考察：「文解析アラカルト」

ここ３ヶ月、日本語文の文法解析と意味解析プログラムを作ってきました。とは言っても、文法は解析らしいことしてますが、意味解析の方は表層フレーム構造を内部意味構造に変換しているだけで、文脈処理、曖昧性処理はまだ先のことです。構想はできているのですが、なかなか時間が無くて困ってます。気が多いのかな、あぶはちとらずに陥らないように気を着けたいところです。

今回は、プログラムを作ってきて見つけた手法を幾つかご紹介したいと思います。
（１）曖昧性への対応について
（２）辞書の記述規則について
（３）係り受け解析アルゴリズムについて

第１章　曖昧性への対応

曖昧性に対応するには、データ駆動型の制御を採用することとデータ構造を工夫していくことが大切です。ハードコーディングでは対応しきれません。

曖昧性には、
（１）係り受け関係の多義性
（２）単語の品詞、文法属性の多義性
（３）意味の多義性、曖昧性
（４）文章構造の多義性、曖昧性
など多数あります。

これに対応するには、RDB を想定して、次のような工夫が必要です。
（１）目的に応じて、テーブルを起こす。無数のテーブルができます。
（２）多義性は同じキーを持つエントリとしてデータを記述していく。
（３）辞書から得るデータクラスも曖昧性管理クラスの配下にコレクションとして保持して、体系だって管理できるようにする。
（４）パターンマッチングにより適用するエントリを決定し、エントリにコマンドを記述することにより、データ駆動型を実現する。

１．１　テーブル

未夢プロジェクトは現在次の１３個のテーブルを持っています。
（１）dict01　　分かち書き辞書

例

stext（キー）	sno	smax（最長）	ttext (品詞項目)
好きだ	1	6	<VB>好きだ</VB>

最長一致法を用いて分かち書きを行っています。

（２）dict02　　分かち書き結果の絞り込み（共起関係辞書）

例

stext (キー)	own_condition （自身条件)	pre_condition (直前共起条件)	post_condition (直後共起条件)	pre_value	post_value
好きだ	ＶＢ (*:+fevolite)	*	VB_[##,XX(と)]	*	*

未夢では曖昧性は全数洗い出して処理していくという方針をとっています。デフォルトを設定して、必要に応じてバックトラックを掛けて、真実に迫るという方針はとっていません。デフォルトの決定には学習システムとの関わりがあるし、バックトラックを何処までするかという大きな問題があります。意味処理に入ってから、分かち書きに戻らねばならない事態もありうるわけです。でも、それは人間的な手法ですから、可能性は探りたくあります。今後の研究課題です。

(３)dict03_modify　係り先定義辞書

例

stext (キー)	pattern （前後共起関係）	target （修飾先）
ND(*:+time)	ND(this)	*_VB(*)

時間の名詞は動詞を修飾することを示しています。

（４）dict03_modified　修飾される条件の辞書

動詞では「を格」とったり「へ格」を取ったりと種類によって違います。これを制御する辞書です。

（５）dict03_cutter　係り受け曖昧性除去の辞書

「～する時～だ」とか、「～だから～だ」とかの文ですと、「時」や「だから」を越えて前の単語が後ろの単語を修飾しません。そんなパスを持つ係り受け結果を刈り込むための辞書です。そんな文パターンを辞書に記述しています。

（６）dict04　　単語の意味を記述した辞書です。文法項目で、文解析してきましたが、それに意味項目を追加します。

（７）dict04_serface2deepframe　単語列を深層フレームに変換するための辞書です。

例

stext （キー）	condition （係り関係）	action （処理コマンド）
VB(走る)	{ND(*)_XX(は), ND(*)_XX(が), ND(*)_XX(を)}	{agent=??(1), subagent=??(3), location=??(5)}

は格の名詞の意味をフレームの agent に設定、が格の名詞の意味をフレームの subagent に設定、を格の名詞の意味をフレームの location 格に設定することを指示しています。

（８）dict05_key　曖昧性管理の「鍵」行動情報辞書

例

stext （キー）	condition （条件）	action (処理コマンド)
VB(好きだ)	*	{actor:object.fevolite=100, actor:object.know=100}

