修士論文　　2005（平成 17）年度

論文題目

英文の依存構造に基づく
意味タグ自動付与システムの開発

慶應義塾大学大学院理工学研究科
開放環境科学専攻

学籍番号　80420028　　深津 康行

指導教員　教 授　　櫻井 彰人

慶應義塾大学大学院理工学研究科
2006 年 3 月

英文の依存構造に基づく
意味タグ自動付与システムの開発

※題目は Web 配布の題目用紙（Microsoft Word 形式）を使用

慶應義塾大学大学院 理工学研究科

開放環境科学専攻

学籍番号 80420028

深津康行

2005 年度

目次

図目次

表目次

1 序論

1.1 背景

コンピュータが，ノートやペンと変わらない知的作業の「道具」となって久しい．これほどまでにコンピュータが一般に普及することになった背景には，コンピュータが，人にとって最もなじみ深い「言語」によって記述されたデータを効率的に処理できるようになったことがある．また，この文書データのコンピュータネットワーク上での運用方法が普及を後押しした．互いに関連する複数の文書データを関連付けるハイパーテキストの仕組みは，広く一般の人々に文書の新しい取り扱い方を提示した．また，誰もが比較的容易に文書を作り他の文書と関連付けることができたことからインターネット上のデータ量は爆発的に増加し，現在の巨大なワールドワイドウェブができた．現在ではこれらのデータを取り扱うハードウェア環境が極限まで普及し，ますます多くの情報が電子化して生成され，蓄積され，利用される状況ができあがっている．

ところが，ネットワーク上のデータ資源が増加し続けた結果，その中から真に必要な情報を選択することが困難になるという状況が生まれてしまった．インターネット上のデータ資源は膨大であり，また誰もが情報を発信できることから信憑性などの質の不均一が生じているためである．これらの問題を解決するための仕組みとしては，Yahoo!のように各種のサイトを整理し見出しを付けて提示するディレクトリサービスや，Googleに代表されるような検索システムなどが作られている．データ資源をより効率的に利用できるようにするためのこれらの仕組みはどちらも，そのままでは無秩序で利用しづらいデータ資源を何らかの形で構造化し，利用しやすい形で人間に提示しようというものである．この場合，前者は辞書のように既に人が一定の基準をもってデータを構造化していることで利用者は望む情報を探しやすくなり，また後者は「多くの人が有用と思うものが有用なものである」という基準をもとにキーワードに対応するデータを並べることで利用者に選択をさせやすくしている．一定の基準で構造化されているデータは統一的に処理することが容易であり，機械による自動処理にも適している．そこで，データを何らかの基準に基づいて構造化し，より効率的に処理することが求められるようになった．例えば近年非常に活発に研究が行われているセマンティックウェブは，データの内容を明示する**メタデータ**の記述方法を標準化し，あらかじめデータを構造化して生成し普及させることで計算機にデータの「意味」を処理させることを基本姿勢としている．

だが一方，あらかじめ構造化して作成されることが期待できないデータも存在する．本来，文書データは人間にとって自然な言語によって記述されているものが多数であり，これらの文書データすべてをあらかじめ人間が計算機可読な形に構造化して作成するということは期待できない.[*1]また，現在のコンピュータネットワーク上では，既に大量のコンテンツが構造化されていない冗長な文書データとして作成されてしまっている．絶えず冗長な形で生成され続けているこれらの文書データを資源として有効にかつ効率的に利用するためには，これら構造化されていない自然言語記述のデータを計算機で統一処理可能な形に構造化する技術が必要となる．近年，この問題に対して自然言語処理と呼ばれる技術が用いられるようになってきている．

1.2 自然言語処理

自然言語処理（Natural Language Processing：NLP）とは，人間が普段使用している**自然言語**を用いて記

[*1] 橋田らの提唱するセマンティックオーサリングでは，人間の論理的思考過程に沿って文とその間の意味的関係を記述し計算機可読性の高い文書データを生成することが提案されている．

述された情報を，計算機によって処理する技術の総称である．自然言語は非常に複雑かつ冗長な構造を持ち，無限の表現能力を持っており，また語の用法ばかりか文法までもが時とともに変化する性質を持っている．このため，単純な文字列として記述されたテキストデータから言語が本来表現している情報を計算機によって取り出し，扱うことは非常に困難なタスクとなる．自然言語を計算機で扱うためにはそのモデルを構築する必要があるが，自然言語の真のモデルを構築することは不可能であるため，さまざまな手がかりから自然言語の近似的なモデルを作成することで情報を部分的に取り出し，利用する試みが続けられている．

言語の解析は，その方法で大きく二つに分けられる．(1) 制約に基づく文法理論によるものと (2) 統計に基づく機械学習によるものである．前者は詳細な文法的制約を直接ルールとして記述し，これを用いて文の解析を行うというものである．後者は計算機上の進歩によって発達した方法論であり，実際に人によって解析された大量の文データから言語モデルを学習・獲得させるものである．統計（機械学習）に基づく言語処理は，単純に規則を列挙するだけでは解析できない未知の言語現象に対応する頑強性の獲得のために有効な手段であり，計算言語学（Computational Linguistics）と呼ばれる学問分野からの知見を取り入れ，近年発達している．本論文では，この統計に基づく自然言語処理技術を中心に論じることとする．

「言語を解析する」というタスクは，より定義の厳密で工学的に実現可能なサブタスクに分解される．その最も基本的なレベルのタスクとして**形態素解析（**morphological analysis**）**が挙げられる．**形態素**とは文を構成する最小の単位であり，多くの場合単語がこれに該当する．文中の単語を特定し，**品詞（**part-of-speech : PoS**）**や語尾変化など形態素が単体で持つ性質を決定することが中心となる．英語のように単語を分かち書きする言語においては早くから品詞タグ付け（PoS Tagging）が中心的課題となり，現在では 96〜98 %程度の精度で語の品詞を推定することが可能となっている．一方，中国語や日本語のように単語間に区切りを入れる習慣のない言語においては，適切な形態素の区切りを発見することも重要な課題となっている．

次に，ひとつの文の中でその文法的構造を解析対象とするのが**構文解析（**syntactic analysis, parsing**）**である．文の構造を記述する文法は，言語学の分野でさまざまな形態が提案されているが，英語の自然言語処理では**句構造（**phrase structure**）**に基づく**構成文法（**constituency grammar**）**が主流となり，解析結果として**構成素木（**constituency tree**）**と呼ばれる木構造を出力するタスクとして発達している．一方，日本語では語順が比較的自由であることから，単語や文節間の**依存関係（係り受け関係）**に基づく**依存文法（**dependency grammar**）**がよく用いられている（図 1）．この構文解析についても，データからの機械学習に基づくモデルの生成，自動解析技術が活発に研究され，形態素解析には及ばないもののかなりの精度での自動解析が可能になってきている．

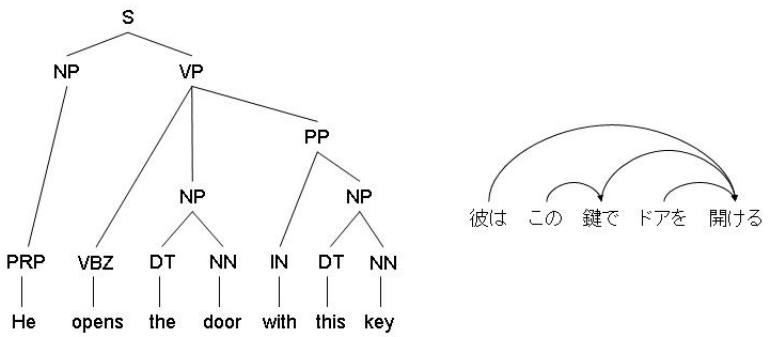


図 1　構成素木（左）と依存構造（右）

最後に，自然言語で記述された情報の処理に直接的に取り組もうとするのが**意味解析**（semantic analysis）である．これは形態素や文法といった統語構造ではなく，文の持つ意味構造を解析の対象とするものであるが，「意味」の定義は一意ではなく，上記二種類の解析と比べて問題の定式化が若干曖昧なままである．文の意味する内容は文脈（context）によって異なり，一文中では特定できないことが多いため，実用的には，キーワードや文書ベクトルによる大域的な特徴量を計算するようなドキュメントレベルの意味の解析が中心となっている．しかし，よりプリミティブな「意味」の処理として，言語学において「言語表現の意味」と定義されている一文単位での意味を解析するタスクが検討され，近年研究が活発になっている．欧米では “Semantic Parsing” や “Semantic Role Labeling : SRL” と呼ばれるようになったこのタスクは，主に Fillmore の提唱する格文法理論 [35] をベースとして意味解析問題を定義している．この格文法理論に基づく自然言語処理研究では，文の中心となる**述語**（predicate）と呼ばれる語に対してその他の語が持っている**意味役割**[*2]（動作主，経験者，対象物，始点，終点，道具，時間，場所など）を特定することで文の意味の解析としている．一般的にには，文の中心となる動詞が述語となり，その周辺に主に名詞や名詞句などが何らかの格を持って存在している（図 2）文がデータとして作成され，研究に用いられている．直感的には，これらの意味役割を特定することで，「誰が（Who）いつ（When）どこで（Where）誰に（To Whom）何を（What）なぜ（Why）どのくらい（How）したか」を解析するというものである．一文を単位とするファーストステップ的な意味の解析であることから，Shallow Semantic Parsing などと呼ばれることもある．同様のタスクは，日本では一般的には深層格推定問題と呼ばれている．こうした意味解析の試みは 1990 年代後半頃から実際にデータを用

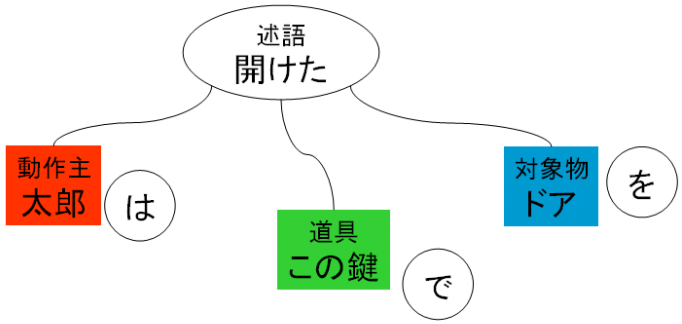


図 2　文の「意味」（述語と意味役割）

いて行われるようになった．

こうした意味役割のタグ付けは，質問応答や機械翻訳などの応用的自然言語処理のための基礎技術として注目されており，実際に，これらの意味的な役割がタグとしてドキュメント中の自然言語文に付与されていることで要約の自動生成や類似ドキュメントの検索など応用的な文書処理が効率的に行えるということもいくつかの研究で示されてきている．[26][24]

文の部分的な意味情報を抽出する取り組みとしては，この他にも「人名」「地名」「組織名」などの固有名詞的な語を特定する「固有表現抽出」や文と文との意味的な関係を解析する「修辞構造解析」などが行われているが，上で紹介した「意味役割タグ付け」のタスクは言語学的な研究成果をもとに文の基礎的な水準の「意味」を定義して解析しようというものであり，これまで中心的に行われてきた統語的な解析の次ステップのタスク

[*2] 格文法理論では**深層格**と呼ばれ，実際に語として出現する**表層格**の背後に存在する語の意味的な関係性とされている．Fillmore は，深層格は言語に共通の一般的な概念であり語の表層的な実現形態とは必ずしも一意に対応しないことを主張している．

として代表的なものとなっている．本論文では，将来的に文書の高次の意味処理を実現するための基礎技術として，この「文単位での意味解析」に注目して論じることとする．

1.3　意味解析研究の概要と現状の課題

前節で述べた文の意味解析の研究では，データに基づく統計による自然言語処理の方法論が中心となっている．これは，新聞記事や雑誌，辞書などの文データを集めて人手で解析したデータセット（**コーパス：**corpus）を意味解析の模範解答とし，このデータセットの一部を使って言語解析モデルを作成（統計学習）するというものである．この場合，学習によって獲得された言語解析モデルの解析能力は，学習に使ったものとは別のテスト用データセットを解析させることで評価される（図 3）．

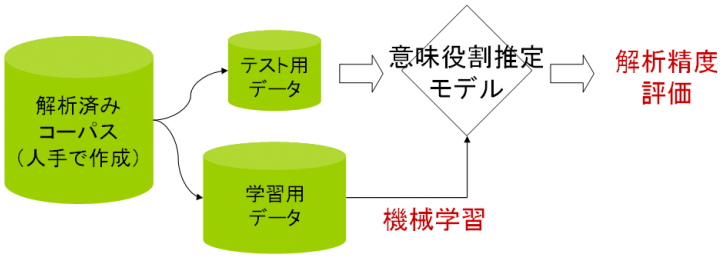


図 3　コーパスからの言語解析モデルの学習・評価

コーパスの作成においては，大量の文データについてその統語構造や意味構造を人がひとつひとつ記述していかなければならない．そのため，コーパスの作成は大学や企業の研究者などを中心とする一大プロジェクトとして運営され，大量の人手を動員して行われる．しかし一方で，コーパスの作成者は対象言語の構造についてある程度の知識を持っていなければならず，またコーパス全体で記述方法に一貫性が保たれていなければならないため，実際に巨大なコーパスを作成する際には作成者に多大な負担がかかることになる．結果として，質・量ともに優れたコーパスはなかなか作成できないという問題がある．

また，現在の意味解析研究では，機械学習の結果得られる言語解析モデルの客観的な評価が難しいという側面もある．文データの統計的学習では，一般的にテストセットでの解析精度をパーセンテージで評価することが多い．そのため客観的な評価のためには，同様の学習セットでモデルを学習し，同様のテストセットで評価しなければならない．欧米では 2004 年及び 2005 年に，研究者間でデータセットを共有して意味解析モデルの性能を競い合う枠組み（後述）が作られたが，日本ではまだこういった試みは十分な形で行われていない．さらに，コーパスのテストセットによる評価は，あくまで同質のコーパス内での閉じられたデータ中での評価であり，実際に解析モデルを適用するフィールドで同様の性能が維持できるかどうかは，明らかではない．

こういったデータ作成やシステム評価の困難さがありながらも，意味解析研究は徐々に進歩し，一定の成果を示すようになってきている．しかし，大量の文書データを実際に計算機で自動的に意味解析し，これを何らかのアプリケーションとして利用しようという動きには至っていない．これは現状では意味解析モデルの解析性能がまだ十分でないためだが，研究において考案された意味解析のアルゴリズムを未完成ながらもソフトウェアとして実装し，誰でも試験的に利用できるように整備しようという動きが，統語解析の研究と比べて非

常に少ない．これには，現状の意味解析モデル構築方法に含まれる若干の問題点が影響している．

コーパスから統計学習によって意味解析モデルを作成するにあたっては，品詞タグや構成素木，文の時制や態など，文データにおけるさまざまな特徴量が既知のものとして利用される．しかし，これらの中には実際に生文から自動的に得るのが難しい特徴量もいくつか含まれている．そのため，実際にこの意味解析モデルをまったくの未知文に対して適用するには，これらの特徴量を自動解析によって推定する必要がある．コーパス中で学習・推定を行い精度を評価することはできるが，実際に未知文を処理できるソフトウェアとして実装するのにいくつかの手続きが必要になるため，なかなか研究成果が実利用に広がらないようになってしまっているものと思われる．

このコーパス依存性の問題は，別の面でもあらわれている．そもそも文の意味として定義された文中の意味役割（深層格）は言語普遍性を意識した概念であり，このため意味解析研究には将来的な多言語間処理技術（機械翻訳など）への応用が期待されているが，現段階ではこの意味解析を利用して異言語間のギャップを埋めることを試みる研究は行われていない．この原因には，意味解析の精度が現段階ではまだ不十分であることの他に，作成されるコーパスが言語依存性の高いものになってしまっており意味解析モデルの学習アルゴリズムもこれに依存しやすいこと，共通の基準で異なる言語の文を記述したコーパスが少ないことなどが考えられる．

1.4 本論文の目的

本研究では，自然言語処理による文書データの構造化技術の発展に貢献し，意味タグ付き構造化文書を積極的に作成できる環境を整えることを目的とする．そのための課題として，現在，自然言語処理研究が直面している英語の意味解析タスクを対象とし，前節で紹介した現状と問題点を踏まえ，

- 自動解析に繋げやすい意味解析モデル学習アルゴリズムの考案,
- 日本語等の他言語とも共通性のある意味解析手法の提案,
- ツールとしての意味解析システムの実装

を行うことを目標とする．

1.5 本論文の構成

本論文の第二章以降の構成は以下のとおりである．

第二章では，これまで構築されてきた言語資源（コーパス）と，これを用いた意味解析の既存研究を紹介する．第三章では，第二章で紹介した既存研究等を分析し，これに対して本研究の主題である依存構造に基づいた意味解析手法の提案をおこなう．第四章では，提案する意味解析を実装するために用いた機械学習アルゴリズムである Support Vector Machine の導入をおこなう．第五章では，提案システム 1 として依存構造を意味解析の特徴量に利用する意味解析システムを実装し，精度評価をおこなって結果を考察する．第六章では，提案システム 2 として依存構造解析と意味解析を交互におこなうシステムを実装し，評価をおこなう．第七章で結論と今後の課題を述べる．

