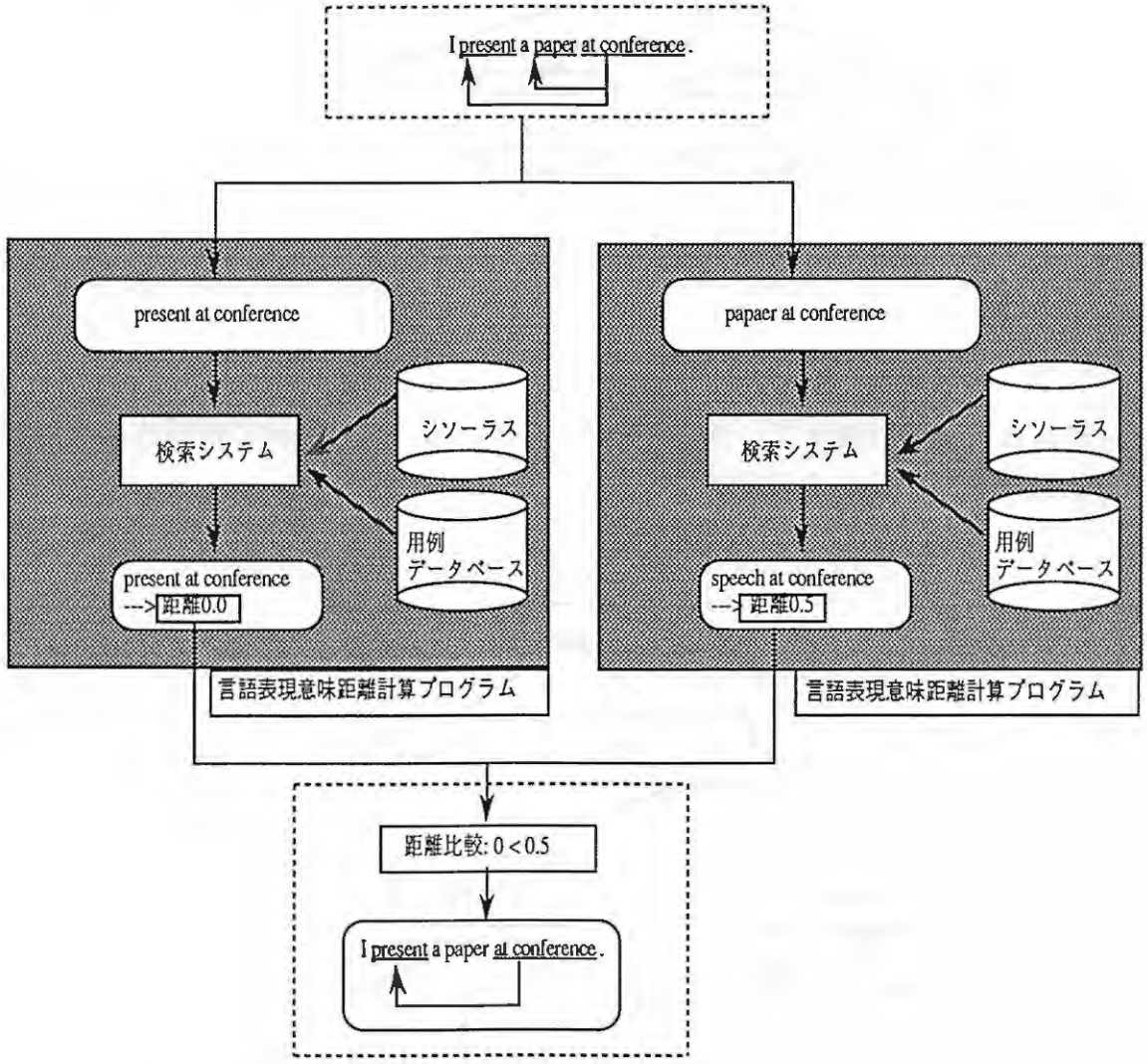

図３　言語表現間意味距離計算プログラムを使った
係り受けの曖昧性解消のイメージ

用例データベースは、"「BofA」という形の名詞句"、"「NのN」という形の名詞句"といった特定のタイプの用例とその翻訳パターンの対を１レコードとして、用例のタイプごとに集めたものです。