「好きだ」と言っている本人（actor）の対象（object 格）の好意度を１００とし、どうように認知度を１００とせよ・・・という処理を指示しています。

（９）dict05_keyhole　曖昧性の「鍵穴」の待ち情報辞書

例

stext (待ちキー)	wait_condition (待ち条件)
f001	[actor:object.fevorite>50]

曖昧性のフレームの選択条件としてタグ「f001」を設定します。タグは actor の object の fevorite 値が５０以上の時に待ち状態が解けることを示しています。

（１０）dict06_coexist　共起条件辞書

例

stext（キー）	co_item_set（プライミングするセット）
AV(速く)	VB(*:+move)

副詞「速く」が現れた時には動詞の意味項目に「move」のあるもの全てをプライミングすることを指示しています。

（１１）dict06_knowledge　状況知識辞書（検討中のもの）

例

stext (キー)	frame_set （フレーム セット指定）	frame （フレーム定義）
ND(レストラン)	f_restrant	*
f_restrant	*	{ND(テーブル）,ND(椅子),ND(食べ物),ND(食器）}

レストランには、テーブル、椅子、食べ物、食器があることを示します。

（１２）dict06_priming　プライミング制御辞書

例

stext (キー)	co_condition （プライム セット指定）	sentence_item_set （プライムセット内の文書要素指定）
VB(走る)	movement, mcomputer	*
movement	*	VB(歩く:+move),VB(投げる:+make.to.move), VB(泳ぐ:+move+in.water)
computer	*	VB(動く:+execute),VB(異常終了する:+execute)

単語を沢山収集すれば、co_condition のアンド条件で、分野が決定し、より適切なぴらいみんぐができるようになっていきます。その制御テーブルです。

（１３）semantics_pattern01　表層文の整形用辞書

例

stext	condittion_pattern	mean_frame
rewrite	ND(*)_VB(する)	VB(#1#)

「実行する」などのサ変動詞を形成します。名詞を動詞化しています。

１．２　曖昧管理クラス

一つのキーに対して辞書のエントリを複数設けることで、曖昧性（多義性）に対応するように、コア内のデータエリアもそうした機構を設けて曖昧性に対応しています。

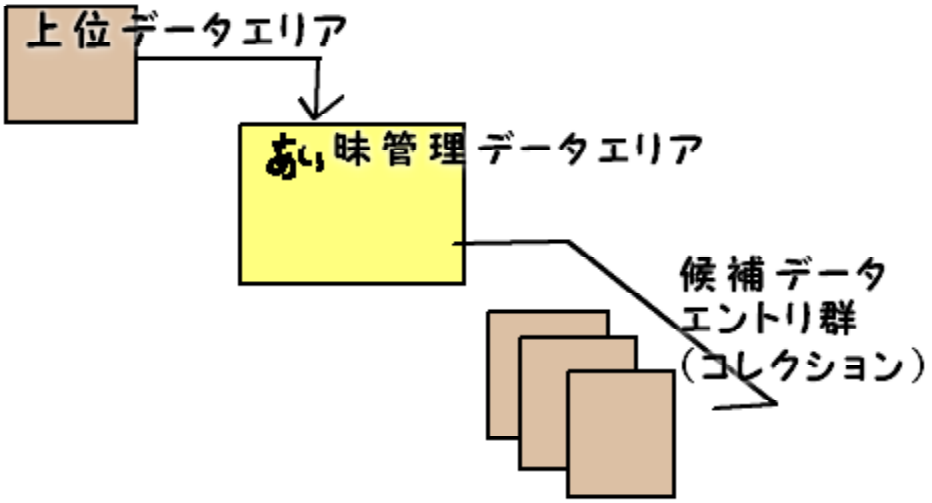

図１　曖昧管理

データエリアは基本的に一つの抽象クラスからつくります。そうすると、いたるレベルの曖昧性も一つの機構で表現できて便利です。将来はデフォルトを指定できるようにしたいと考えています。

第２章　辞書の記述規則

今までは XML で記述してきたのですが（名残が分かち書き辞書に残っている)、今回から自然言語向きのもっと簡潔な記法を採用しました。次のように単語の固まりを強く意識したしたものにしたのです。