2　意味解析の試み

序論で述べたとおり，意味解析の研究では統計的機械学習による方法論が主流となり，コーパスを用いて解析モデルを構築するのが一般的となっている．意味解析研究に用いられるコーパスはあまり多くないが，それぞれ記述の方法や内容に特徴があり，それゆえ解析モデルの学習アルゴリズムにも違いが表れてくる．そこで本章では，自然言語処理で用いられている代表的なコーパスと，それぞれのコーパスを用いた意味解析の既存研究を紹介する．

2.1　言語資源

まず，自然言語処理研究に用いられている代表的なコーパスを紹介する．はじめに英語の自然言語処理研究で非常に重要な役割を占めている Penn Treebank を，次に英語の意味解析研究で主に用いられている FrameNet と PropBank を紹介する．最後に，日本語の自然言語処理研究で広く利用されている EDR 電子化辞書を紹介する．

2.1.1　Penn Treebank

Penn Treebank[*3] は米ペンシルバニア大学のプロジェクトにおいて作成された，英語の統語的特徴を解析し記述したコーパスである．Wall Street Journal，The Brown Corpus 等四種類のテキスト資源に対して，単語ごとの品詞タグ，名詞句ブラケット（括弧付け），構成素木の三段階の構文的特徴が記述されている．品詞タグと名詞句ブラケットは図 4 のような簡単な構造で記述されるが，木構造である構成素木は図 5 のようなリスト構造で記述されている．

```
[ Rockwell/NNP  International/NNP  Corp./NNP ]
won/VBD
[ a/DT  $/$  130.7/CD  million/CD  Air/NNP  Force/NNP  contract/NN ]
for/IN
[ AC-130U/NN  gunship/NN  replacement/NN  aircraft/NN ]
./.
```

図 4　Penn Treebank データ構造：品詞タグと名詞句ブラケット

Penn Treebank は英文の統語的特徴を体系的に大量にデータ化した初めての試みのひとつであり，現在においても自然言語処理における統語的解析の研究のための重要な資源となっている．現在，英語の構文的解析において高い精度を示している構文解析器の多くは Penn Treebank の構成素木を用いて学習を行っている．この Penn Treebank 以外にも，言語の構文的特徴，特に構文解析木を記述した各種の Treebank がさまざまが言語で作成され，統語的解析システムの構築に利用されている．ただし，Treebank は文の文法的構造を記

[*3] http://www.cis.upenn.edu/ treebank/

```
( (S
    (NP-SBJ (NNP Rockwell) (NNP International) (NNP Corp.) )
    (VP (VBD won)
        (NP
          (NP (DT a)
              (ADJP (QP ($ $) (CD 130.7) (CD million) )
                    (-NONE- *U*) )
              (NNP Air) (NNP Force) (NN contract) )
          (PP (IN for)
              (NP (NN AC-130U) (NN gunship) (NN replacement) (NN aircraft) ))))
  (. .) ))
```

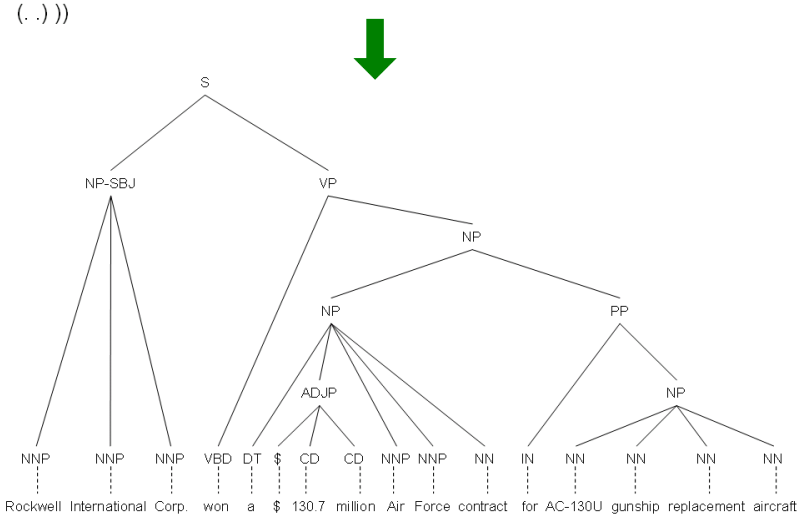


図 5　Penn Treebank データ構造：リストによる構成素木の記述

述したコーパスであり，本論文でテーマとしているような意味的構造に関する記述は含まれていない[*4]．

2.1.2　FrameNet

FrameNet[*5] とは，カリフォルニア大学バークレー校の Fillmore らが中心となり，フレーム意味論と呼ばれる意味構造理論に基づいて文の意味を分析・記述し，オンライン語意情報資源として普及させることを目的としたプロジェクトである．ここで作成されている意味解析済みコーパス自体もこの名前で呼ばれている．

フレーム意味論では「話者がある語の意味を理解するにはその語の**認知的な背景知識の枠組み（フレーム）**を考慮する必要がある」と考え，そのため FrameNet ではネイティブスピーカーの知識を語彙項目ごとに詳細に記述するという姿勢がとられている．Fillmore の格文法理論が元になっているため，基本的な意味構造としては序論で述べたような「述語となる語（FrameNet では “Target” と呼ばれる）とその周辺の意味役割を持つ語または句（“Frame Element” と呼ばれる）」という形が採用されている．ただし，文はフレームと呼ばれる背景知識ごとに整理され，フレームに固有の意味役割名を付与されている．（図 6）

図 6 は動詞が述語となる典型的なフレームだが，実際に作成されている FrameNet コーパスでは，名詞が

[*4] subject, object などの簡単な表層的機能は，構成素木のノードの性質として記述されている．
[*5] http://framenet.icsi.berkeley.edu/

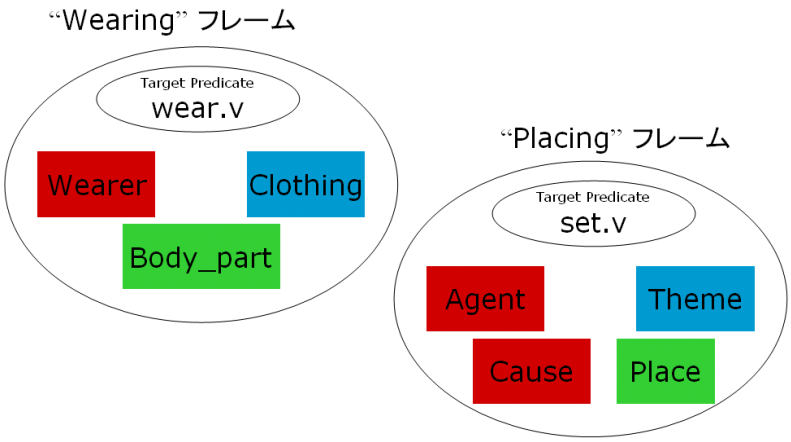


図 6　FrameNet：フレームごとに異なる意味役割：
（ Wearer, Agent, Cause は動作主体的，Clothing, Theme は対象物的，Body_part, Place は場所的役割）

述語となり，その性質を表す形容詞なども Frame Element となるフレームが数多く定義されている（図 7）．

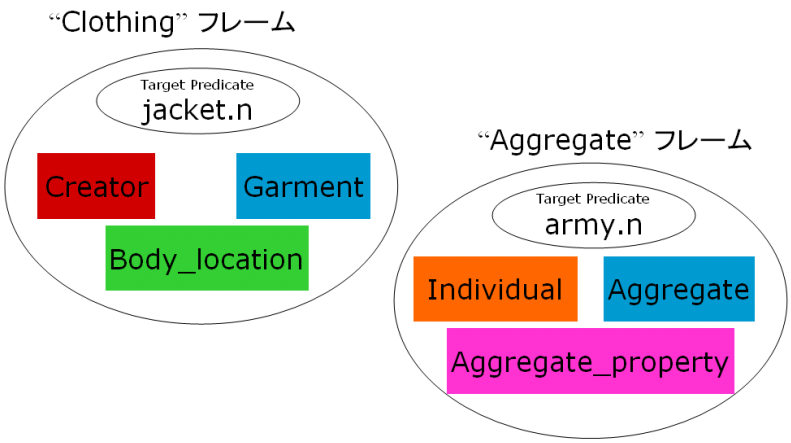


図 7　FrameNet：名詞が述語となるフレーム

当初は小規模なコーパス構築プロジェクトであったが，2005 年の version 2 リリース時点で定義済みフレーム数が 500 強，文例数も 13 万を越すほどになっており，規模の面でも他のコーパスにひけを取らないレベルになってきている．また，日本語やドイツ語，スペイン語など各国でそれぞれ FrameNet に相当するものを構築する動きも広がっており，最終的にはこれら多言語の FrameNet データベースを接続し，異なる言語間で意味構造の理解するための語彙資源となることを目指している．

文の意味的構造を記述した初めてのコーパスであったため，英語の意味解析研究の初期にはこのコーパスを対象とした研究が数多くなされた（後述）．しかし，コーパスとして自然言語処理に利用しようとする立場から見ると，いくつかの問題点も残されている．

構文情報の不足

構文構造から切り離した純粋な意味構造を興味の対象としているため，コーパス中に記述されている構文的情報の記述が少ない（品詞タグと文法的機能，句チャンクのみ）.

データフォーマットの欠点

もともと言語の意味構造研究を目的とする言語学者を中心に始められたプロジェクトであり, FrameNet コーパスの直接の目的は自然言語処理ではなく語彙資源として公開することであるため，自然言語処理研究を想定してデータが記述されているわけではない．その為，データ記述のスタイルが少々汚くなってしまっており（XML を用いて記述されているが，文全体を<text>タグで括り文字インデックス番号で単語を指定するなど構造が複雑），誰もが簡単に扱うことができる形にはなっていない.

フレームの複層性の欠如

本来，フレームはひとつの文の中に複数存在することがありうる．例えば，“Wearing” フレームの文中に登場する “Clothing” という Frame Element には，その内部に入れ子的に “Clothing” フレームという背景知識の枠組みが存在するはずであるが，現在の FrameNet はこれを別々に記述し，データとして結び付けられてはいない.

フレームの網羅性の欠如

文は定義されたフレームごとに分析され，記述されているが，あらゆる状況を網羅する背景知識の枠組みを列挙することは難しく，現状ではコーパス中に多く存在するフレームや，言語学的に特徴のあるフレームを優先して分析が進められている．この状況は人工知能の古典的な問題であるフレーム爆発と同質の問題と思われ，そのため，これを言語資源として意味解析システムを構築したとしても，未知の多様な文書データに対する一般化が難しいのではないかと考えられる[*6] .

2.1.3 PropBank

PropositionBank（以下 PropBank）[*7] は，前出の Penn Treebank に**述語-項構造（**Predicate-Argument structure**）**と呼ばれる意味的構造の記述を追加したコーパスである．述語-項構造とは，文中の動詞が述語となり，その周りにある意味役割を持つ語または句（項と呼ばれる）に対して，Argument と呼ばれる意味役割タグを割り振ったものである.

例：“He wouldn’t accept anything of value from those he was writing about.”

↓

[A0 He] [AM-MOD would] [AM-NEG n’t] [V accept] [A1 anything of value] from [A2 those he was writing about] .

[*6] このため，FrameNet の言語資源としての実用性に疑問を唱え，フレームの階層性の検討と複層的な意味フレームの分析をおこない「使えるコーパス」の構築を目指す試み [3] も現れている.

[*7] http://www.cis.upenn.edu/ mpalmer/project_pages/ACE.htm

Argument は，全ての述語に共通の一般的なタグとして定義されている．述語に対して中心的な役割を示すコアなタグとして Arg0〜Arg5 が定義されており，Arg0 は動作主体的（Agentive）なもの，Arg1 は対象物的（Objective）なもの，などとされている．ただし，これらの対応付けは固定的なものではなく，Argument 何番がどんな意味役割を表すかは述語となる動詞ごとに別に定義されており，この定義ファイルを参照することで詳細な意味役割を特定する．意味役割への番号付けは文中での出現位置などにも関連しており，そのためある意味役割が述語動詞の間で異なる番号に割り振られることがある（図 8）．

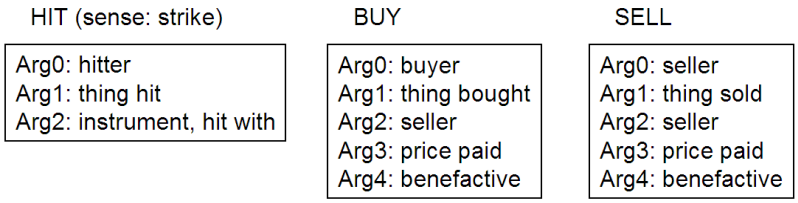


図 8　PropBank：述語ごとに異なる Argument 定義：
BUY における buyer，SELL における seller は，HIT における hitter 同様，共に Agentive（動作主格）である．HIT における instrument（道具格）は Arg2 とされているが，述語によっては Arg3 や Arg5 になることもある．

場所や時間などの付加的な意味役割は ArgM と記述されている．ArgM には表 1 のようなものがある．

表 1　付加的 Argument タグ

TagName	意味する内容
ArgM-LOC	動作などの場所
ArgM-TMP	動作などの時間
ArgM-MNR	動作などの様子
ArgM-DIR	方向（source, goal などを含む）
ArgM-CAU	原因
ArgM-NEG	否定語（not, No など）
ArgM-MOD	助動詞（will, may など）
ArgM-EXT	変化の程度
ArgM-PRP	目的
ArgM-ADV	修飾語一般

複文などで複数の動詞がある場合にも，それぞれを述語としたときの述語-項構造が複層タグ付けされており，この点，FrameNet よりもリッチな記述がなされているといえる．ただし，述語となる語を動詞に限っているため，FrameNet に見られるような名詞を述語とするモノ・コト概念を説明する意味タグは定義されていない．（例：“the discovery of a new continent” のような句について，「“discovery” の対象物（Objective :

Arg1）が “a new continent” である」といった意味タグ付けは行われていない.）

PropBank は FrameNet と違い，自然言語処理の研究者がはじめから中心的に活動して作成されていることから，計算機によって処理することを意識して設計されている．特に，意味役割タグを一般性の高いものに絞り，詳細な意味は述語ごとに定義しなおす形を採用したことで，計算機による推定をおこないやすくなっていることが特徴的である．Treebank を元に作られているため構文的特徴の記述も十分であり，このため英語の意味解析研究ではここ数年，PropBank を利用するのが事実上の標準となっている．

2.1.4　EDR 電子化辞書

これまで紹介してきた欧米発のコーパスに対して，EDR 電子化辞書（以下 EDR）[*8] は日本で作成された数少ない大規模コーパスのひとつである．知的情報処理のソフトインフラ開発を目的とし，通産省主導の下コンピュータメーカー 8 社の共同出資によって，1986 年度〜1994 年度の 9 年間のプロジェクトで作成された．

EDR はその名が示すとおり，コーパスというよりは辞書としての側面が有名である．日本語コーパス（20 万文）と英語コーパス（12 万文）それぞれに対して人手で解析をおこない，そこに含まれるさまざまな情報を「辞書」という形で記述しなおしたものが作成されている．EDR では，文中で語が表している詳細な意味

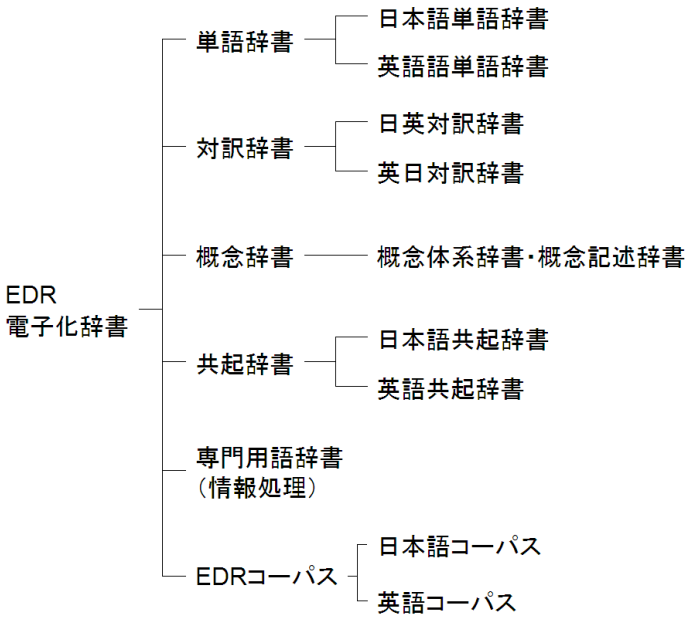


図 9　EDR 電子化辞書の内訳

を「概念」と呼び，これを一意に特定する ID を割り振っている．EDR 辞書はこの概念番号をキーとし，単語の概念や共起関係などを詳細に記述している．

[*8] http://www2.nict.go.jp/kk/e416/EDR/J_index.html

これらの辞書のもととなっている「EDR コーパス」が，欧米で作成されている各種コーパスに対応する部分である．EDR コーパスには，文ごとに

- 形態素
- 形態素の読み仮名（英語ならば基本形（stem））
- 形態素の品詞
- 形態素の概念番号
- 構文情報
- 意味情報

が記述されている．このうち形態素の概念番号は，辞書作成時に人手で定義された語の概念を一意に特定する ID であり，EDR 辞書群の中でのキーとなるものである．また，構文情報と意味情報は，形態素の情報とは別のリスト構造でそれぞれ記述されている．