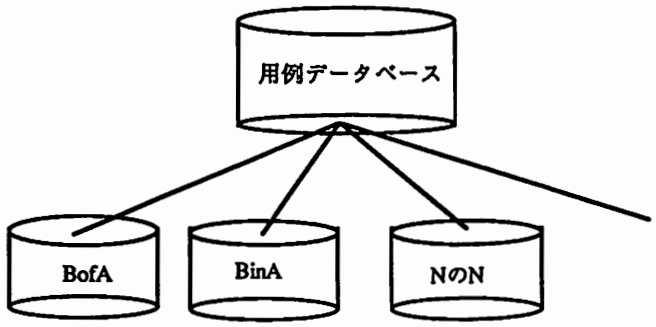

図４　用例データベース

シソーラスは、共通の特徴をもつ単語をグループ化し、ツリー状に階層化した辞書です。図５に英語シソーラスの例を示します。

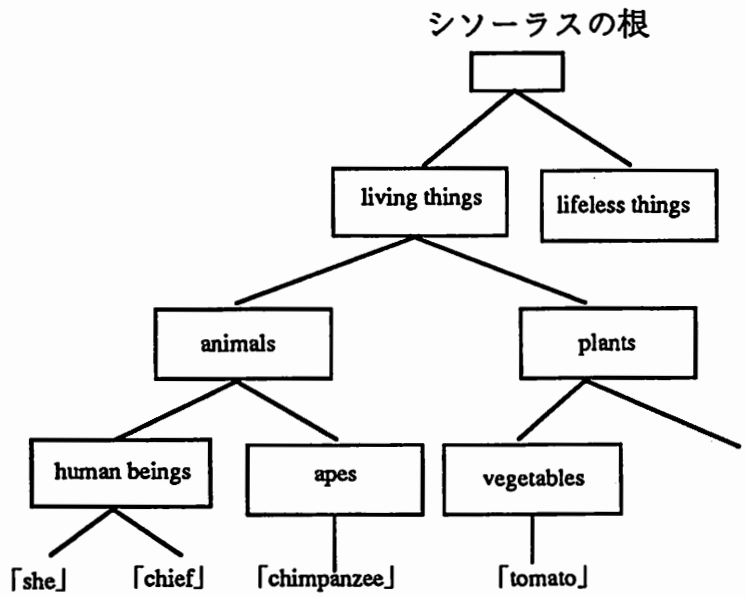

図５　英語シソーラスの例

１．２　距離計算の方法

言語表現間意味距離計算プログラムでは、用例をいくつかの要素に分割し（たとえば単語単位に分割する）、対応する要素を比較して距離を計算します。例として、

入力　＝　"conference in Kyoto"
用例　＝　"company in Tokyo"

という2つ名詞句の距離を考えてみます。それぞれを単語単位に分割すると、

入力　＝　("conference" "in" "Kyoto")
用例　＝　("company" "in" "Tokyo")

となります。対応する要素は、

名詞1　"conference"－"company"
前置詞　"in"－"in"
名詞2　"Kyoto"－"Tokyo"

です。そこで、"conference"と"company"、"in"と"in"、"Kyoto"と"Tokyo"の距離をそれぞれ比較し、それを合計したものを入力と用例の距離とします。

対応する要素間の距離の定義方法には、

●単純に文字列の一致・不一致だけで決める（これをSTR型と定義します）
●辞書引きしてシソーラス上の距離を用いる（これをWORD型と定義します）

という２通りがあります。上例では、名詞1と名詞2はWORD型、前置詞はSTR型です。

正確には、入力と用例の距離は、次の式により計算されます。ただし、Ii・Eiはそれぞれ入力と用例でi番目の要素、またWiは要素iに対する重みです。

$$d(I, E) = \sum_{i} d(Ii, Ei) \times Wi$$

WORD型、STR型の要素の距離d(Ii, Ei) は、それぞれ次のように計算します。

(1) WORD型

入力と用例の要素iの値に対応するシソーラス上の概念間の距離によって定義します。概念間の距離はシソーラスにおける最小の共通上位概念の位置にしたがって0～1までの値にします。値0は2つの概念が同じであることを意味し、値1は無関係であることを意味します。シソーラスの階層数が(n+1)階層であるとすると、下から0、1/n、2/n、・・・、1を距離として割り当てます。