・　（）で、単語を表現する。 ・　_で単語のシーケンスを示す。 ・ {}で羅列を表す（アンド条件)。 ・　[]で選択を表す
例)　「私はかもめ」は、ND(私:+pronoun)_XX(は:+case)_ND(かもめ:+bird) と表現されます。　ND,XX は品詞、+pronoun,+case,+bird は文法属性や意味属性です。

図２　辞書の記述方法

これで大分処理が軽くなりました。見通しも良いです。

第３章　係り受け解析アルゴリズム

単語間の係り受けは交差しないという原則があります。

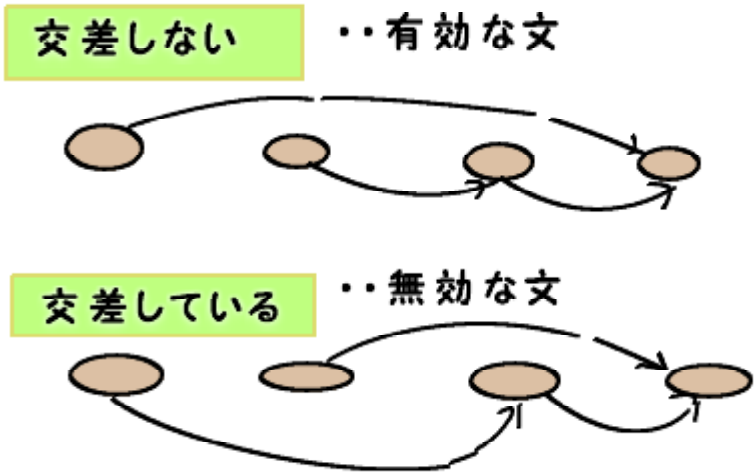

図３　係り受けの原則

前に作った文解析プログラムでは、文解析の係り受け解析で係り受けが解決できない状況になるとバックトラックして再処理してました。そのとき、この交差無しの原則に従う

ために、バックトラックと供に係り受けパスの再張り直しという作業をしたのでした。

今回はもう、曖昧性の全件を数え上げることにしましたので、別のアルゴリズムを採用しました。リカーシブルコールで、係り先を縮小しながら全件数え上げていくのです。これは結構エレガントなアルゴリズムになりました。この交差無しの原則を使うのです。

アルゴリズムは、最外パスを決め、残りをその内部へのパスになるからという風に解析していくものです。

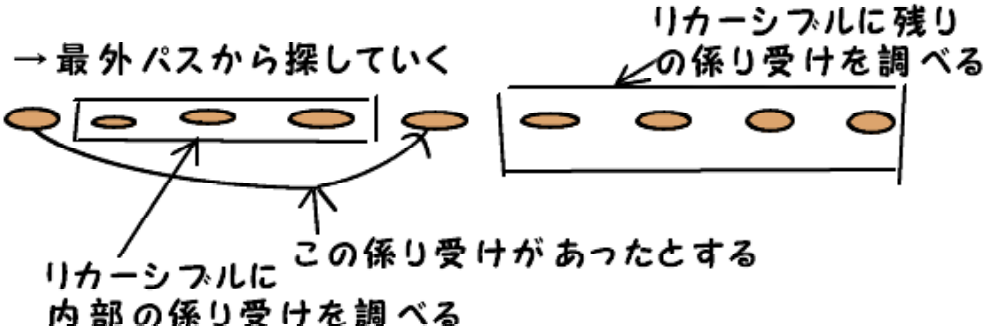

図４　係り受けアルゴリズム

第４章　おわりに

曖昧性を全件処理していくというのと、データ駆動型のプログラム制御を採用したことで、大分プログラミングが簡単になった・・・というより、自然言語が解析できるようになったと思います。未夢プロジェクトの前回挫折地点を軽くクリアできました。

あとは、文脈処理、知識ベースの実現、「文脈空間」実現へと挑戦していくつもりです。今のところは何とかなりそうとの見通しを持てています。

暫くは、今できたプログラムのデバックを兼ねて日本語文のデータを収集していくつもりです。春にはなんか結果出したいなと・・・・・・。

おわり