EDR の構文情報の書式は日本語・英語共通で，形態素同士を合成関係によって結びつける独特な木構造（図 10）で表現されているが，その基礎となっているのは形態素間の依存関係であり，比較的簡単に依存構造木に変換することができる[*9]（図 11）．英語については，前述したように句構造に基づく構成素木が主流であるため，英文の依存構造が記述された数少ないデータ資源となっている．

```
(S (t (M (S (t (M (W 1 "その")
          (t (M (S (t (M (W 2 "原価")
                    (t (W 3 "計算") ) ) ) )
                (W 4 "の") )
               (t (W 5 "方式") ) ) ) ) )
        (W 6 “に”) )
       (t (M (S (t (W 7 "僕") )
            (W 8 "は") )
           (t (M (S (t (W 9 "ケチ") )
                (W 10 "を") )
               (t (S (t (W 11 "つけ") )
                    (W 12 "た") ) ) ) ) ) ) ) )
  (W 13 "。")
)
```

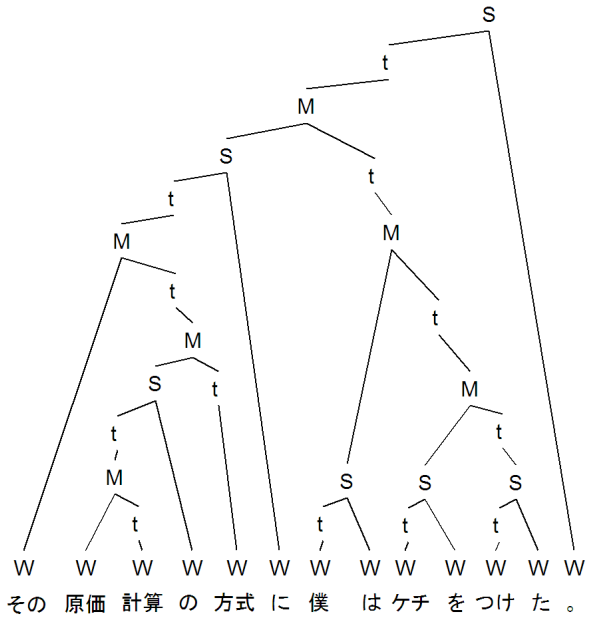


図 10　EDR の構文情報の例（日本語文）
「その原価計算の方式に僕はケチをつけた。」

欧米の意味解析研究における意味役割に相当するものは，「意味情報」の部分に**概念関係子**という形で記述されている（図 12）．これは，「文の依存関係において何番の形態素が何番の形態素に対してどんな意味関係を持

[*9] 変換によって生成できる依存木は，日本語ならば格助詞，英語ならば前置詞等の位置が一般的な依存木と多少異なっている．

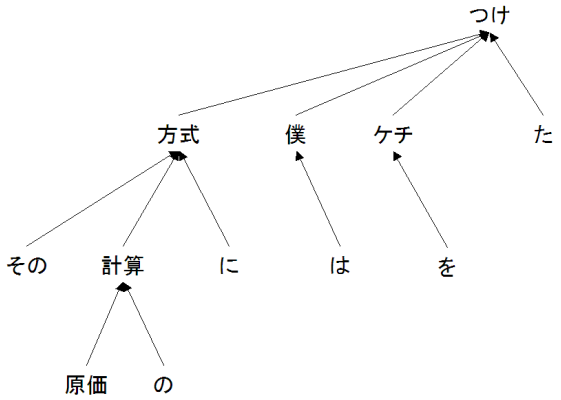


図 11　図 10 から得られる依存構造

つか」を表すデータである．概念関係子は，agent や object など格文法理論における深層格に相当するものが中心になっており，そのため研究者によってはこれを深層格タグなどと呼ぶこともある．しかし，実際には動詞を述語とする意味役割的な深層格だけではなく，名詞に対する「材料（material）」や「所有者（possessor）」，「量（quantity）」などのように，モノ，コト概念の性質を表現する関係子も含まれている．また，これらの概念関係子は文中の特定の「述語」に対してではなく，「ある語が依存関係にある別の語に対して持つ意味」として定義されている．そのため，文中に特別な「述語」というものは定義されておらず，PropBank には見られない，名詞を「述語」とみなす形態のタグ付けも非常に多い[*10]．また，単語ではなく句や文節が意味役割を持つ場合，FrameNet や PropBank では句全体をまとめて意味役割タグが付与されているが，EDR では句を代表する単語にのみタグが付与されている．

```
[ [main [ [main 11:つけ:“=Z  ある状態を加える”]
          [attribute already end]
          [object 9:ケチ:“=Z  相手の欠点を数えたてる”] ] ]
  [agent 7:僕:2dc301]
  [goal [  [main 5:方式:1e8626]
          [modifier [          [main 3:計算:0ef08d]
                               [object 2:原価:0e8c01] ] ]
          [modifier 1:その:3d04df] ] ]
]
```

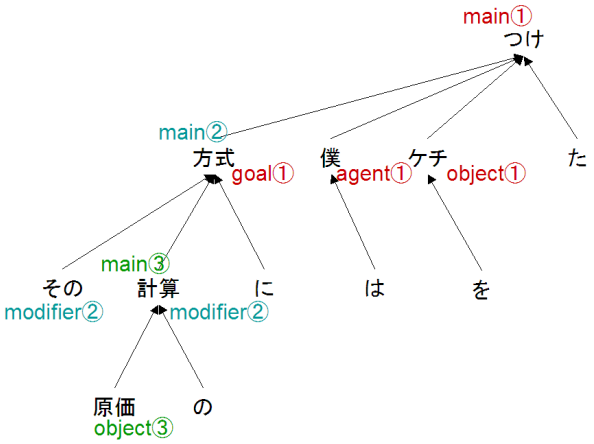


図 12　EDR の構文情報の例（日本語文）
左のリストによる意味構造記述を図 10 の依存構造木上に展開したもの

[*10] 前出の “the discovery of a new continent” という句の例の場合，「“discovery” に対して “continent” が object である」というタグ付けがなされる

大規模な日本語文に意味役割タグを付加したコーパスは他に存在しないため，日本語の意味解析研究では主にこの EDR コーパスが用いられている．文例数がかなり多く，構文的特徴・意味的特徴ともに豊富に記述され，さらに概念番号の整備など非常に情報量の多いコーパスであるが，データの品質の面で若干問題があるとも言われている．前述のように，非常に大規模なプロジェクトとして大量の人手を動員して作成されたため，分析作業者間でタグ付け基準に揺れが多いと言われており，また長期に渡って作成されたため作業時期によってもタグ付けの質に違いが見られるとも言われている．

2.1.5　コーパスまとめ

紹介した三つの意味構造付きコーパスの特徴を簡単にまとめる．

表 2　三コーパスの比較

特徴	FrameNet	PropBank	EDR
意味タグの粒度	詳細（フレーム毎に定義）	一般的（別途詳細定義）	一般的
構文的情報	句構造（チャンク）	句構造（構成素木）	依存構造
「述語」の種類	動詞，名詞	動詞のみ	すべて
意味タグ付けの単位	語，句，節	語，句，節	語
複層の意味タグ付け	なし	あり	あり
	構文より意味に重点	構文情報が豊富	語と語の関係に注目
その他の特徴	フレームの網羅性が不十分	一般性があり学習が容易	日英のコーパスあり
	初期の研究で利用	英語研究で主流	日本語研究で主流

2.2　既存研究

ここまで紹介してきた各種のコーパスに対して，文の意味役割要素を特定する意味解析の研究が既に数多くおこなわれている．ここでは，対象とするコーパスごとに代表的な既存研究を紹介する．

2.2.1　FrameNet の意味解析

英文のコーパスに基づいた意味解析の試みは，2002 年に Gildea と Jurafsky によっておこなわれた研究 [4]（以下 G&J と呼ぶ）がきっかけとなった．G&J は，FrameNet コーパスで定義された意味役割（Frame Element）を自動推定し，タグ付けするタスクをおこなった．

例：“She blames the Government for failing to do enough to help .”

↓

[JUDGE She] [PREDICATE blames] [EVALUEE the Government] [REASON for failing to do enough to help] .

学習アルゴリズムとして用いられたのは，コーパス中の文例から語句とその意味役割タグとのペアを抜き出し，語句のさまざまな特徴量 $features$ から，その語句が意味役割タグ r を持つ条件付確率

$$P(r|features) = \frac{\#(r, features)}{\#(features)}$$

を計算するというものであった．

この研究で，Gildea らは FrameNet コーパスの文に構文解析を施し，生成された構成素木中の語句ノードごとに割り振られる意味タグを推定するというアルゴリズムを提案した（図 13）．

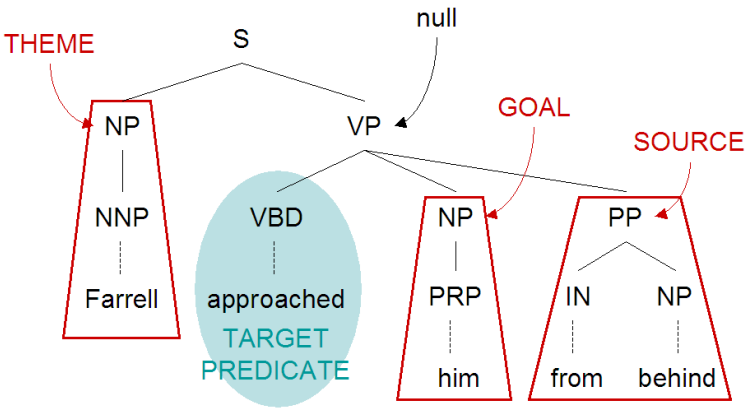


図 13　構成素木のノードごとに意味役割タグを推定：
NP,VP といったノードごとに特徴量を抽出し，そのノードが表す語または句に AGENT, THEME などの意味タグが付くか，何も付かない（null）かを条件付確率モデルを使って推定する

確率モデルの学習のための特徴量 $features$ には，以下のようなものが利用された．

1. Phrase Type
   該当句ノードの種類（NP, VP, PP など）．
2. Parse Tree Path
   該当句ノードの，述語ノードへの経路．（NNP ↑ NP ↑ S ↓ VP ↓ VBD など）
3. Head Word
   該当句ノードのヘッドワード
4. Position
   該当句ノードの，述語との相対位置（before or after）
5. Voice
   文の述語の態（受動 or 能動）

G&J で提案された「構文的特徴を利用した意味解析」は意味解析の研究に大きな影響を与え，以降の意味解析研究では G&J で提案された特徴量が頻繁に用いられるようになった．その後，学習アルゴリズムや特徴

量を工夫することで FrameNet の意味タグ付けの精度を上げる研究が数多くおこなわれた [5][6] が，G&J でも指摘されていたように，

- FrameNet はフレームごとに細分化しすぎた意味役割が定義されており機械学習に向かないこと
- フレームの網羅性が少なく一般的な文書データを処理できる意味解析システムへの汎化が難しいこと

などから，研究の中心は次に述べる PropBank コーパス上でのタグ付けタスクに移ってきている．

2.2.2　PropBank の意味解析 〜CoNLL 2004 Shared Task〜

機械学習による自然言語処理研究の分野では，CoNLL（Conference on Computational Natural Language Learning）と呼ばれる国際会議において，毎年何らかの自然言語処理タスクを共通課題（Shared Task）として取り上げ共通のデータセットを用いて言語解析システムの性能を競いあうという試みが行われているが，2004 年と 2005 年には PropBank の意味タグ付けが課題として取り上げられ，数多くの意味解析モデル構築の研究が発表された．ここでは，2004 年の Shared Task で最も良い成績を出した Hacioglu らの意味解析モデル [11] を紹介する．

• CoNLL 2004 Shared Task

2004 年度の課題は，PropBank データの中から簡単な構文的特徴量のみを使って，意味役割タグである Argument を推定するというものである．モデルの学習・評価に利用できるデータ[*11]は，以下のようなフォーマットで配布された．構文的特徴としては，形態素とその品詞以外には句のチャンク（塊）という簡単な情報のみが与えられ，完全に構文解析された構成素木の情報は Not Given とされた．

[形態素]	[品詞]	[句]	[節]	[固有表現]	[述語]	[意味ラベル 1]	[意味ラベル 2]
The	DT	B-NP	S*	O	-	(A0*	*
San	NNP	I-NP	*	B-ORG	-	*	*
Francisco	NNP	I-NP	*	I-ORG	-	*	*
Examiner	NNP	I-NP	*	I-ORG	-	*A0)	*
issued	VBD	B-VP	*	O	issue	(V*V)	*
a	DT	B-NP	*	O	-	(A1*	(A1*
special	JJ	I-NP	*	O	-	*	*
edition	NN	I-NP	*	O	-	*A1)	*A1)
around	IN	B-PP	*	O	-	(AM-TMP*	*
noon	NN	B-NP	*	O	-	*AM-TMP)	*
yesterday	NN	B-NP	*	O	-	(AM-TMP*AM-TMP)	*
that	WDT	B-NP	(S*	O	-	(C-A1*	(R-A1*R-A1)
was	VBD	B-VP	(S*	O	-	*	*
filled	VBN	I-VP	*	O	fill	*	(V*V)
entirely	RB	B-ADVP	*	O	-	*	(AM-MNR*AM-MNR)
with	IN	B-PP	*	O	-	*	*
earthquake	NN	B-NP	*	O	-	*	(A2*
news	NN	I-NP	*	O	-	*	*
and	CC	I-NP	*	O	-	*	*
information	NN	I-NP	*S)S)	O	-	*C-A1)	*A2)
.	.	O	*S)	O	-	*	*

[*11] PropBank 中の Wall Street Journal の section15〜18 で学習，section20 でパラメータ調整，section21 でテストをおこなう

※句チャンクや固有表現は，形態素ごとに名詞句（NP）や動詞句（VP）の始まり（B-），中（I-），外（O）という IOB 方式で表現されている．

• Hacioglu 2004

Hacioglu らは，この Semantic Role Labeling の問題を CoNLL 2000 での課題であった**チャンキング（chunking）**[*12]と類似の問題と定義し，“Semantic Role Chunker” という形で意味解析アルゴリズムを構築した．チャンキングとは，文の形態素列を，下のように句ごとの塊りに分けることである．

例：“The Sun Francisco Examiner issued a special edition around noon yesterday that was filled entirely with earthquake news and information .”

↓

(NP The Sun Francisco Examiner)（VP issued）(NP a special edition)（PP around）(NP noon)（NP yesterday）（NP that）（VP was filled）（B-ADVP entirely）（PP with）（NP earthquake news and information）（O .）

チャンキングでは，「形態素ごとに句の始まり（B-XXX）か内部（I-XXX）か外部（O）かをひとつずつ順に判定していく」という解析アルゴリズムが考えられていた（図 14）．Hacioglu らはこのアルゴリズムをその

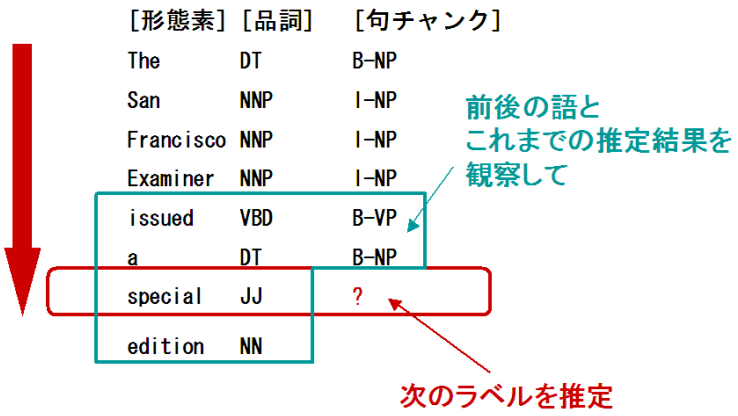


図 14　文の前から順番にチャンク構造を推定

まま意味タグの推定に拡張し，意味タグを IOB 方式で推定していくモデルを作成した．さらに，単純な形態素の並びよりも意味タグの付く句チャンクを基本単位として同様の意味タグ推定をおこなう（図 15）方がより精度の高い推定ができることを示した．

[*12] CoNLL 2001 では節境界の判定，2002 及び 2003 では固有表現抽出が課題とされた．すなわち，CoNLL 2004 で与えられた特徴量は，これまでの研究である程度の自動解析が可能になっているものである．

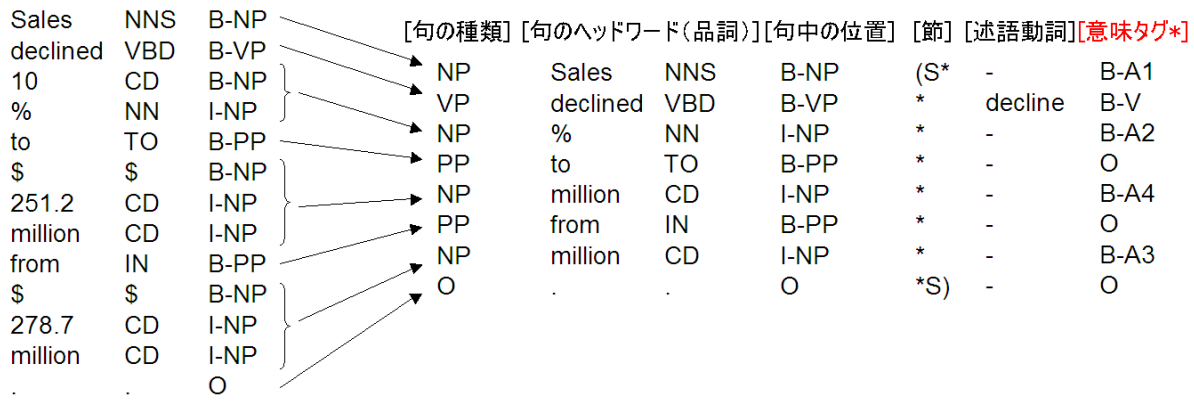


図 15　ヘッドワードが代表する句チャンクを単位として意味タグを推定

Hacioglu らはこのアルゴリズムで，Shared Task のテストセットに対して Precision 72.43% ，Recall 66.77% ，$F_{\beta=1}$ 値 69.49 [*13] [*14]の精度での意味ラベル付けを実現した．この他にも多くの研究者がさまざまなアルゴリズムで意味解析システムを考案し，$F_{\beta=1}$ 値にして 50% 〜 65% 強の精度での意味ラベル付けが可能であることを示した．