言語表現間意味距離計算プログラムでは、各概念にシソーラス上の位置にしたがって数字化した意味コードを割り当てており、この意味コードを使って距離を計算します。

図６は、既出のシソーラスの例を使って、距離の定義を示したものです。

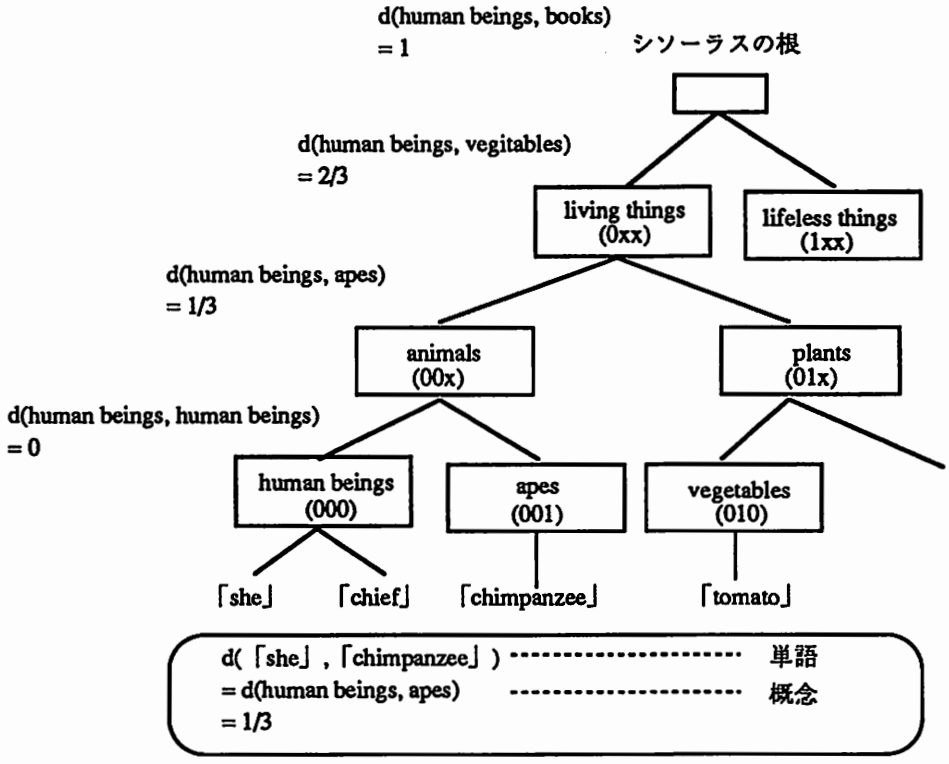

図６　シソーラス上の距離
（概念の下の数字は意味コード）

(2) STR型

文字列が一致したら0、文字列が不一致ならば1を割り当てます。

なお、高速化のため、類似用例の検索は実際には、次の3つのステップに分けて行っています。

(1) ハッシングを使った字面マッチの用例の検索

該当するものがあれば、その用例に関する情報を検索結果として返します。該当するものがなければ(2)を実行します。

(2) ハッシングを使った意味コードマッチの用例の検索

該当するものがあれば、その用例に関する情報を検索結果として返します。該当するものがなければ(3)を実行します。

(3) 部分一致の全件検索

距離の小さいものから、指定された数（「４．４．２　類似用例の検索数」参照）までの用例に関する情報を検索結果として返します。

2 インストール

2　インストール

ここでは、このパッケージに含まれているソフトウェアの内容やインストールに必要な環境、具体的なインストールの手順等について説明します。

2．1　言語表現間意味距離計算プログラムパッケージについて

言語表現間意味距離計算プログラムパッケージは、距離計算を使って用例データベースから入力と類似の用例を検索し、その用例の翻訳パターンと距離を用例とともに返す機能を提供します。この検索機能は、翻訳用と係り受けの曖昧性解消用の２種類の関数の形で使用します。このパッケージをインストールするだけで、実際にこれらの関数を実行してみることができますが、実用的なシステムを構築するためには、ユーザーがここで提供されている知識を参考にして、独自にシソーラス・用例データベース・ユーザー定義ファイルを作る必要があります。

用例データベースは、日英または英日のいずれか、または両方が必要です（それ以外の言語ペアはサポートしていません）。また、日英の用例データベースを使用する場合には、日本語シソーラス、英日の用例データベースを使用する場合には英語シソーラスが必要です。さらに、これらを使用するためのユーザー定義ファイルを用意する必要があります。これらに関する具体的な手順については、「3　シソーラスと用例データベース」を参照してください。

この章では、本パッケージの内容およびインストールに必要な項目について説明します。

2．1．1　パッケージの内容

本パッケージは、Lucid Common Lisp™/SPARC™のバージョン4.0.0以上で動作するSun4およびSPARC用のパッケージになっています。

付属のテープに収められているソフトウェアの概要は次のとおりです。

- ●距離計算を使った検索
- ●小規模な用例データベース
- ●小規模なシソーラス
- ●プログラムの実行に必要なユーザー定義ファイルのサンプル

２．１．２　インストールに必要な環境

本パッケージをインストールし、実行するには、次のようなハードウェア、ソフトウェア環境が必要になります。インストール作業を行う前に、インストールするシステムを確認してください。

マシン

SunOS4.1が動作するSun Microsystems Sun4、SPARCstationシリーズまたは SPARCserverシリーズおよびこれらの100%互換機。

ソフトウェア

Lucid Common Lisp™/SPARC™のバージョン4.0.0以上が、インストールされている必要があります。また、日本語環境が整っているかどうかも確認してください。

カートリッジ・テープデバイス

ディストリビューション・テープは、1/4カートリッジ・テープですから、このテープを読み込むデバイスが必要になります。

ディスクの空き領域

言語表現間意味距離計算プログラムのパッケージをインストールするだけで、約55kバイトを必要とします。実際には、この他に、言語表現間意味距離計算プログラムで使用する用例データベースとシソーラス（各ユーザーが、個別に構築します）の容量を見積って置く必要があります。

2．2　インストールの手順

言語表現間意味距離計算プログラムをインストールするために必要な環境やディスクの空き容量について確認したら、以下に述べる手順でインストール作業を始めます。

2．2．1　ログイン

言語表現間意味距離計算プログラムのインストールは、システムのユーザーが行います。まず、ユーザーのアカウントでインストールするマシンにログインしてください。次例は、アカウントがEBMTの場合です。

(例)

```
Login: EBMT
Password: (EBMTのパスワードを入力)
```

2．2．2　言語表現間意味距離計算プログラム用ディレクトリの作成

ログインしたら、まず言語表現間意味距離計算プログラム用のディレクトリ"EBMT"を、適当な場所に作成します。たとえば、/usr/atrの下にディレクトリを作るのであれば、

```
% cd /usr/atr
% mkdir EBMT
%
```

とします。本パッケージの内容は、すべてこのディレクトリの下にコピーします。

2．2．3　テープからのコピー

ディストリビューション・テープの内容は、tarフォーマットで収められています。

ここでは、ローカル・ドライブを使ってインストールする方法を説明します。リモート・ドライブを使用する場合には、ここで説明する方法に沿って、rshコマンドを使ってインストールしてください。

ディストリビューション・テープを正しくセットし、ファイルをコピーします。ディストリビューション・テープ内のファイルは、tarフォーマットですから、tarコマンドのxオプションを使用します。指定するデバイス名は/dev/rst8または/dev/rst0になります。

```
% pwd
/usr/atr/EBMT
% tar xvf /dev/rst8
    (コピーしたファイルの情報)
    . . . . . .
%
```