2.2.3　PropBank の意味解析 〜CoNLL 2005 Shared Task〜

句チャンクのような単純な構文的特徴しか与えられなかった 2004 Shared Task に対して，2005 Shared Task では Penn Treebank に記述された構成素木の情報をもフルに利用して同様の意味ラベル付けをおこなうことが課題とされた．CoNLL 2004 で利用された句チャンクや節と，2005 で追加された構成素木表現の対応関係は下のようになる．

［形態素］	［句チャンクと節］	［フル構成素木］
The	(NP* (S*	(S(NP*
$	* *	(ADJP(QP*
1.4	* *	*
billion	* *	*))
robot	* *	*
spacecraft	*) *	*)
faces	(VP*) *	(VP*
a	(NP* *	(NP*
six-year	* *	*
journey	*) *	*
to	(VP* (S*	(S(VP*
explore	*) *	(VP*
Jupiter	(NP*) *	(NP(NP*)
and	* *	*
its	(NP* *	(NP*

*13 Precision（適合率）：$\frac{\text{システムがタグを付与したもののうちの正解事例数}}{\text{システムがタグを付与した事例数}}$ Recall（再現率）：$\frac{\text{システムがタグを付与したもののうちの正解事例数}}{\text{真にタグが付与されるべき事例数}}$
$F_{\beta=1}$（適合率と再現率の調和平均）：$\frac{2*Precision*Recall}{Precision+Recall}$

*14 固有表現（Named Entity）などの特徴量を用いず，形態素・品詞・句チャンクなど配布データから得られる基本的な構文的特徴量のみを用いた場合の精度は Precision 69.04% ，Recall 54.68% ，$F_{\beta=1}$ 値 61.02 とされている

```
16          *     *          *
known       *     *          *
moons       *)    *)         *)))))))
.           *     *)         *)
```

より精密な解析をおこなった結果である構成素木の構造を与えられるので，2004 Shared Task では作れなかった特徴量を利用することができるようになった．構成素木を意味ラベル付けに利用するというタスクは前出の G&J でかなり詳細に研究がおこなわれていたため，ここで得られた知見を容易に盛り込むことができた．特に，G&J で提案された特徴量のひとつである「構成素木上で句ノードから述語ノードへ向かう経路（Parse Tree Path）」は意味ラベルの境界を発見するのに非常に有効であることが分かっており，CoNLL 2005 でも多くの意味解析モデルがこの特徴量を利用した．提案された意味解析モデルの精度を概観すると，フルに解析された構成素木を利用すれば，意味ラベル付けの精度を $F_{\beta=1}$ 値にして 65% 〜 80% 程度にまで向上できることが分かった．

このように，CoNLL 2004 及び 2005 の Shared Task で数多くの意味解析モデルが提案され，計算機による自動的な意味タグ付けをかなりの精度で実現できることが示された．ただし，Shared Task では構文的特徴や固有表現，voice（態）などが既知のものとして与えられているため，ほとんどの提案モデルの精度はこれらを Given のものとして扱っており，ゼロから自動解析した精度を計算していないものも多い．また，特徴量として意味役割そのものにかなり近い概念分類などを人手で付加している [14] 場合もあり，まったくの自動解析で上記のような精度の解析が可能であるとは言いにくい状況にある．

2.2.4 EDR の意味解析

ここまでは欧米で中心的におこなわれている英語の意味解析研究を紹介してきたが，最後に，日本語における意味解析研究を紹介する．日本語では，EDR コーパスに記述されている概念関係子を推定するタスクが一般的になっている．概念関係子は格文法理論における深層格をベースにしていることから，深層格の推定とも呼ばれている．

日本語では，語順がかなり自由に変化して文が生成されるため，英語で一般的な句構造に基づいてデータを作成することが難しく，より簡単な単語・文節間の修飾関係を表す依存構造（係り受け構造）が中心的に利用されている．前節で述べたとおり，EDR でも構文情報としては依存構造が記述されている．一方，意味情報である概念関係子は「ある語が他の語との関係において持つ意味」であり，そのため概念関係子は構文的依存関係上に付与されているとも解釈できることから，この依存関係を中心にして深層格の推定がおこなわれる．また，日本語では表層格として文に現れる**助詞**が深層格に密接に関係しており，例えば「が」「は」などの格助詞は主語としての動作主格（Agentive）を導くことが非常に多いため，名詞と助詞のセットを単位として深層格を推定することも多い．

構文解析によって依存関係を与えられたときに深層格を推定するタスクに特化した研究例としては，小山ら [29] や渋木ら [30] の研究成果がある．小山らは，EDR の共起辞書中から「名詞＋格助詞などの表層格」のペアをセットとして抜き出し，その組み合わせパターンとその間にある深層格対応を頻度付きでデータベース化することで，未知文の「名詞＋表層格」パターンから対応する深層格を推定するモデルを構築した．このモデルでは，未知文例 648 文に対して 70.8% の正解率（Accuracy）で正しい深層格を推定できたとしている．ま

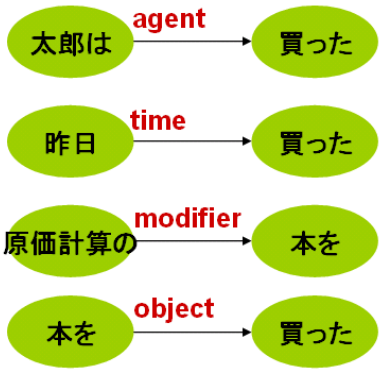


図 16　依存（係り受け）関係ごとに関係子（深層格）を推定する

た渋木らは，「単文中ではひとつの述語は同じ深層格を二つ以上持たない」という**一文一格の原則**（図 17）を制約条件として用いることで深層格推定の精度を向上することを試みた．この工夫を加えた深層格推定モデルを用いて，総依存関係数 5470 個のテストデータで 66.0% の正解率を達成している．

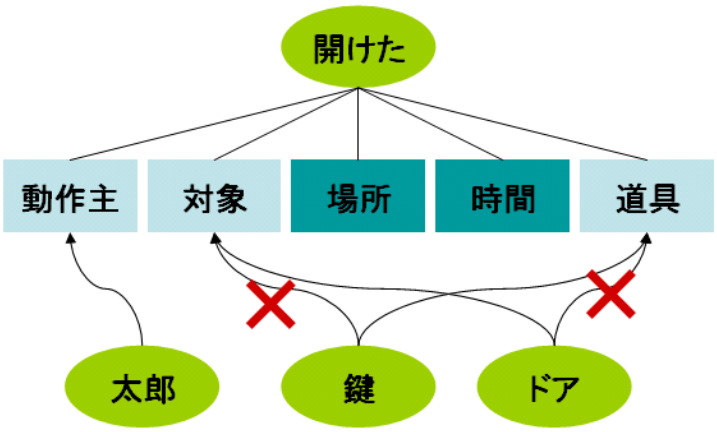


図 17　一文一格の原則

一方，深層格の推定を含めた意味解析を自動でおこなうことを意識し解析システムとして実装した例として EDR コーパスの意味解析でもっとも有名なのは，青山学院大学で開発された SAGE（Semantic frame Automatic GEnerator）[27][28] である．

SAGE は，係り受け解析器 CaboCha[*15] を前処理に利用し，生成された係り受け関係（依存関係）を入力データとして，係り受け関係のペアごとに EDR 辞書から計算した確率値を用いて対応する深層格を推定し，元の係り受け木（依存木）に深層格タグを付加することで「意味グラフ」として出力するシステムとして実装されている．深層格推定の正解率は 2003 年 1 月時点で 80.6% と報告されており，解析精度の面でもかなりの性能を示している[*16] ．意味解析を全て自動で行えるシステムとして唯一実装されている非常に貴重な研究成

[*15] CaboCha : http://chasen.org/ taku/software/cabocha/
[*16] ただし，SAGE はコーパスをもとに作成された共起辞書などから確率値を算出しているため，コーパス中の文例によるテスト・評

果であり，その出力結果である意味グラフを自動要約，文書分類などの応用アプリケーションに利用する研究[26][25] なども進められている．

以上のように日本語についても意味解析の研究は活発におこなわれているが，一方でシステムの評価に関しては問題な点がある．上で紹介した三つの研究はそれぞれ解析手法が異なり，そのため精度評価用のテストセットの作成もそれぞれ独自におこなっている．また個々の意味役割タグの推定精度もあまり詳細に報告されておらず，そのため英語研究における Shared Task のように解析システムの性能を客観的に評価する基準がなく，意味解析技術の現状を捉えにくい状況である．

価の結果には若干の疑問がある．

3 現状分析と提案

これまで，機械学習による意味解析の既存研究を概観してきた．本章では，目的で述べた「意味解析システムを実際の文書処理に利用しやすい環境作り」の立場に立って既存の意味解析システムの問題点を分析し，それに対する本研究の提案を述べる．

3.1 現状分析

3.1.1 構文的特徴の利用のコストについて

前章で紹介した G&J やそれに続く CoNLL Shared Task の種々の研究によって，英語の意味解析においては，従来から構文的解析の中心であった「句構造」を利用することでかなり高い精度を実現できることが分かった．特に，同レベルの大きさの句をまとめていくことでできる「構成素木」を特徴量として利用することで精度の向上が見られたことから，句構造とこれに基づく構成文法が英語の意味解析において非常に重要な情報を有していることが推察される．

Shibui[7] は FrameNet の意味タグ付けタスクにおいて構文的特徴をほとんど用いずとも意味タグ付けの精度があまり低下しないことを示し，構文的解析の結果に依存しない意味解析の可能性を示唆したが，実験に用いた文例数が非常に少なく，また文の所属するフレームを既知のものとして，同じような文例の集合であるフレームごとに学習・テストをおこなっていることから，精度維持の主な理由は学習・テストセット内の文例の偏りにあるものと推察される．また FrameNet の定義フレームの網羅性の低さを考えると未知文への頑強性の維持が困難だと考えられるため，一般的な文を解析するシステムとしての応用可能性が十分に示されたとは言えない．総合的に見ると，やはり意味解析における構文的特徴，特に句構造を利用することの有効性は否定できない[*17] ．

ところが，句構造に基づく構成文法に沿って作成されたコーパスは，言語によっては利用することができない．句構造文法はもともと言語普遍的なものとして考案されているため，日本語でも同様の句構造に基づく構成素木を作成することはできる．しかし前述のように，日本語では語順が比較的自由に変化できることから，句構造だけでは文の構造を十分に捉えることができない．そのため，EDR でも構文情報としては句構造ではなく比較的単純な依存構造が記述されている[*18] ．

また，句構造に基づく構成素木は，作成するのにある程度の知識を必要とする．句構造は下記のような「句構造規則」に基づいて生成される．

- S → PP VP
- PP → NP P
- VP → PP VP
- VP → V AUXV
- NP → N

[*17] むしろこの実験は，フレームという手がかりが与えられた際の統計的意味役割推定の容易さを示している．ただし，一般的な文に対してこのシステムを適用するには，フレームが自動解析でどこまで推定できるのか，フレームの階層性などをどう扱うのか，フレームの網羅性の欠如の問題をどう解決するのかが課題となる．

[*18] 東京工業大学 田中・徳永研究室では EDR の文に句構造に基づく構成素木を付与する作業をおこなっている．

句構造に基づく構成素木をコーパスとして作成するには，タグ付け作業者がその言語の句構造文法についての一定の知識を持つことが必要とされる．これには，単にタグ付け作業者がその言語を母国語とするネイティブスピーカーである，というだけでは不十分である．

以上のように，英語における意味解析アルゴリズムで有効性が認められている文の句構造情報は他言語では整備が難しく，現段階では英語の意味解析研究の成果を利用して他言語の意味解析システムを構築しにくい状況にある．また計算言語学研究の立場から見ると，人間の自然言語の計算論的な普遍的性質を解明するという目標に迫るためには，他言語では利用しにくい特徴量に依存した学習理論のままでは不十分であるともいえる．

3.1.2 「述語」の定義について

前章の既存研究の紹介でも述べたように，近年の意味解析研究（主に PropBank の意味役割タグ推定）では上述の句構造を含むかなり詳細な文の特徴量を既知のものとして利用しており，その特徴量を生成するためのコストを度外視してしまっている場合が多く，これらの現状が総合的に自動意味解析システム[*19]構築への障害になっていると考えられる．

平文からの自動解析をおこなう場合一般的なシステムでは，

1. 平文に構文解析を施した後，
2. 文中の全ての動詞を述語とみなし，
3. それぞれが述語となる場合の特徴量を計算し，
4. 全てのパターンについて意味タグの推定をおこなっている．

こうすることで解析の焦点となる「述語」を特定し，語句と述語との関係性を表す特徴量を用いた意味解析アルゴリズムを使用することができる．こういったことが可能なのは，PropBank をベースとする意味解析システムでは「述語」になりうるのが文中の動詞に限られているためである．しかし，実際には「述語」にあたるものは文中の動詞に限られるわけではない．

例えば FrameNet の “Clothing” フレームでは “jacket” や “coat” などの「衣服」が述語として定義され，この周りに 意味役割（Frame Element）として Material, Style などの修飾語的なものが登場することもある．EDR でも，概念関係子タグとして possessor や material など名詞に対しての関係性が定義されており，“discovery” のような動作を表現する名詞に対する意味役割も記述されている．Hacioglu は FrameNet の意味解析のために，述語（Target Word）を与えられないと得られない特徴量を排除し述語となる語をも意味タグと同時に推定によって発見するしくみを提案し，Precision 67.6% , Recall 55.9% を達成したと報告している [10] が，学習・テストに用いた文例数が極端に少なく（学習 3000 文，テスト 750 文），また似た文例の多いコーパスであることなども考えると，この精度もやはりコーパスの偏りによるものと考えられる[*20] .

そもそも「述語」が「他の語との意味関係において主要素となる側の語」というものであるとすれば，述語

[*19] 数少ない実装例のひとつとしては，Hacioglu による PropBank のタギングのデモ（http://sds.colorado.edu/SERF-II/）が公開されている．

[*20] 筆者も FrameNet のデータ 17000 文に対して同様の実験を試みたところ，特徴量として単語と品詞のみを使った場合でも Precision 62.9% , Recall 45.6% が確認できた．しかし，名詞が述語となるフレームでは述語そのものにもそのフレームを代表するタグ（“Clothing” フレームなら “Garment” など）が付くことが多く，コーパス内には同様の述語を含む文が数多く用意されているため簡単に高い精度が出てしまうことが分かった．

動詞に限らず，文中のかなり多くの語が他の何らかの語に対する「述語」になりうるのではないかと考えられる．動詞を述語とする場合はそのまわりに動作の主体者や対象物などが登場するが，名詞を述語とする場合はその名詞の性質を示す意味役割が登場するはずである．こうした関係性まで含めて解析の対象とするためには，文の中心となる「述語」を少数だけ定義してタグ付けしたコーパスを元に意味解析システムを構築するのでは不十分なのではないかと考えられる．

3.2　提案 〜依存構造に基づく意味解析〜

本論文では，上で述べた問題点を解決し多言語にわたって意味解析をおこない構造化文書を作成しやすい環境を構築するために，意味解析モデルの学習において**文の句構造ではなく依存構造を中心に利用する**ことを提案する．

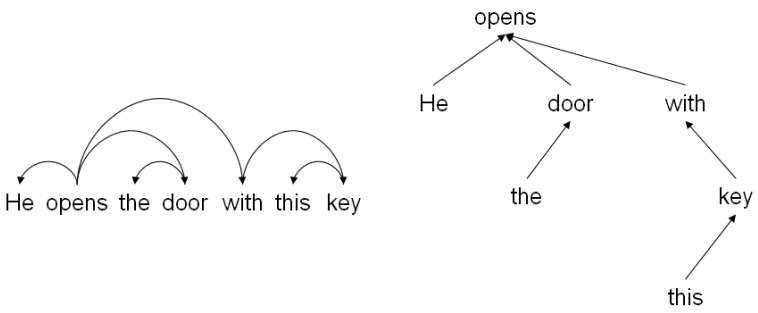


図 18　依存構造：(左) フラット表現 (右) 依存構造木表現

依存構造とは，「語と語の間にあるなんらかの修飾関係」による結合構造である．語の形態や意味は全て他の語との結合に基づいて決定されるため，構文などの「形態」と「意味」との関係を考えていく上でもっとも基本となる情報が語間の依存関係であるといわれている [36][37]．句構造のような句や節といった構成素間の関係は語の依存関係から派生する二次的なものであるとも考えられている．直感的に見ても，依存関係は語同士の結びつきをボトムアップに形作る原始的な情報のように思われる．そのため，どの言語においても依存関係の情報を得ることは比較的容易だと考えられる．一方で，依存関係は語と語の意味的な結びつきにも深い関係を持っているため，これを特徴量として用いることでより言語依存性の低い意味解析モデルを作成できるのではないかと考えられる．

英語の解析では早くから句構造に基づく構成文法が大きな成果を出していたため，文法理論的には正反対の視点を持つ依存文法に基づく方法はほとんど用いられてこなかった．電子テキストデータとしても句構造に基づく構成素木を採用した Treebank などが主流となったため，依存構造木をデータとして作成している例はあまりなかった．しかし近年，英文の依存構造に注目する動きがあり[*21] ，一部では依存構造に基づくデータを独自に作成して研究が試みられている．このため英語の解析においても依存構造を利用することが可能な条件が整ってきている．

[*21] 2006 年度の CoNLL Shared Task はオランダ語やスウェーデン語の統計的依存構造解析となった．

以下に，依存構造に基づいて意味解析をおこなうことの根拠と，考えられる利点を説明する．

3.2.1　根拠 ～述語-項構造との比較～

言語学研究における依存文法の研究者は，文の依存構造が文の意味とも密接な関連を持っていることを主張している．これは，自然言語処理研究で定義されている意味解析問題においても確認できる．意味解析問題は，前述のように「文中の「述語」に対する周辺の語の「意味役割」を決定すること」といった形で定義されている．述語とその周りの意味役割を持つ語（PropBank に倣って「項」と呼ぶ）とは，これまで主に用いられてきた構成素木上では図 19 のような位置関係にある．一方，文中の個々の語がノードを形成する依存構造木上では，述語と項は図 20 のような位置関係になる．依存構造木では修飾関係にある語の間に直接のリンクが張られるため，「述語」とそれに対して意味役割を持つ「項」とは，ほとんどの場合依存関係にある．この依存構造木の上では，述語ノードと句にあたる部分木[*22]のルートノードとの間に存在する関係性として「意味役割」を推定することになる．

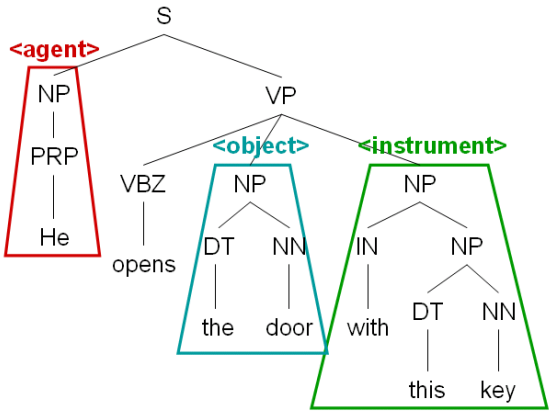


図 19　述語と意味役割の位置関係（構成素木）

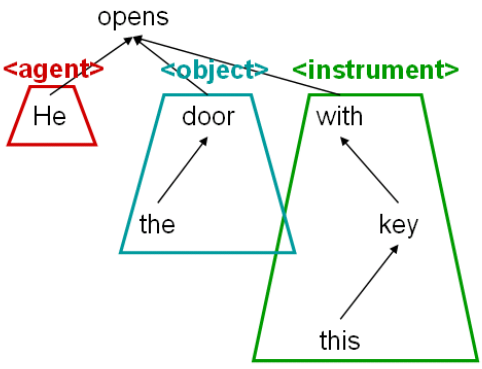


図 20　述語と意味役割の位置関係（依存構造木）

[*22] 依存構造木における部分木は，構成素木の句チャンクに対応することが多い．このことから，依存構造は句構造における構成素木と句チャンクの中間に位置する情報を持っているのではないかとも考えられる．

また，依存木においては，複数の「述語」が存在する場合も，図 21 のように再帰的構造によってその「述語」に関する意味構造を入れ子で表現することができる．

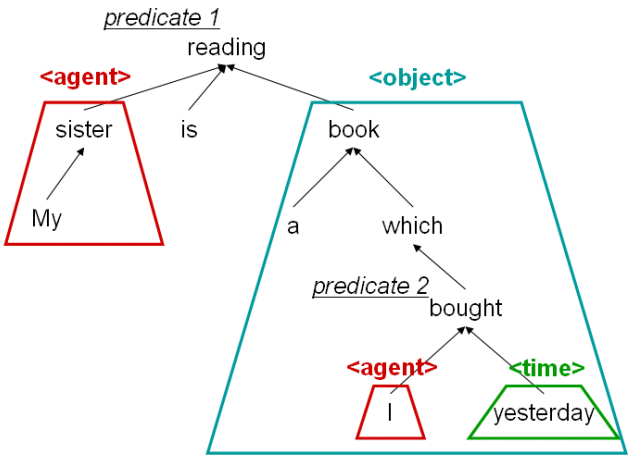


図 21　依存構造木と複層の述語-項構造

すなわち，意味解析問題における「意味役割」を，「依存関係にある係り先への語との間の意味関係」という形で定義しなおせば，依存構造に基づいて意味構造をより直接的に表現することができるようになると考えられる．また，従来の意味解析問題における述語-項構造とほぼ同様のものを表現することができるため，現在自然言語処理研究においておこなわれている意味解析問題にも，この依存構造を中心的に利用することは非常に有用なのではないかと考えられる．

3.2.2　利点 ～コーパス作成のコスト～

依存構造はもともと語と語の局所的な関係性から成り立つため，句構造と比べると比較的プリミティブな情報であると考えられる．そのため，文の依存構造を見つけるのはおそらく句構造よりも（または少なくとも構成素木よりも）容易であり，対象言語を母国語とする人ならばほとんどの場合間違えずにおこなえると思われる．

日本語を母国語とする人を例にとってみても，「太郎は次郎が昨日買った本を読んだ」という文を解釈する場合，「太郎は読んだ」「本を読んだ」「買った本を」「次郎が買った」「昨日買った」といったふうに依存関係を直感的に容易に発見することができる．「太郎は「次郎が昨日買った本」を読んだ」という句に基づく大局的な文全体の構造は，これらの依存関係を利用して二次的に生成される情報なのではないかと考えられる．

現在の統計的学習によって生成された言語解析モデルは，モデルの学習に利用したコーパスの性質に強く依存するため，コーパスとは性質的に異なる分野の文書データに対しては解析能力が低下する．そのため，実際の問題に統計的言語解析モデルを利用する際には，その分野の文書データで一定量の学習用データを用意してモデルを学習しなおすことが望まれる．すなわち，構造化された文書データが普及していない現段階では，意味解析システムのユーザーが自身の関心領域の文書のタグ付けを行う必要が生じる場合がある．その際，ユーザーに複雑な文法的知識を求めることは難しいため，依存構造のような比較的単純な構文構造のみを特徴に利用して意味解析ができるのであれば，ユーザーに要求することになるタグ付けの負担も軽くすることができ

る．依存関係の発見は構成素木のような文全体を見た解釈の必要がないので，タグ付け作業においても部分部分だけをみて局所的に作業することができるのではないかと考える．

3.2.3 利点 ～文データ記述の標準化～

依存構造が比較的原始的な情報であり，あらゆる言語で表現しやすいものだと考えられることから，将来的な文書データの構造化においても言語普遍的な標準的記述スタイルとして依存構造が適しているのではないかということが考えられる．

序論で述べたように，広範囲で取り扱われる文書データについてひとつひとつの文に構文構造や意味構造を記述して流通させるという取り組みはまだ普及していない．これには，文に対する構造のタグ付けスタイルの標準化がおこなわれていないことも大きな原因となっている．この現状に対して，文書データ中の個々の文にタグ付けをおこない構造化する記述方式を現在唯一提唱している例として，産業技術総合研究所の橋田らによる**大域的文書修飾（**Grobal Document Annotation：GDA**）**[*23] [20] がある．

GDA とは，あらゆる言語の文書データに対して文の意味的構造を共通のスタイルで記述するための，XML を基盤とするタグ付け体系である．橋田らは，意味的な構造が明示された電子テキストデータをインテリジェント・コンテンツと名付けている．GDA が提案するタグ付け体系によって文書へのアノテーションの標準化をおこない，タグによって構造化されたインテリジェント・コンテンツの普及を促すことによって，データの持つ意味を人間と計算機との間で効率よく共有し知的生産性の向上を目指す「セマンティックコンピューティング」の実現を目指している．

GDA では意味のタギングに焦点を置いており，

- 意味解析研究の意味役割に対応する**主題役割（深層格）**
- 文間の意味関係である**修辞関係**
- 代名詞や指示詞の指す対象を発見する**照応**

を基本的なタグ付け対象としている．このうち一番目の「主題役割」は，例えば図 22 のように記述される．主題役割のタグは自然言語処理研究における意味役割タグとほとんど同等のものであり，意味解析研究の成果を直接的に活かすことができる．意味解析システムで文書データを自動的に意味タグ付けし，GDA 形式で出力することができれば，構造化された文書データの普及にとって大きな促進力になると考えられる．このため，本研究でも解析結果を GDA 形式で出力することを念頭に置いている．

ところで，GDA は文の意味構造を整備することに焦点を置いているとはいえ，意味構造を表現するための最低限の前提として簡単な統語構造の記述の仕組みを備えている．そして，GDA ではその表現力に汎用性と冗長性を持たせるため，係り受け（依存）構造を統語記述の基礎としている．GDA は現在，日本語と英語の二ヶ国語についてアノテーションマニュアルを公開している．これに従って例文の統語構造を日英ともにタグ付けしてみると，図 23 のようになる．GDA では，XML の木構造において兄弟関係にあるノードのうち，末尾に “p” が付くノードが “p” の付かないノードに対して係っている（依存している）と定義している．日本語でも英語でも，基本的に動詞がその他の語を支配している構造が多いため，図 23 では<v></v>が依存構造

[*23] http://i-content.org/gda/

図 22　GDA による文中の意味役割のタグ付け：
"He opens the door with this key." のタグ付け．opr 属性の値によって，"He" が動作主格（agt），"the door" が対象格（obj），"with this key" が手段・道具格（mns）であることが示されている．

図 23　日英で共通の統語構造記述法

木上のルートノードに該当している．

このように，依存構造は言語依存性の低いプリミティブな情報であるため，将来的な構造化文書の普及のための標準アノテーション体系においても利用しやすいことが分かる．このため，句構造に基づく複雑な構成素木を作らずとも依存構造で十分な意味解析が可能であるならば，GDA のような標準アノテーションフォーマットへの文書の自動変換を効率的におこなうことが容易になると考えられる．

3.3　提案システム

依存構造を利用して英文の意味解析をおこなう提案システムの概要を示す（図 24）．

文の依存構造と意味タグの付いた文例を大量に有するコーパスから機械学習によって意味役割タグの推定をおこなう意味解析モデルを生成し，平文の入力に対して形態素解析・依存構造解析をおこない，依存構造木中の依存関係ひとつひとつに対して意味タグ付けをおこない，意味タグの付与された依存構造木としてまとめ上

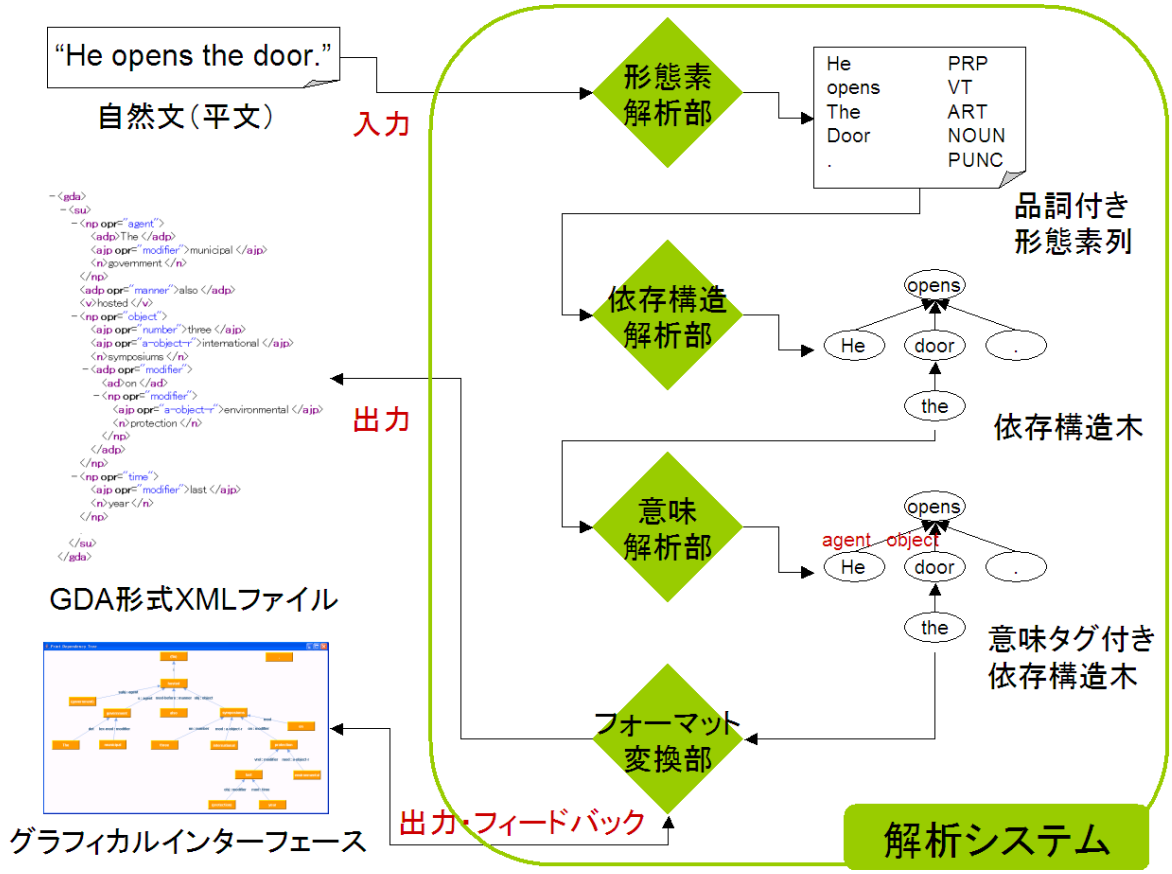


図 24　提案システム概要

げてひとつのタグ付き文データとして出力する．

今回，意味解析モデルの機械学習に用いるデータとしては，前章で紹介した EDR コーパスを採用する．理由は，以下の二点である．

1. 英文の意味タグ付きコーパスとして唯一，依存構造情報が記述されている．
2. 同様のタグ付けルールに基づいて記述された日本語コーパスも存在している．

EDR の英語コーパスは，作成された日本以外ではポピュラーでなく，英語の自然言語処理研究で利用された例はない．そのため，従来の意味解析研究の結果との直接的な比較をすることができなくなってしまうが，英文の依存構造が記述された唯一のコーパスとしてこれを採用する．最近では主辞規則と呼ばれる変換規則を用いて構成素木を依存構造木に変換することが可能になり，Penn Treebank から依存構造木データを生成することもおこなわれているが，もともと依存構造をもとに意味関係をタグ付けしてあるコーパスを用いる方が，言語依存性の低い依存構造に基づく意味解析の可能性を評価するのに適しており，今回の目的には適していると考えた．また，特徴量として依存構造を利用することの利点として言語依存性の低さを期待しているため，これを確認するためには同様の特徴量に基づいた日本語文の解析が可能なコーパスが適している．

Web で流通しているような大量の文書データを意味的情報で構造化し，効率的に利用することができる環境を作るためには，前述の GDA のような標準化されたタグ付けルールに則ったファイル形式で結果を出力できることも重要であると考える．また，上記のようなシステムで自動的に文データを解析することはできるが，現段階ではまったく自動に完全な解析をすることは不可能であり，結果を人間が見て手直しをする必要がある．また，機械学習のためのコーパスを作成するタグ付けツールとしても，途中で解析結果のフィードバックを可能にする必要がある．そのため，マウスなどで操作・修正の可能なグラフィカル・ユーザーインターフェース（GUI）で解析結果の依存構造木を表示する機能なども必要になると思われる．

3.4　関連研究

依存構造を意味解析システムの学習アルゴリズムに利用するにあたって，類似の関連研究と基礎となる技術の研究例を紹介する．

3.4.1　依存構造に基づく PropBank 意味解析

意味解析に文の依存構造を利用することは，既に Hacioglu[12] が試みている．Hacioglu は，PropBank の構成素木を主辞規則を使って依存構造木に変換し，PropBank の意味役割タグである Argument タグを付加して DepBank と名付け，これを機械学習して Argument タグを推定するシステムを提案している．