2．2．4　ディレクトリ構成

以上の手順を終了すると、次のようなファイルが作成されています。

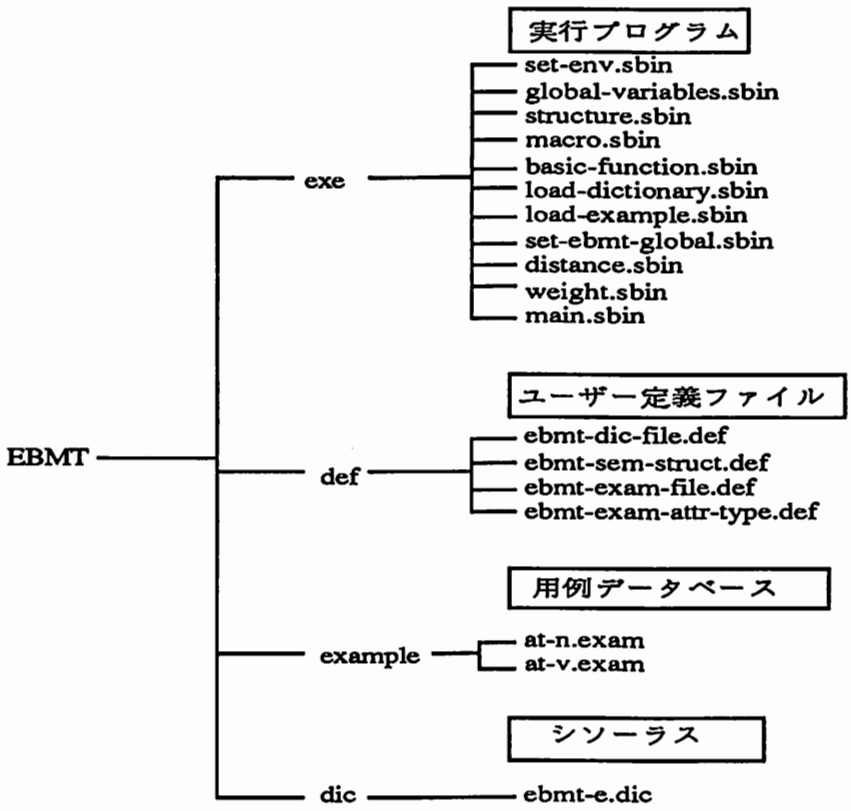

図7 インストール後のディレクトリ構成

3　シソーラスと用例データベース

3　シソーラスと用例データベース

言語表現間意味距離計算プログラムのパッケージでは、シソーラスと用例データベースについては、非常に小規模なものしか提供していません。したがって、このシステムを実用的なものにするためには、以下に述べる規約に従って、各ユーザーが、独自のシソーラスと用例データベースを構築する必要があります。また、構築したシソーラスおよび用例データベースを使って、言語表現間意味距離計算プログラムを実行するためには、あらかじめ以下の設定をしておく必要があります。

- ●シソーラスのファイル指定
- ●シソーラスの意味コードのフォーマット指定
- ●用例データベースのファイル指定
- ●用例の要素のタイプ指定

ここでは、これらについて説明します。

3．1　シソーラス

シソーラスは、共通の特徴をもつ単語をグループ化し、ツリー状に階層化した辞書で、各エントリーは概念を数字化した意味コードを持ちます。言語表現間意味距離計算プログラムでは、日英の用例データベースを使用する場合日本語シソーラス、英日の用例データベースを使用する場合英語シソーラスが必要になります。このパッケージには、小規模な英語シソーラスが含まれており、これを使って検索システムを稼働させることはできますが、実用的なシステムにするためには、ユーザーがここに述べる方法にしたがって、独自にシソーラスを作る必要があります。

3．1．1　シソーラスのフォーマット

シソーラスの各レコードは、次のフォーマットに従って記述します。

("見出し語" (意味コード1　意味コード2・・・・意味コードn))

見出し語は、任意のストリングです。
意味コードは、シソーラス上の各概念を数字で表したもので、１つの見出し語に対して１つ以上の任意の数だけ与えることができます。

下の例では、"会議"は、123、507、517という３つの意味コードを持つ多義語であり、"京都"は、709という意味コードを１つだけ持つことを表しています。また、意味コードは、１バイト文字のみから成る数字列として記述しなければなりません。

（例）
("会議" (123　507　517))
("京都" (709))

３．１．２　意味コードのフォーマット

意味コードは次のようなフォーマットを持つものとします。

(1) 階層

シソーラスの階層（「１．２　距離計算の方法」参照）は、2以上の任意の正の整数個設定することができます。ただし、階層の深さは、シソーラス全体で均一でなければなりません。したがって、次のように、場所によって深さの異なる構造は認められません。

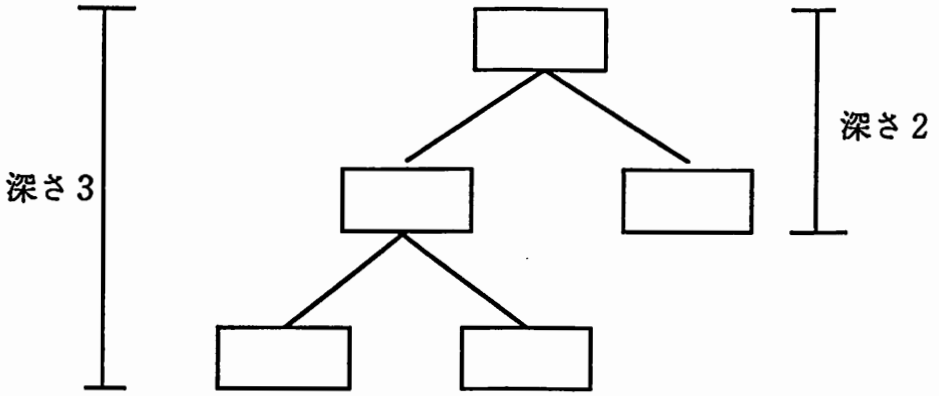

図８　不適当なシソーラスの例

(2) ノードの枝分れの数

シソーラスでは、ノードごとに下位分類への枝分れが起こります。それぞれのノードでいくつの下位分類に枝分れするかは、ユーザーが定義する（「３．１．４　意味コードのフォーマット指定」参照）枝分れの最大数10^nを超えない範囲で自由に決められます。実際のコードの記述では、すべての階層のコードを10進n桁で記述します。コードがn桁に満たない場合には、空の桁を0で埋める必要があります。たとえば、最大数を$10^3 = 1000$とした場合、各階層のコードは、000、001、002、・・・・・、999といった形で記述することになります。