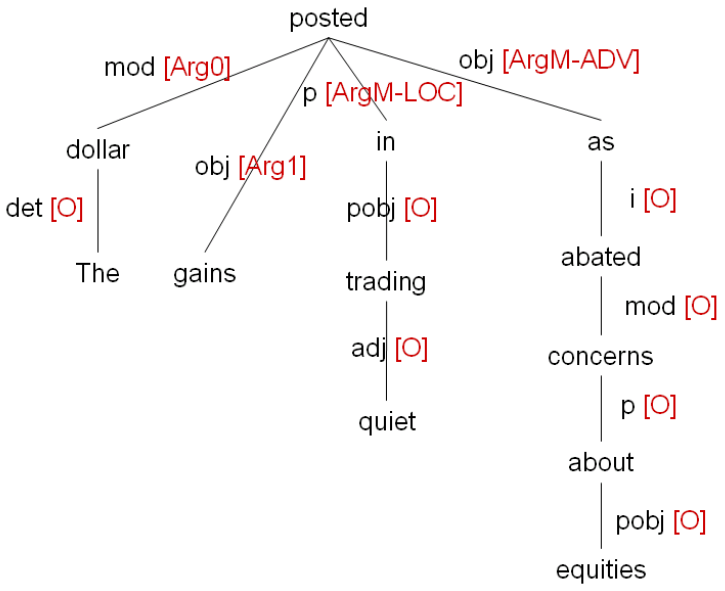


図 25　DepBank の依存構造木と意味タグ

Hacioglu は前に紹介した PropBank の意味タグ付けシステム [11] で，単語ごと（word-by-word）や句ごと（phrase-by-phrase）に意味タグの推定をおこなうことを提案しているが，この研究では依存構造木における依存関係のエッジごと（relation-by-relation）に推定することを提案している．生成した図 25 のような依存構造木から依存関係エッジを抜き出して列挙し，それぞれに以下のような特徴量を抽出して意味タグ推定の学習・推定をおこなっている．

1. Type
   依存関係エッジの種類（主辞規則による木の変換時に同時に生成．主語 subj, 目的語 obj など）
2. Family membership
   依存関係エッジの，述語との位置関係（child, grandchild など）
3. Position
   依存関係エッジの係り先の単語の，述語との字面上の相対的位置関係（before or after）
4. Headword（PoS）
   依存関係エッジの係り先の単語（その品詞）
5. Dependent word（PoS）
   依存関係エッジの係り元の単語（その品詞）
6. Path
   依存関係エッジの，述語への経路

この他に，述語の持つ子ノードの品詞の組み合わせや，述語と子ノードとの依存関係エッジの種類の組み合わせなどをグローバルな特徴量として用いている．

Hacioglu は CoNLL 2004 Shared Task のデータで学習・推定・評価をおこない，2000 文強のテストセットで $F_{\beta=1} = 84.6$ とかなり高い精度を示したとしている[*24] ．ただし，Hacioglu 自身が述べているように，PropBank の意味タグ（Argument タグ）は本来構成素木にタグ付けされており，依存構造木への変換の際に一部のタグが消失してしまうため，他のシステムとの直接的な比較をすることはできない．

Hacioglu のおこなった実験は，文の依存構造を意味解析に利用することの有用性を一部実証しているといえる．しかし，あくまで PropBank 中で定義された「述語」に対する意味タグ推定をおこなっており，述語が与えられていることを前提とした特徴量をモデルの学習・推定に用いているため，述語として文中の特定の語が与えられた条件下での意味タグ推定であることに変わりはない．本論文では，提案で述べたように PropBank コーパスよりもやや広義の「述語」を定義し，依存関係にある語に対しての局所的な意味関係を直接的に解析する方法を提案している．

3.4.2 MINIPAR

文の依存構造を自動解析することのできるソフトウェアはそう多くないが，制約に基づく文法理論によって文の依存構造を解析するシステムとして，MINIPAR が挙げられる．MINIPAR は英語の依存文法に関する知識を制約ルールとして詳細に書き込み，このルールを用いて文を解析するプログラムである．複数の解が存在する場合は，コーパスで学習した確率値によってもっともありそうな依存構造木を選ぶことで最終的な依存関係を決定し，出力する．

単純な依存構造解析をするだけでなく，依存関係の種類（主語 subj, 目的語 obj などの表層格）も認識できる．また節の範囲や関係子の先行詞なども見つけることができ，さらに本来意味的に存在するはずの主語などのトレースも予測することができる．形態素解析機能も実装しているため，平文を入力として上記のような結果を出力することができる．データから統計的機械学習によって作成されたシステムではなく，英語文法に関する豊富な知識を用いて作られているため，他の言語などへ応用ができる汎用的なシステムではないが，現状で利用できる英語の依存構造解析器の中ではもっとも性能の高いもののひとつであるといえる．

[*24] この精度は全体の平均値とみられ，それぞれの意味タグごとの推定精度は報告されていない．

3.4.3 機械学習による依存解析器

上記の MINIPAR に対し，データから機械学習によって依存構造解析モデルを構築する方法も既に考案されている．山田が考案した依存構造解析アルゴリズム [17][18][19] は，依存構造木を学習して作成したパターン識別器 Support Vector Machine（SVM, 後述）を用いて，決定論的に語の依存関係を解析していくものである．本論文の提案システムではこの依存構造解析アルゴリズムを利用するため，ここで詳しく紹介することにする．

この依存構造解析アルゴリズムは，形態素解析され品詞タグの付いた形態素列を入力とし，この形態素列を前から順番にスキャンし，隣り合う二つの形態素が依存関係にあるか否かをパターン識別器 SVM で判定していくというものである．SVM によって「隣の単語に依存する（係る）」と判定された形態素は，係り先の形態素の下に移動し，形態素列から削除されていく．最終的に全ての形態素が別の形態素に係り，形態素列がひとつだけになれば解析が終了する．

判定は，以下の三パターンのいずれかになる．

- Left：右の形態素が左の形態素に係る
- Right：左の形態素が右の形態素に係る
- Shift：二つの形態素間に依存関係がないので，何もせず次のペアへ移動する

このアルゴリズムの擬似コードを図 27 に示す．

この判定をおこなうパターン識別器の SVM は，形態素とその品詞を手がかりとして依存構造を学習する．学習には，既に依存構造が解析されたデータを用意し，上記のアルゴリズムをシミュレートすることで学習用の教師データを生成する．SVM の機械学習アルゴリズムは後述することにして，ここでは SVM がパターン識別のための特徴ベクトルとして用いる特徴量を説明する．このアルゴリズムでは形態素とその品詞のみを既知とし，その相対的位置関係から依存構造を学習する．あるふたつの隣り合う形態素間の依存関係を判定する際，特徴量としては，自分たちを含めて左右に任意の文脈長の形態素を調べ，その形態素，品詞，子ノードの形態素，品詞を用いる．（図 28 参照）

山田らは Penn Treebank の構成素木を主辞規則を用いて依存構造木に変換し，これを教師データとして依存構造解析の SVM を学習させた[*25]．文脈長など各種のパラメータでさまざまな実験をおこない，最終的に

- 個々の形態素の係り先精度（Dependency Accuracy） 90.6%
- 文全体の係り先精度（Complete Rate） 40.0%

を達成している[*26]．この精度は従来の句構造に基づく構文解析器のうちもっとも優れたものと比較して若干劣っているが，句構造に関する文法的知識を一切用いていない点を考慮すると非常に優れた結果を示していると山田らは主張している．

[*25] Penn Treebank の section 02 から section 21 の約 40000 文を学習データとして用い，section 23 の 2416 文でテストをおこなっている．

[*26] 操作を 6 種類にまで増やしたモデルによる数値．カンマやピリオドなどの記号については評価対象から除外している．

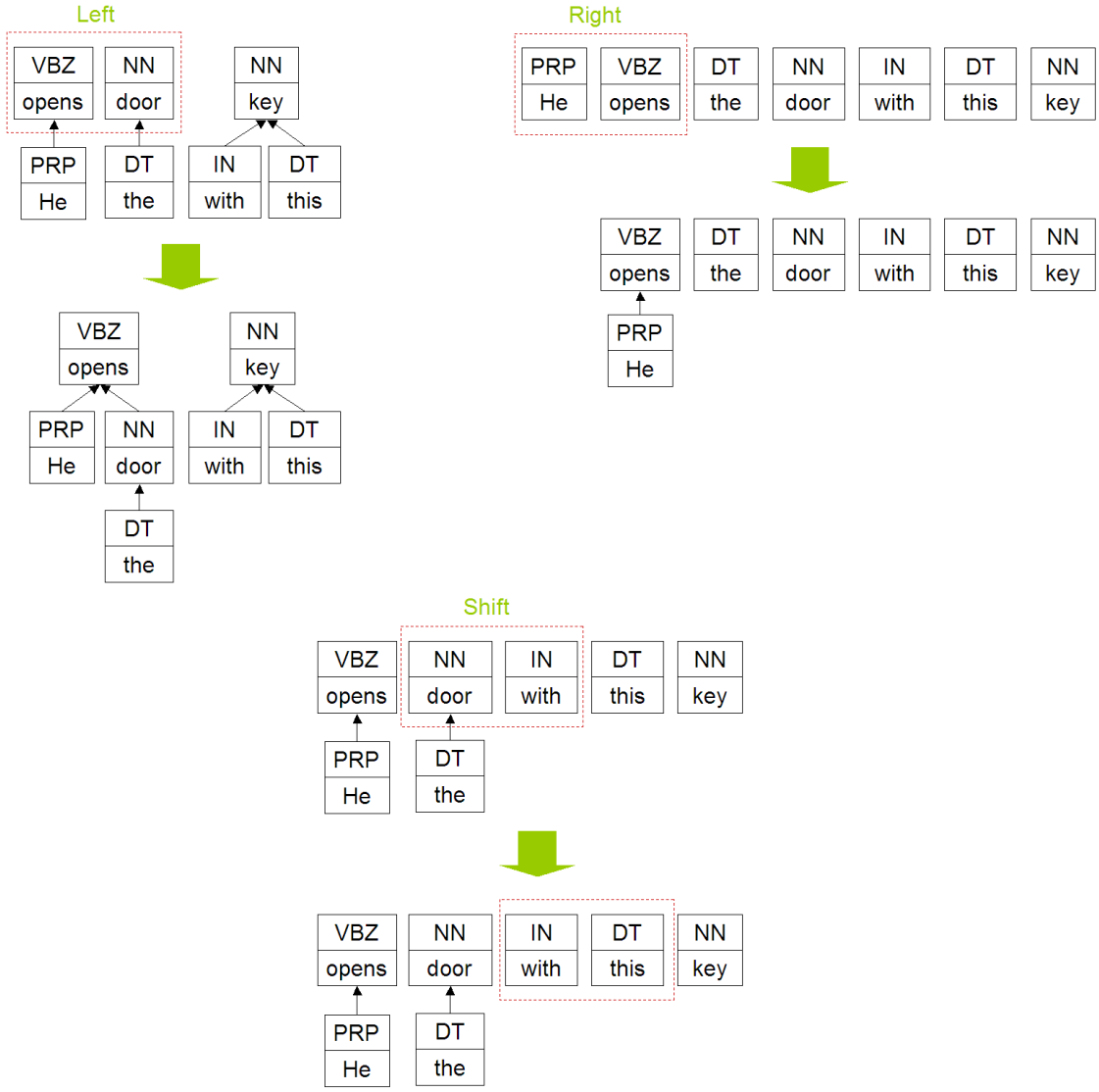


図 26　三種類の操作による依存構造解析

Input Sentence: $c_1, c_2, \ldots, c_i, \ldots, c_n$
Initialize:
  $i = 1; s = c_1, c_2, \ldots, c_i, \ldots, c_n$
  no_construction = *true*
Start: while *true* do begin
  if $i == |s|$ then
    if no_construction == *true* then break;
    no_construction = *ture*
    $i = 1$;
  else
    x = *get_contextual_features*(*s*, *i*);
    y = *classify*(*model*, x);
    if *y* == **Left** or **Right** then
      $s = construction(s, i, y)$;
      no_construction = *false*;
    else if *y* == **Shift** then
      $i = i + 1$;
    end;
  end;
end;

図 27　山田 [17][18][19] の依存構造解析アルゴリズム：
get_contextual_features(*s*, *i*) は配列 s 中の i 番目の形態素について周囲の特徴量を取得する関数
classify(*model*, x) は特徴量 x を SVM モデル *model* で処理した判定結果を返す関数
construction(*s*, *i*, *y*) は配列 s 中の i 番目の形態素に対して操作 y $\in \{Left, Right, Shift\}$ を施す関数

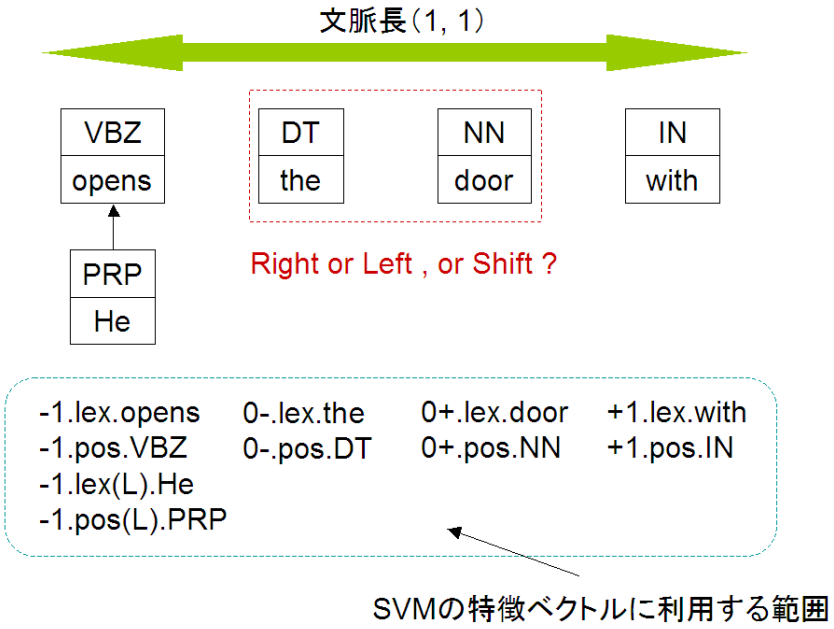


図 28　依存構造の学習・推定に用いられる特徴量の範囲
それぞれの形態素についての形態素そのもの（lex）と品詞（pos）を特徴量として利用する．他の形態素に依存され（係られ）ている場合は，その形態素と品詞も特徴量となる．（L）は左から係った形態素，（R）は右から係った形態素であることを示す．推定の実行時は，それまでの推定で係ると判断された形態素を動的に利用することになる．

4　機械学習アルゴリズム

4.1　Support Vector Machine

Support Vector Machine (SVM) はマージン最大化の基準に基づく線形学習機械であり，高次元ベクトルの二値分類器として用いられる．教師付き学習を凸二次計画問題に定式化することで局所最適に陥ることなく標本数に応じた統計学習をすることができ，高い汎化能力を比較的容易に得ることができることから，近年，確率的モデルと並んで多くの自然言語処理タスクで用いられている．本論文で取り扱う文の統語的解析及び意味解析では，単語やその品詞，それらの相対的な位置関係などの組み合わせが機械学習における特徴量となることから，学習データが非常に高次元のベクトルとなる．そこで，高い汎化能力を持つ SVM を学習に用いることとする．

SVM の学習は，n 次元ユークリッド空間上に分布する l 個の正・負を表す訓練事例 $(\mathbf{x}_i, y_i), (i \leq l, \mathbf{x}_i \in \mathbf{R}^n, y_i \in \{+1, -1\})$ を与えられたとき，この訓練事例を正例・負例に正しく分離する超平面（hyper plane）$f(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} + b)$ $(\mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R})$ を求めるアルゴリズムである．

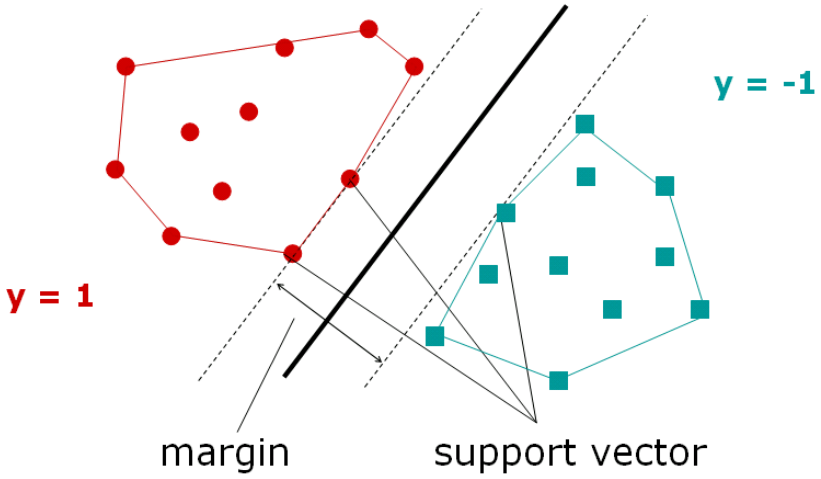


図 29　Support Vector Machine

超平面は，最も距離の近い正例・負例との距離（マージン）を最大化するように求められる．マージンは，

$$d = \min_{x_i y_i = +1} \frac{|\ \mathbf{w} \cdot \mathbf{x}_i\ |}{\|\ \mathbf{w}\ \|} + \min_{w_i y_i = -1} \frac{|\ \mathbf{w} \cdot \mathbf{x}_i\ |}{\|\ \mathbf{w}\ \|} = \frac{2}{\|\ \mathbf{w}\ \|} \tag{1}$$

で表されるため，マージン最大化超平面は以下の最適化問題の解となる．

$$\texttt{Minimize} : L(\mathbf{w}) = \|\ \mathbf{w}\ \|^2 \tag{2}$$

$$\texttt{Subject to} : y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \tag{3}$$

この最適化問題は，ラグランジュ定数αを導入し双対問題に変換することで以下のようになる．

$$\texttt{Maximize} : L(\alpha) = \sum_{i=1}^{i} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{i} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{4}$$

$$\texttt{Subject to} : \alpha_i \geq 0, \quad \sum_{i=1}^{i} \alpha_i y_i = 0 \tag{5}$$

また，最終的な分離関数は以下のようになる．

$$f(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} + b) = sgn \left\{ \sum_{i=1}^{i} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \right\} \tag{6}$$

このとき，超平面に最も近い正例・負例をサポートベクトルと呼ぶ．SVM は，正例・負例それぞれの凸包に対して，分離境界に最も近いサポートベクトルのみを考慮して学習することから，過学習が起こりにくいという特長を持っている．

上で述べた SVM の基本的なアルゴリズムは，正例・負例の 2 クラスのデータを線形に分離することが可能であることを前提としている．しかし，実際の学習問題では線形分離可能の前提は成り立ちにくく，より一般的な非線形の分類問題に対応する必要がある．そこで SVM では，正例・負例の特徴ベクトルを非線形変換し，高次元の特徴空間上で線形分離を行う．一般に，線形分離問題は事例数が多いほど難しく，逆に特徴空間の次元数が多いほどやさしくなる．高次元空間に事例を写像することで，低次元空間では線形分離できない問題を線形分離可能にしようというものである（図 30）．

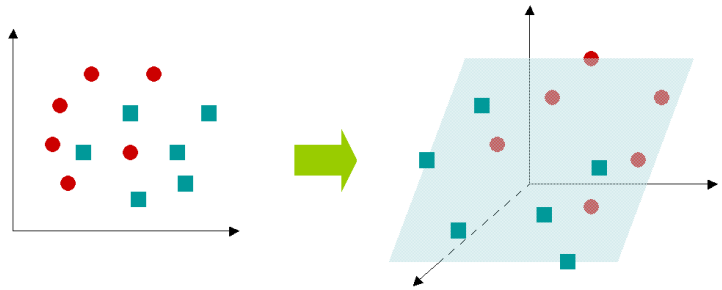


図 30　高次元空間への射影

特徴ベクトル $\mathbf{x}$ の空間を高次元に射影する関数を $\phi(\mathbf{x})$ とすると，式 (4) は

$$\texttt{Maximize} : L(\alpha) = \sum_{i=1}^{i} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{i} \alpha_i \alpha_j y_i y_j (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)) \tag{7}$$

となる．ここで，$\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)$ は共に高次元のベクトルであり，この最大化問題の関数 $L(\alpha)$ における内積の計算量が増大してしまっていることが分かる．SVM では，**カーネルトリック**と呼ばれる手法を用いてこの計算量の増大を抑えている．これは，関数 $\phi()$ で射影された高次元空間における内積を，実際に射影を行うことなく求めることができるカーネル関数（式 (8)）を利用する方法である．

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{8}$$

いくつかの代表的なカーネル関数が考案されているが，今回は最も一般的な関数のひとつである多項式カーネル（式 (9)）を利用する．

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d \tag{9}$$

例：

$$d = 2, \mathbf{x}_i = (a_1, a_2) \in \mathbf{R}^2, \mathbf{x}_j = (b_1, b_2) \in \mathbf{R}^2$$

の場合，

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2 \\ &= (a_1 b_1 + a_2 b_2 + 1)^2 \\ &= a_1^2 b_1^2 + a_2^2 b_2^2 + 2a_1 b_1 + 2a_2 b_2 + 2a_1 a_2 b_1 b_2 + 1 \\ &= (a_1^2, a_2^2, \sqrt{2}a_1, \sqrt{2}a_2, \sqrt{2}a_1 a_2, 1) \cdot (b_1^2, b_2^2, \sqrt{2}b_1, \sqrt{2}b_2, \sqrt{2}b_1 b_2, 1) \end{aligned}$$

となる．すなわち，
$\phi : (z_1, z_2) \mapsto (z_1^2, z_2^2, \sqrt{2}z_1, \sqrt{2}z_2, \sqrt{2}z_1 z_2, 1)$ なる関数によって 2 次元空間を 6 次元空間に射影した状態での内積を，陰に計算することができる．

4.2　Support Vector Machine による多クラス分類

上記で説明した SVM は，二値分類アルゴリズムである．自然言語処理における複雑な分類タスクに SVM を用いるにあたっては，これを多値分類に拡張する必要がある．二値分類器を多値分類に拡張するにあたっては，代表的な方法としてふたつのやり方がある（図 31）．

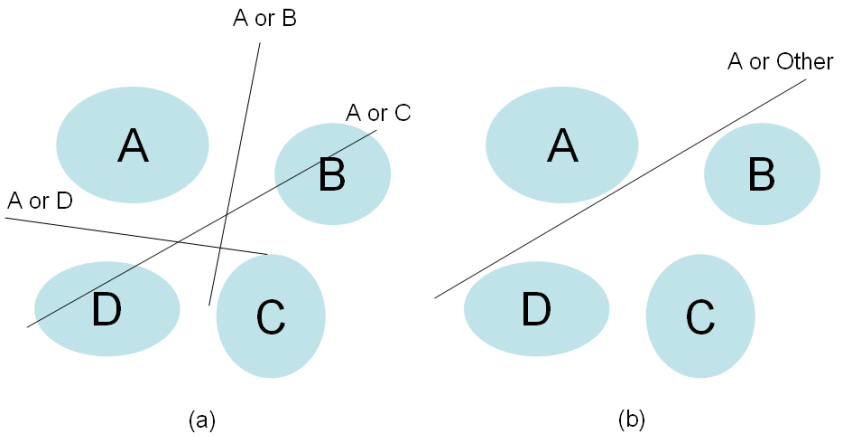


図 31　SVM による多クラス分類：(a) Pair-wise 法 (b) One vs All 法

Pair-wise 法

事例を n 個のクラスに分類するタスクの場合，それらの組み合わせパターンに対応する数の SVM を学習し，それら全てで分類を行って最も多く分類されたクラスを分類結果として出力する．

One vs All 法

事例を n 個のクラスに分類するタスクの場合，あるクラスの事例とそれ以外の全てのクラスの事例を分類する SVM を n 個学習し，それら全てで分類を行って最も多く分類されたクラスを分類結果として出力する．

Pair-wise 法はクラスラベル数の組み合わせ数分の SVM を学習する必要があり，一方 One vs All 法はクラスラベルの数だけ SVM を学習すればよい．しかし，学習された SVM 群全体での分類精度やひとつひとつの SVM の学習所要時間は問題によってさまざまであり，その都度実験によって優れた方法を採用する必要がある．今回は，何度かおこなった予備実験で優位性の見られた One vs All 法を主に用いた．

4.3　ソフトウェア

本研究では，前節で述べた SVM の実装ソフトウェアとして，TinySVM[*27] 及び YamCha[*28] を利用する．TinySVM はベクトルデータからの SVM 学習・分類プログラムであり，純粋な SVM 本体に該当する．一方 YamCha は，テキストデータを加工して TinySVM 用のベクトルデータに変換し，TinySVM を利用してテキストデータの学習・分類を行うプログラムである．

テキストデータの機械学習においては，前述したように単語や品詞，文法的構造などが特徴量として用いられる．しかし，計算機にとってテキストデータは単なる文字コードの羅列に過ぎない．このため，計算機で扱えるのは単語や品詞を一意に特定する “ID” である．YamCha では，データとして与えられる単語数分だけの次元を持つベクトルを用意し，ある単語 ID を特定のベクトル要素に対応させ，その要素を 1，それ以外の全ての要素を 0 とすることで単語を一意に特定する[*29]（図 32）．実際には単語だけではなく品詞やその他の

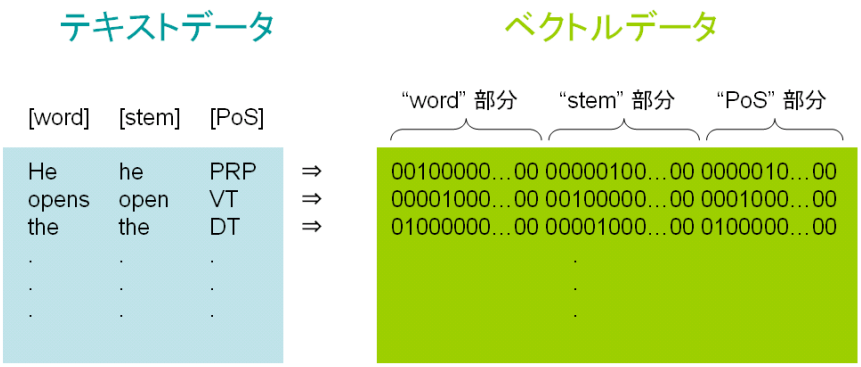


図 32　テキストデータの 0-1 ベクトルデータへの変換

情報もテキストデータとして与えられるため，これらの組み合わせから生成される特徴ベクトルは，万を超えるほどの超高次元ベクトルとなる．YamCha は，テキストデータをこうした高次元ベクトルデータに変換し，TinySVM で学習を行い，この学習済み SVM モデルを用いて未知データの分類を実行することができる．

[*27] http://chasen.org/ taku/software/TinySVM/

[*28] http://chasen.org/ taku/software/yamcha/

[*29] 訓練データ中に存在しない未知語に対しては，十分な量の空きベクトルを余分に確保しておくことによって対処していると思われる．

YamCha は元々は文データのチャンキング（句などの固まりを発見するタスク）を目的として作成されたため，形態素列に対するクラスラベル推定を基本用途としているが，テキストデータから効率良く一意のベクトルデータを生成できることから，自然言語処理の他のタスクでも広く用いられるようになっている．本研究では，後述する依存構造の解析及び意味タグの推定において，この TinySVM + YamCha をツールとして用いる．

5　実装と評価 1 ～依存構造を利用した意味解析～

提案する依存構造に基づいた意味解析システムを実装し，コーパスによるテストで精度の評価をおこなう．ここでは，依存構造を特徴量として意味解析に利用するシステムを実装する．

5.1　実装

ここで実装する意味解析モデルは，依存構造解析済みの文からすべての依存関係を抜き出して列挙し，それぞれの依存関係上にどんな意味関係があるかを推定するというものである．

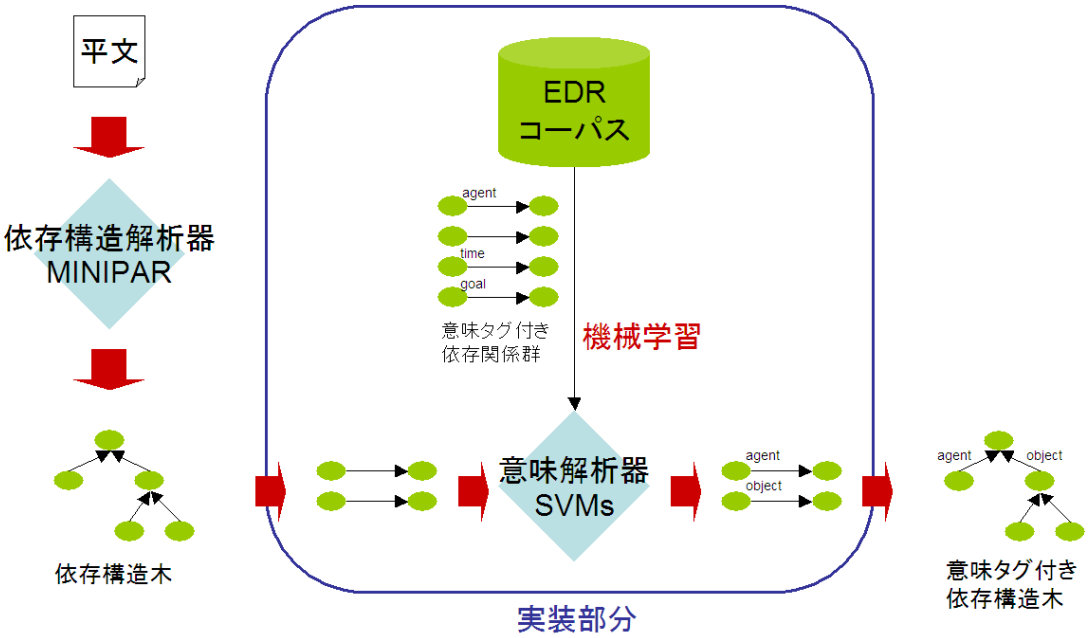


図 33　実装 1：システム概要

EDR コーパスの依存構造解析済みの文データは，計算機で扱う際は以下のようにタブで区切られたテーブル型のテキストデータで表現する．もともとは形態素列からのチャンキングなどのタスクで採用されていたデータ表現スタイルであり，一行にひとつの形態素の情報を記述している．

[Index]	[形態素]	[品詞]	[係り先 Index]	[係り先に対する意味タグ]
1	He	PRON	2	agent
2	gave	VT	*	_
3	me	PRON	2	goal
4	a	ART	5	_
5	chocolate	NOUN	2	object
6	.	PUNC	2	_

これを整理しなおし，一行にひとつの依存関係を記述すると，以下のようになる．

[Index]	[形態素]	[品詞]	[係り先 Index]	[係り先形態素]	[係り先品詞]	[係り先に対する意味タグ]
1	He	PRON	2	gave	VT	agent
2	gave	VT	*	-	-	_
3	me	PRON	2	gave	VT	goal
4	a	ART	5	chocolate	NOUN	_
5	chocolate	NOUN	2	gave	VT	object
6	.	PUNC	2	gave	VT	_

このひとつひとつの依存関係について，係り元の単語が持つ意味役割（深層格）タグを SVM によって学習し，推定するシステムを実装する．自動解析の際の依存構造解析については，現在もっとも高い精度での解析が可能な MINIPAR を利用することを想定する（図 33）．

5.2　予備実験

依存構造を特徴量として利用することの効果を測定するため，基本的な特徴量のみを使ったベースラインモデルを作成し，意味解析の精度を測定した．

深層格推定のための特徴量としては，形態素とその原形と品詞のみを用いた．これは，構文的解析のファーストステップである形態素解析の結果のみを利用する場合を想定している．英語においては，日本語の助詞のようにかなり明確に深層格を反映する表層格は存在せず，主語（名詞）・述語（動詞）・目的語（名詞）のような品詞の並びが表層格として得られる．そのため，ある単一の形態素のタグを推定する際にも前後の形態素の情報を利用することで精度が向上することがチャンキングなどのタスクにおいて示されている．そこで，形態素の列をはじめから順にスキャンし，自分を含めて前後二つの形態素と原形，品詞を SVM の特徴ベクトルとして深層格タグを学習・推定するモデルを作成した（図 34）[*30] ．この意味解析では，述語となる語が何であ

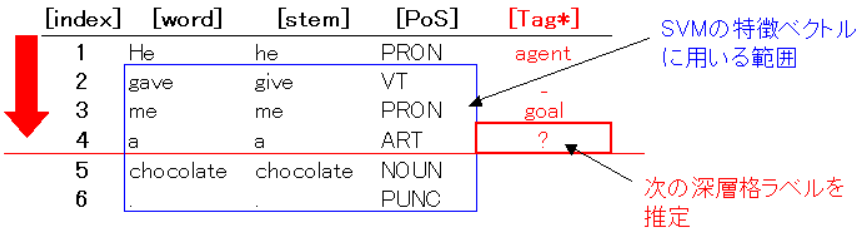


図 34　ベースラインモデルの学習に用いる特徴量

るかをまったく指定しておらず，また依存関係にある語も与えられていない．つまり，ここでおこなっている意味解析では，「何らかの語に対して XXX のような意味的役割を持っている」ということを学習・推定させていることに過ぎない．

[*30] テキストデータからベクトルデータへの変換，SVM の学習・推定に前述の TinySVM+YamCha を利用した．

EDR コーパスからランダムに抽出した 10000 文に対して，9 割を学習に，残り 1 割をテストに用いて評価したところ，表 3 のようなタグ付け精度が測定された．

表 3　予備実験 結果（ベースラインモデル）

TagName	Precision	Recall	$F_{\beta=1}$
agent	69.04%	71.96%	70.47
goal	45.45%	31.39%	37.14
implement	20.00%	9.33%	12.73
manner	48.71%	50.37%	49.52
object	53.44%	60.08%	56.56
place	51.12%	42.24%	46.26
possessor	36.23%	24.04%	28.90
purpose	34.55%	31.15%	32.76
source	71.43%	47.62%	57.14
time	64.98%	61.04%	62.95
AVERAGE(weighted)	55.12%	54.96%	55.04

結果を見ると，想像以上に高い精度が観測されている．agent は動作主格であり，文の主語になることが非常に多いことから，品詞の相対的位置関係からのみでも「agent らしさ」が比較的容易に得られることが分かる．source なども，コーパス内に “from 〜” のような文例が多ければ，周りの形態素だけからでも十分に推定できるのではないかと思われる．一方，goal などは “to 〜” の形が多いと予測できるが，英語では “to” に不定詞など他の用法が多いこと，また goal は文の主語にくることもあることなどから推定が難しいのだと考えられる．とはいえ，形態素・原形・品詞しか用いていないにもかかわらずこれだけの予測精度が記録できるということは，意味解析問題で定義されている意味役割（ここでは深層格）が単語自体の “ID” と，その分類である品詞の位置関係でかなりの程度決まっているということを示唆している．逆に言えば，単語ごとの意味タグを，他の語との関係を無視して推定するというタスクは，比較的単純な問題であるといえる．