次に例を示します。()内は、わかりやすいよう各階層の区切り（分類のレベル）を示したものです。階層数は、区切りの数より1つ多くなる（ルートを１階層として数えるため）ので注意してください。

・階層＝4、枝分れの最大数＝10^1 ＝10
---> 123(1¦2¦3)、507(5¦0¦7)

・階層＝4、枝分れの最大数＝10^3 ＝1000
---> 124033609(124¦033¦609)、13159034(13¦159¦034)

・階層＝3、枝分れの最大数＝10^5 ＝100000
---> 3244500910(32445¦00910)、50807765(508¦97765)

なお、"~EBMT/dic"の下に、"ebmt-e.dic"というサンプル・ファイルがあります。これを参考にして、各自のシソーラスを構築してください。

３．１．３　シソーラスのファイル指定

構築したシソーラスを言語表現間意味距離計算プログラムで使用するためには、"~EBMT/def"の下の"ebmt-dic-file.def"という名前のファイルに、言語表現間意味距離計算プログラムで使用するシソーラスのパス名と言語を指定する必要があります。

指定の方法は、各言語ごとに

```
(define-dic-file　言語　シソーラスのパス名)
```

と記述します。言語は、必ず

```
日本語　----->　J
英語　　----->　E
```

のいずれかを指定します。これ以外の指定があった場合は、システムのロード時にエラーメッセージが出力され、ロードは中止されます。

たとえば、日本語シソーラス・英語シソーラスの２つのシソーラスがあり（実際にはどちらか１つでもかまいません）、それぞれ次のようであったとします。

言語	シソーラスのパス名
日本語	"/usr/atr/EBMT/dic/ebmt-j.dic"
英語	"/usr/atr/EBMT/dic/ebmt-e.dic"

この時、"ebmt-dic-file.def"の内容は、次のようになります。

```
(define-dic-file　J　"/usr/atr/EBMT/dic/ebmt-j.dic")
(define-dic-file　E　"/usr/atr/EBMT/dic/ebmt-e.dic")
```

なお、"~EBMT/def"には、"ebmt-dic-file.defというサンプル・ファイルがあります。これを参考にして各自の"ebmt-dic-file.def"を編集してください。

３．１．４　意味コードのフォーマット指定

次に、"~EBMT/def"の下の"ebmt-sem-struct.def"という名前のファイルに意味コードの階層数と枝分れの最大数を指定します。枝分れの最大数は、10ⁿ という形で考え、このファイルではnを指定します（「３．１．２　シソーラスの意味コードのフォーマット」参照）。たとえば、枝分れの最大数を10^3 = 1000にしたいなら、nに3を指定します。

指定の方法は、各言語ごとに

```
(define-sem-struct　言語　階層数　n)
```

と記述します。言語については、「３．１．２　シソーラスのファイル指定」を参照してください。階層数は2以上の任意の正の整数、nは正の整数です。

たとえば、日本語シソーラスの階層が4、枝分れの最大数が$10^1 = 10$、英語シソーラスの階層が4、枝分れの最大数が$10^3 = 1000$であったとすると、このファイルの内容は次のようになります。

```
(define-sem-struct　J　4　1)
(define-sem-struct　E　4　3)
```

階層数は分類のレベルの数より１つ多くなる（ルートを1階層と数えるため）ので注意してください。たとえば、大分類・中分類・小分類の3つのレベルを持つシソーラスの場合、階層数は4になります。

なお、"~EBMT/def"には、"ebmt-sem-struct.def"というサンプル・ファイルがあります。これを参考にして自分の"ebmt-sem-struct.def"を編集してください。

3．2　用例データベース

用例データベースは、日本語または英語の用例とその翻訳パターン（原言語が日本語なら英訳のパターン、原言語が英語なら和訳のパターン）を１組とするレコードを集めたファイルです。

言語表現間意味距離計算プログラムでは、日英・英日のいずれか、または両方の用例データベースが必要です。このパッケージには、小規模な英日の用例データベースが含まれており、これを使ってシステムを稼働することはできますが、実用的なシステムにするためには、ここで述べる方法にしたがって、ユーザーが独自に用例データベースを作る必要があります。

3．2．1　用例データベースのフォーマット

各レコードは、次のフォーマットに従って記述します（次ページに例があります）。

```
(用例　翻訳パターン)
```

係り受けの曖昧性解消にのみ用いる場合は、翻訳パターンに、適当なダミーデータを入れてください。

用例は次の形で記述します。

```
("要素1"　"要素2"　・・・・・・・"要素n")
```

すなわち用例は、任意の数のストリングから成るリストとして記述します。ただし、同じ用例タイプ※の用例は、すべて同じ要素数でなければなりません。

※用例タイプ

用例タイプとは、「名詞　at　名詞」という形の名詞句を集めた用例データベースならat-n、「動詞　in　名詞」という形の動詞句を集めた用例データベースならin-vというように、用例データベースごとにユーザーが与える任意の名称のことです。

たとえば、(動詞　at　名詞)という３つの要素から成る用例タイプat-vを考えます。この時、用例の記述は次のようになります。

（例）

○present at conference　--->　("present" "at" "conference")
○get at station　--->　("get" "at" "station")
○arrive at 7:00　--->　("arrive" "at" "7:00")

次のような用例は、要素数が３つでないので、上記の用例タイプat-vの用例に混入させることはできません。

（例）

×present paper at conference --->　("present" "paper" "at" "conference")

また、翻訳パターン（p.５参照）は、記号・ストリング・リスト等Lispでひとまとまりのデータとして扱われるような形で記述してください。

（例）

○BのA、BでのA、"BのA"、"B　の　A"、(BのA)、(B　の　A)

×B　の　A、B　での　A

次に、先に例に上げた用例タイプat-vの用例データベースの例を示します。

（例）

(("present"　"at"　"conference")　Bで)

(("get"　"at"　"station")　Bに)

(("arrive"　"at"　"7:00")　Bに)

なお、"~EBMT/example"の下に、"at-n.exam"、"at-v.exam"という用例データベースのサンプルがあります。これを参考にして、各自の用例データベースを構築してください。

３．２．２　用例データベースのファイル指定

構築した用例データベースを言語表現間意味距離計算プログラムで使用するためには、"~EBMT/def"の下の"ebmt-exam-file.def"という名前のファイルに、言語表現間意味距離計算プログラムで使用する用例データベースのパス名と言語ペアを、用例タイプごとに指定する必要があります。

指定方法は、各用例タイプごとに、

(define-example-file　言語ペア　用例タイプ　用例データベースのパス名)

と記述します。

ここで、言語ペアとは、日英または英日という翻訳の方向のことで、必ず

日英 ---> JE
英日 ---> EJ

のいずれか一方を指定します。これ以外の指定があった場合は、システムのロード時にエラーメッセージが出力され、ロードは中止されます。

用例タイプは、「名詞 at 名詞」という形の名詞句を集めた用例データベースならat-n、「動詞 in 名詞」という形の動詞句を集めた用例データベースならin-vというように、用例データベースごとにユーザーが与える名前のことです。用例タイプには、各用例データベースを一意に識別するため、ユーザーが用例データベースごとに任意の名前を与えます。これは、必ずLispで記号として認識される形で書いてください。

(例)

○in-n、at-v、test、xxx

×at n、"at-n"、(xxx)

用例データベースのパス名には、用例データベースの所在をフルパスで指定します。

たとえば、使用する用例データベースの用例タイプ、言語ペア、用例データベースの所在が次のようであったとします。

用例タイプ	言語ペア	用例データベースの所在
NnoN	日英	"/usr/atr/EBMT/example/NnoN.exam"
at-n	英日	"/usr/atr/EBMT/example/at-n.exam"
in-n	英日	"/usr/atr/EBMT/example/in-n.exam"

この場合、"ebmt-exam-file.def"の内容は次のようになります。

```
(define-example-file JE NnoN "/usr/atr/EBMT/example/NnoN.exam")
(define-example-file EJ at-n "/usr/atr/EBMT/example/at-n.exam")
(define-example-file EJ in-n "/usr/atr/EBMT/example/in-n.exam")
```

なお、"~EBMT/def"には、"ebmt-exam-file.def"というサンプル・ファイルがあります。これを参考にして、"ebmt-exam-file.def"を編集してください。

３．２．３　用例の要素のタイプ指定

最後に、"~EBMT/def"の下の"ebmt-exam-attr-type.def"というファイルに各用例データベースの用例の各要素のタイプを指定します。
指定の方法は、各用例データベースごとに、

(define-example-attr-type　言語ペア　用例タイプ　要素のタイプリスト)

と記述します。言語ペアと用例タイプについては、「３．２．２　用例データベースのファイル指定」を参照してください。
ここでは、要素のタイプリストについて説明します。ここでいう要素とは、("present" "at" "conference")という用例の"present"、"at"、"conference"に当たるものです。要素のタイプリストでは、この"present"、"at"、"conference"をそれぞれWORD型にするかSTR型にするかを指定します。これは、次のような形で指定します。

((要素名1　タイプ)(要素名2　タイプ)・・・・・(要素名n　タイプ))

要素名には、各要素を一意に識別するための任意の名前を与えます。要素名の順序は、実際の用例データベースの用例中での順序と対応していなければなりません。先の例では、"present"に当たるもの（動詞）が要素1、"at"に当たるもの（前置詞）が要素2、"conference"に当たるもの（名詞）が要素3になります。
タイプには、距離計算の際の扱いによって、

・WORD型　～　辞書引きにいき、シソーラス上の距離を使用 ・STR型　～　文字列の一致・不一致だけで0または1を割り当てる

の２種類があります（「１．２　距離計算の方法」参照）。タイプ指定する際には、必ず1byte文字で

・WORD型　----->　WORD ・STR型　----->　STR

と記述してください。これ以外のタイプを指定した場合は、システムのロード時にエラーメッセージが出力され、ロードは中止されます。

例として(動詞　at　名詞)という形の用例を持つ、at-vという用例タイプの用例データベースについて、ebmt-exam-attr-type.defを作成してみます。この用例データベースの中味は次のようなものです。

(("present" "at" "conference")　Bで)
(("get" "at" "station")　Bに)
(("arrive" "at" "7:00")　Bに)
・・・・・・・・・・・・・

動詞と名詞は、シソーラスの情報を使いたいのでWORD型、前置詞は単純な文字列の比較を使うSTR型に設定することにします。この時、ebmt-exam-attr-type.defの内容は、次のようになります。

```
(define-exam-attr-type　EJ　at-v　((VERB　WORD) (AT　STR) (NOUN　WORD))
```

VERB、AT、NOUNは任意の要素名ですから、別の名前を使っても、もちろんさしつかえありません。

なお、"~EBMT/def"には、"ebmt-exam-attr-type.def"というサンプル・ファイルがあります。これを参考にして自分の"ebmt-exam-attr-type.def"を編集してください。

4 実　　行　　方　　法

4　実行方法

ここまでに説明した手続きで、システムを稼働する環境が整いました。ここでは、実際にプログラムをロードし、翻訳用と係り受けの曖昧性解消用の２種類の検索関数を実行する方法について説明します。

4．1　プログラムのロード

(1) ログイン

まず、ユーザーのアカウントで言語表現間意味距離計算プログラムをインストールしてあるマシンにログインしてください。

(2) Lispの起動

次にshellまたはemacsからLispを起動します。

(3) パッケージの変更

Lispが立ち上がったら、次のようにパッケージをebmtに変更します。

```
> (in-package 'ebmt)
#<Package "EBMT" 63390E
>
```

(4) プログラムの存在するディレクトリに移動

次に、"~EBMT/exe" にcdコマンドで移動します。

```
> (pwd)
#P"/usr/EBMT"
> (cd "exe")
#P"/usr/EBMT/exe/"
>
```

(5) プログラムのロード

ここで、

```
> (load "set-env")
```

とすると、次のようなメッセージが表示され、プログラムがロードされます。

```
> (load "set-env")
;;; Loading binary file "set-env.sbin"
;;; Loading binary file "global-variables.sbin"
;;; Loading binary file "structure.sbin"
;;; Loading binary file "macro.sbin"
;;; Loading binary file "basic-function.sbin"
;;; Loading binary file "load-dictionary.sbin"
;;; Loading binary file "load-example.sbin"
;;; Loading binary file "set-ebmt-global.sbin"
;;; Loading binary file "distance.sbin"
;;; Loading binary file "weight.sbin"
;;; Loading binary file "main.sbin"
#P"/usr/EBMT/exe/set-env.sbin"
```

以上で、プログラムのロードが終了しました。次章では、シソーラスと用例データベースのロードについて説明します。

4．2　シソーラス・用例データベースのロード

4．2．1　ロードに必要な環境の確認

このパッケージをインストールしただけの状態では、そのままシソーラスと用例データベースのロードを実行することができます。この場合、パッケージに含まれている小規模なシソーラスと用例データベースがロードされます。ユーザーが独自に構築したシソーラスと用例データベースを使いたい場合には、「3．シソーラスと用例データベース」で説明したシソーラス・用例データベースのファイルやフォーマットが正しく指定されているかどうか確認してください。具体的には、以下のファイルの記述が、シソーラスと用例データベースの実体と対応していることを確認する必要があります。

- ~EBMT/def/ebmt-dic-file.def ：　シソーラスのファイル指定
- ~EBMT/def/ebmt-sem-struct.def ：　シソーラスの意味コードのフォーマット指定
- ~EBMT/def/ebmt-exam-file.def ：　用例データベースのファイル指定
- ~EBMT/def/ebmt-exam-attr-type.def ：　用例の要素のタイプ指定

4．2．2　ロード

指定に間違いのないことを確認したら、

```
> (init)
```

とします。すると、次ページのようなメッセージが表示され、シソーラスと用例データベースがロードされます。

```
> (init)

Started initializing...

Started loading dictionary: ../dic/ebmt-e.dic.
100  <----  ロード中のシソーラスのエントリー数（１００件ごとに表示）
Finished loading dictionary: ../dic/ebmt-e.dic. cnt = 185  <----  シソーラスの総数

Started loading example: ../example/at-n.exam

Started calculating weight.

Finished loading example: ../example/at-n.exam. cnt = 55  <----  用例の総数

Started loading example: ../example/at-v.exam
100 200  <----  ロード中の用例数（１００件ごとに表示）
Started calculating weight.

Finished loading example: ../example/at-v.exam. cnt = 231  <----  用例の総数

Finished initializing.

NIL
>
```

４．２．３　ロードの所要時間

以下に、SparcStation2、メイン・メモリ32Mバイトの環境で計測した用例データベースのロードの所要時間を示します。各自の環境をロードする際の目安にしてください。

用例数	時間（秒）
500	5
1000	8
1500	10
2000	14
2500	18

4．3　検索関数の実行方法

言語表現間意味距離計算プログラムの検索システムは、翻訳用と係り受けの曖昧性解消用の２種類の検索関数の形で使用します。両者の基本原理は同じですが、各要素に対する重みづけが異なり、また、返値のフォーマットも多少異なります。この章では、これらの関数について説明します。

4．3．1　翻訳用検索関数

翻訳用検索関数は、次のように実行します。

```
>(eb-transfer　用例タイプ　用例)
```

用例タイプは、「名詞　at　名詞」という形の名詞句を集めた用例データベースならat-n、「動詞　in　名詞」という形の動詞句を集めた用例データベースならin-vというように、用例データベースごとにユーザーが与える任意の名称のことです。言語表現間意味距離計算プログラムでは、検索の際、同じ用例タイプの用例データベースだけを検索するようになっているため、用例タイプを渡す必要があります。

用例は、引数用例タイプで指定した用例タイプの用例と同じフォーマットでeb-transferに渡してください。用例タイプや用例のフォーマットに誤りのある場合は、エラーメッセージが出力され、NILが返ります。正常に実行された場合、結果は次のような形で返ります。

```
#S(EBMT-INFO
  VALUE  ((距離1 (翻訳パターン1　件数 ((用例1) . 件数)((用例2) . 件数)・・・)
                 (翻訳パターン2　件数 ((用例1) . 件数)((用例2) . 件数)・・・)
                  ・・・・・・・・・・・・・・・・
                  (翻訳パターンn　件数 ((用例1) . 件数)((用例2) . 件数)・・・))
          (距離2 (翻訳パターン1　件数 ((用例1) . 件数)((用例2) . 件数)・・・)
                 (翻訳パターン2　件数 ((用例1) . 件数)((用例2) . 件数)・・・)
                  ・・・・・・・・・・・・・・・・
                  (翻訳パターンn　件数 ((用例1) . 件数)((用例2) . 件数)・・・))
                 ・・・・・・・・・・・・・・・・・・・・・・・・
          (距離n (翻訳パターン1　件数 ((用例1) . 件数)((用例2) . 件数)・・・)
                 (翻訳パターン2　件数 ((用例1) . 件数)((用例2) . 件数)・・・)
                 ・・・・・・・・・・・・・・・・
                 (翻訳パターンn　件数 ((用例1) . 件数)((用例2) . 件数)・・・)))
  STEP　ステップ名)
```

（注）

○距離1 < 距離2 < ・・・・・< 距離n

○翻訳パターン1の件数 >= 翻訳パターン2の件数 >= ・・・・>= 翻訳パターンnの件数

○用例1の件数 >= 用例2の件数 >= ・・・・・>= 用例nの件数

○ステップ名は、検索のどの段階で結果が返されたかを示す情報です（「1．2　距離計算の方法」参照）。各名称の意味は次のとおりです。

・STRING-MATCH	字面マッチのハッシング
・SEM-CODE-MATCH	意味コードマッチのハッシング
・PARTIAL-MATCH	部分マッチの全件検索

次に、実際の実行例を示します。なお、以下の実行例では、結果が見やすいようpprintを使って出力します。

```
> (pprint (eb-transfer 'at-n '("conference" "at" "kyoto")))

#S(EBMT-INFO
  VALUE ((0.5 ((N2 での N1) 1 (("speech" "at" "conference") . 1)))
      (1.0892556509887897
       ((N2 の N1) 4 (("we" "at" "foundation") . 1)
        (("someone" "at" "hotel new otani") . 1)
        (("someone" "at" "conference") . 1)
        (("someone" "at" "office") . 1))))
  STEP PARTIAL-MATCH)
>
```

```
> (pprint (eb-transfer 'at-v '("hold" "at" "kyoto")))

#S(EBMT-INFO
  VALUE ((0.5
      ((N2 で) 5 (("hold" "at" "hotel new otani") . 2)
       (("take" "at" "immigration office") . 1)
       (("take" "at" "station") . 1) (("hold" "at" "restaurant") . 1))))
  STEP PARTIAL-MATCH)
>
```

４．３．２　係り受けの曖昧性解消用検索関数

係り受けの曖昧性解消用検索関数は、次のように実行します。

```
> (eb-disamb　用例タイプ　用例)
```

用例タイプと用例については、eb-transferと全く同じです。用例タイプや用例のフォーマットに誤りのある場合は、やはりエラーメッセージが出力され、NILが返ります。正常に実行された場合、結果は次のような形で返ります。翻訳用検索関数と違い、翻訳パターンには意味がないので、翻訳パターンの情報は含まれていません。

```
#S(EBMT-INFO
  VALUE  ((距離1 ((用例1) . 件数)((用例2) . 件数)・・・)
          (距離2 ((用例1) . 件数)((用例2) . 件数)・・・)
           ・・・・・・・・・・・・・・・・・・・・・・・・
          (距離n ((用例1) . 件数)((用例2) . 件数)・・・))
  STEP  ステップ名)
```

（注）
○距離1 < 距離2 < ・・・・・・< 距離n
○用例1の件数 >= 用例2の件数 >= ・・・・・・>= 用例nの件数

次に、実際の実行例を示します。

```
> (pprint  (eb-disamb  'at-v  '("present"  "at"  "conference")))

#S(EBMT-INFO  VALUE  ((0  (("present"  "at"  "conference") . 1)))
        STEP  STRING-MATCH)
>
```

```
> (pprint  (eb-disamb  'at-n  '("paper"  "at"  "conference")))

#S(EBMT-INFO
  VALUE  ((0.5  (("speaker"  "at"  "conference") . 3)
          (("workshop"  "at"  "conference") . 1)
          (("speech"  "at  "conference") . 1)))
  STEP  PARTIAL-MATCH)
>
```

４．４　パラメータの設定

言語表現間意味距離計算プログラムには、ユーザーが設定可能なパラメータがいくつかあります。ここでは、これらの設定方法について説明します。

４．４．１　未知語の表示

未知語の検出は、用例データベースのロード時および検索関数の実行時に行われます。ただし、用例データベースのロード時には必ず表示されますが、検索関数の実行時には、次の関数でパラメータ設定しないかぎり表示されません。これを表示するには、

```
> (show-unknown-word　t)
t
>
```

とします。
また、これを未表示の状態に戻すには、

```
> (show-unknown-word　nil)
nil
>
```

とします。
また、現在の状態を表示するには、引数なしで、

```
> (show-unknown-word)
nil
>
```

とします。

４．４．２　類似用例の検索数

eb-transfer、eb-disambともデフォルトでは、距離の近いものから５番目までの用例の情報を返します（ただし、字面マッチまたは意味コードマッチの用例が存在する場合は、該当するものすべてに関する情報を返します）。

この数を変更したい場合は、

```
> (max-to-retrieve　検索数)
```

とします。
（例）

```
> (max-to-retrieve　10)
10
>
```

また、現在の検索数を表示するには、引数なしで、

```
> (max-to-retrieve)
5
>
```

とします。