そこで次に，「係り先の語が何であるか」の情報を与えた場合の実験をおこなった．EDR コーパスの依存構造情報を抜き出し，これを既知のものとして各形態素の特徴量に与えた．具体的には依存先の形態素・原形・品詞を特徴量として与えたが，「周辺の語の依存先の語」はあまり意味を成さないと考えられるため，依存先の特徴量は推定対象の形態素のもののみとした．これを「依存構造利用モデル」と呼ぶ．

ベースラインモデルと同様のデータで実験をおこなったところ，表 4 のような結果となった．訓練・テストに用いているコーパスが 10000 文とあまり大きくないため，出現頻度の低いタグ（implement, possessor, source など）に関してはテストセットによって精度の分散が大きく，あまり信頼性のある精度を出すことができないが，全体的に見て，形態素の情報しか用いないベースラインモデルよりも精度が向上していることが分かる．

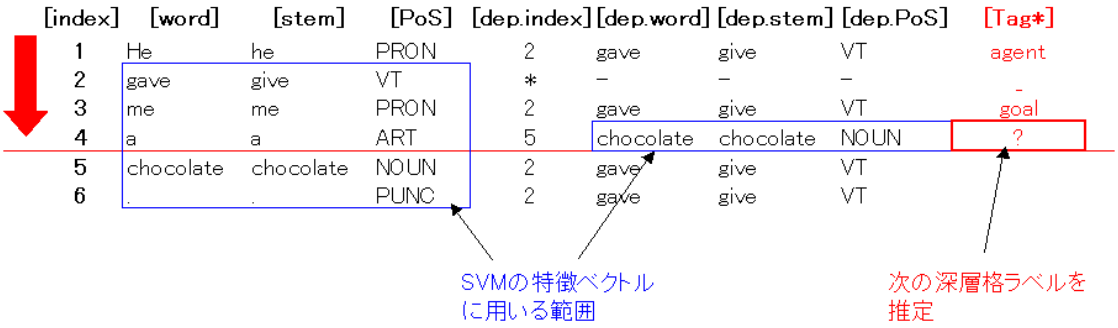


図 35　依存構造利用モデルの学習に用いる特徴量

表 4　予備実験 結果（依存構造利用モデル）

TagName	Precision	Recall	$F_{\beta=1}$
agent	72.02%	75.78%	73.85
goal	52.87%	40.89%	46.11
implement	39.39%	15.12%	21.85
manner	59.06%	62.81%	60.88
object	66.57%	74.59%	70.36
place	54.24%	37.65%	44.44
possessor	37.14%	25.49%	30.23
purpose	65.12%	43.75%	52.34
source	53.13%	34.00%	41.46
time	66.09%	63.19%	64.05
AVERAGE(weighted)	64.22%	63.88%	64.05

5.3　評価実験

依存構造を利用してどの程度まで高い精度で意味解析ができるかを確認するために，EDR コーパスに記述された人手でタグ付けされた依存構造木を教師データとして意味解析モデルを学習し，評価をおこなう．ここで得られる精度は，依存構造解析も含めて自動でおこなった際の理論的最高値となる[*31] ．ここで作成する意味解析モデルを，「提案モデル 1」と呼ぶことにする．

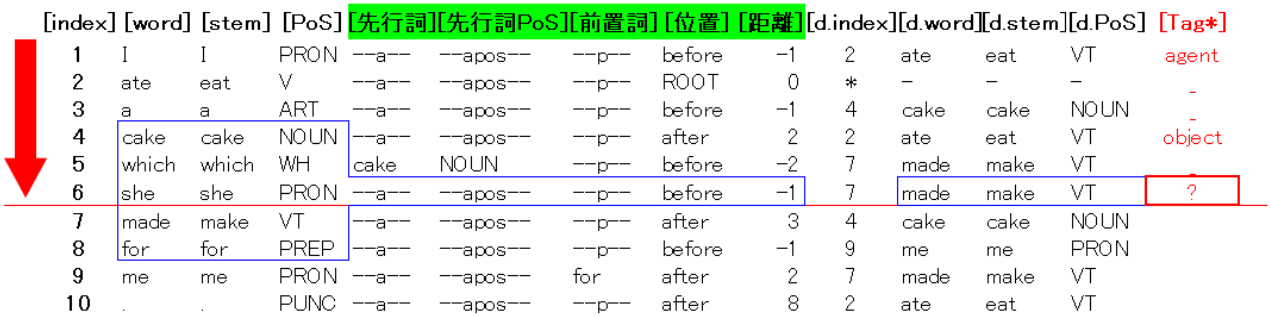
予備実験の依存構造利用モデルの結果を参考にし，いくつかの特徴量を追加した．

1. 先行詞（その品詞）
   その関係詞の先行詞の形態素と品詞（関係詞の場合のみ，依存構造木からルールで抽出）

[*31] 実際には，EDR コーパスの依存構造の記述は誤りも多く，より解析精度を高めるためにはタグ付け基準を逸脱した不良データを修正・削除する必要がある．

2. 前置詞
   その形態素についている前置詞（依存構造木からルールで抽出）
3. Position
   係り先の語との形態素列上での相対的位置関係（before or after）
4. Distance
   係り先の語との形態素列上での距離

EDR コーパスの依存構造は通常とは若干異なった基準で記述されており，特に前置詞が前置詞句の名詞に係るルールになっており，前置詞と名詞が離れていると名詞がその前置詞を認識できなくなってしまうため，名詞側の形態素に，係っている前置詞そのものを加えることにした．



[index]	[word]	[stem]	[PoS]	[先行詞]	[先行詞PoS]	[前置詞]	[位置]	[距離]	[d.index]	[d.word]	[d.stem]	[d.PoS]	[Tag*]
1	I	I	PRON	--a--	--apos--	--p--	before	-1	2	ate	eat	VT	agent
2	ate	eat	V	--a--	--apos--	--p--	ROOT	0	*	-	-	-	-
3	a	a	ART	--a--	--apos--	--p--	before	-1	4	cake	cake	NOUN	-
4	cake	cake	NOUN	--a--	--apos--	--p--	after	2	2	ate	eat	VT	object
5	which	which	WH	cake	NOUN	--p--	before	-2	7	made	make	VT	-
6	she	she	PRON	--a--	--apos--	--p--	before	-1	7	made	make	VT	?
7	made	make	VT	--a--	--apos--	--p--	after	3	4	cake	cake	NOUN	
8	for	for	PREP	--a--	--apos--	--p--	before	-1	9	me	me	PRON	
9	me	me	PRON	--a--	--apos--	for	after	2	7	made	make	VT	
10	.	.	PUNC	--a--	--apos--	--p--	after	8	2	ate	eat	VT	

図 36　提案モデル 1 の学習に用いる特徴量
色付き部分が予備実験の「依存構造利用モデル」からの追加部分

今回作成するモデルは，意味解析システムの意味解析部にそのまま採用することを意識し，モデルの学習に用いるデータの量を増やした．また，いくつかの有用と思われる意味タグを解析対象に追加した．EDR コーパスからランダムに抽出した約 30000 文に対して，9 割を学習に，残り 1 割をテストに用いて評価したところ，表 5 のようなタグ付け精度精度が測定された．

agent, object という最も事例数の多いタグでは $F_{\beta=1}$ 値が 70 を越え，goal, source, time, place など代表的な深層格として知られる主要なタグに関しても 55 ～ 65 強といったそこそこの精度で解析ができている．implement, possessor, scene などの意味タグに対する推定精度の低さが目立つが，これらのタグは学習・テストデータ中に事例数が極端に少なく，学習が十分にできていないことや十分な数のテスト事例が得られていないことなどが原因として考えられる．特に possessor に関しては，所有者を表す記述があるにもかかわらずタグ付けされていない文例がコーパス中にしばしば見られたため，本来は正例になるはずのデータが負例として学習されていることによる性能低下も起きていると考えられる．

ベースラインモデルと提案モデル 1 の結果を比較（図 37）してみると，依存構造を特徴量として利用することで意味解析の精度をかなり向上できていることが分かる．しかし，もともと解析精度が低い意味役割タグの解析結果が極端に良くなるようなことはない．それぞれの意味役割タグの推定でどんな誤判定がおきているのかを調べるため，テストセットへの解析結果のうちタグ付けを誤った箇所を調べてみると，本来 agent, source, goal, implement などの意味役割を持つはずの語に対しては object を振ってしまう誤りが大半であっ

表 5　評価実験 結果（提案モデル 1）

TagName	Precision	Recall	$F_{\beta=1}$
agent	75.50%	79.18%	77.30
goal	56.91%	49.74%	53.08
implement	29.86%	19.37%	23.50
manner	60.04%	59.48%	59.76
number	72.67%	81.02%	76.62
object	70.39%	75.53%	72.87
place	56.09%	56.09%	56.09
possessor	42.24%	30.15%	35.19
purpose	57.72%	48.04%	52.44
quantity	35.71%	26.60%	30.49
scene	36.96%	31.41%	33.96
source	59.65%	59.65%	59.65
time	67.78%	67.33%	67.55
time-from	75.61%	63.27%	68.89
time-to	52.17%	42.86%	47.06
unit	56.00%	56.00%	56.00
AVERAGE(weighted)	65.16%	64.99%	65.07

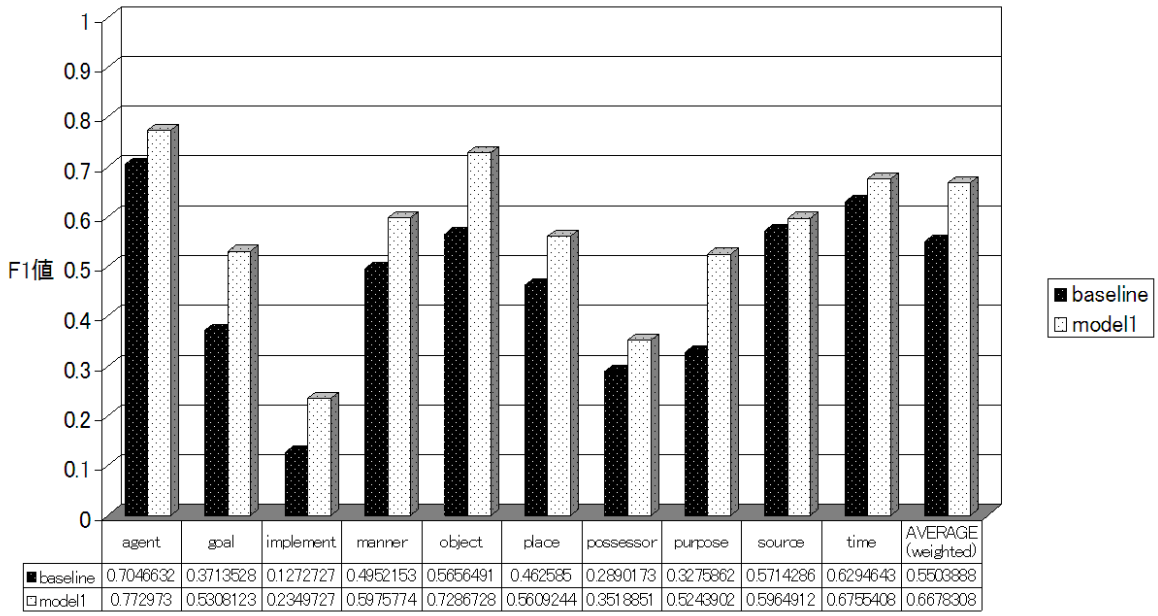
た．これは，EDR コーパス中で定義されている object の範囲が単なる動作の対象物に限らずより広い範囲で用いられ[*32]，結果としてタグ付けの数がもっとも多くなっているためである．そのため，agent なら動作主体性，implement なら道具性がはっきりと表現されない文脈など判断の難しい事例では，同じ名詞で最もケースの多い object を振ってしまうように学習がおこなわれてしまっていると思われる．

また，time, place, possessor などに対しては modifier タグを振ってしまう誤りが最も多く，time タグへのタグ付け誤りの約 30%，place タグの約 50%, possessor タグに至っては 90% 以上がこの誤りに該当していた．modifier は修飾関係にあることを示し，形容詞などに付くことの多いタグであるが，そもそも依存関係にあるはずの語の間に modifier が付くか付かないかの基準があいまいであり，EDR コーパスの中でも特に記述の一貫性の低い状態になっている．このタグの付いた語が，本来意味的にふさわしいタグに対する誤った負例となり，学習モデル全体の精度に悪影響を与えてしまっている可能性は否定できない．

5.4　既存研究との比較

ここでは，作成した 提案モデル 1 の意味解析の性能を既存の意味解析研究において考案されたシステムと比較してみる．ただし，本論文が提案する依存構造を基にした意味解析タスクは問題の定義が既存の意味解析

[*32] 例えば “A is B” の文においては，「A は B の object である」とタグ付けする規則になっている．



	agent	goal	implement	manner	object	place	possessor	purpose	source	time	AVERAGE (weighted)
■baseline	0.7046632	0.3713528	0.1272727	0.4952153	0.5656491	0.462585	0.289073	0.3275862	0.5714286	0.6294643	0.5503888
□model1	0.772973	0.5308123	0.2349727	0.5975774	0.7286728	0.5609244	0.3518851	0.5243902	0.5964912	0.6755408	0.6678308

図 37　ベースラインモデルと提案モデル 1 の比較：タグごとの推定精度

タスクと若干異なっており，EDR 英語コーパスで意味解析をおこなった既存研究もないことから，総合的な解析精度で直接の性能比較をすることはできない．

比較対象として，既存研究において紹介した CoNLL Shared Task 2004, 2005 で良い成績を示している Hacioglu らのシステム [11]（for 2004）と Pradhan らのシステム [13]（for 2005）を取り上げる．Hacioglu らのシステムは，既存研究で紹介したように，句構造の情報をベースとし，意味タグ付けをチャンキングと同様の方法でおこなっている．また形態素ごとではなく，句のチャンクごとに意味タグを推定するほうが精度が向上することを示している．

一方 Pradhan らのシステムは，前年の Hacioglu らのシステムをもとにしたものである．2005 年の Shared Task で利用可能になった構成素木をフルに使い，G&J で提案された手法（p.18）のように構成素木のノードごとに意味タグの推定をおこなったのち，ここで推定した意味タグを仮出力としてチャンクごとに追加特徴量として加え，最後に Hacioglu らのシステムと同様，チャンキングによって意味タグの最終推定をおこなうという二段解析をおこなっている．「構成素木のノード」と「句チャンク」という異なる構文的特徴を組み合わせて利用することで意味タグの推定精度が向上したとしている．これらの意味解析システムの PropBank 上でのテストによる精度を示す（表 6）[*33] ．

提案モデルはそもそも異なるコーパスの異なる意味役割タグを推定しているため，推定精度全体の加重平均を比べることにあまり意味はない．そこで，提案モデル 1 の推定精度（表 5）の中から似た性質の意味役割タグを選び出して比較してみる．

[*33] Hacioglu 2004：8936 文で学習，1671 文でテスト
Pradhan 2005：39832 文で学習，2416 文でテスト

表 6　評価実験 比較対象（CoNLL Shared Task システム）

	Hacioglu 2004			Pradhan 2005		
TagName	Precision	Recall	$F_{\beta=1}$	Precision	Recall	$F_{\beta=1}$
Arg0	82.93%	79.88%	81.37	91.39%	82.23%	86.57
Arg1	71.92%	71.33%	71.63	79.80%	76.23%	77.97
Arg2	49.37%	49.30%	49.33	68.61%	62.61%	65.47
Arg3	57.50%	46.00%	51.11	73.95%	50.80%	60.27
Arg4	87.10%	54.00%	66.67	78.65%	68.63%	73.30
Arg5	0.00%	0.00%	0.00	75.00%	60.00%	66.67
ArgM-ADV	53.36%	38.76%	44.91	61.64%	46.05%	52.71
ArgM-CAU	57.89%	22.45%	32.35	76.19%	43.84%	55.65
ArgM-DIR	37.84%	28.00%	32.18	53.33%	37.65%	44.14
ArgM-DIS	66.83%	62.44%	64.56	80.56%	63.44%	70.98
ArgM-EXT	70.00%	50.00%	58.33	100.00%	46.88%	63.83
ArgM-LOC	46.63%	36.40%	40.89	64.48%	51.52%	57.27
ArgM-MNR	50.31%	31.76%	38.94	62.90%	45.35%	52.70
ArgM-MOD	98.12%	92.88%	95.43	98.64%	92.38%	95.41
ArgM-NEG	91.11%	96.85%	93.89	98.21%	95.65%	96.92
ArgM-PNC	52.00%	15.29%	23.64	56.67%	44.35%	49.76
ArgM-PRD	0.00%	0.00%	0.00	0.00%	0.00%	0.00
ArgM-TMP	64.57%	50.74%	56.82	83.37%	71.94%	77.23
AVERAGE	72.43%	66.77%	69.49	81.97%	73.27%	77.37

- Arg0，Arg1 はそれぞれ EDR コーパスの agent, object に該当する意味タグである．提案モデル 1 の推定結果は句チャンクまでしか用いない Hacioglu 2004 のシステムと比べても若干劣っているが，PropBank の Arg0, Arg1 がほとんどの場合述語動詞に対する主語，目的語であるのに対し，EDR の agent, object はより意味構造に忠実であり，動作を表す名詞などの意味上の主語，目的語などに対してもタグ付けされていることを考えると，77.30（agent），72.87（object）という提案モデル 1 の $F_{\beta=1}$ 値はかなり良い推定精度が出ていると捉えることができる．Arg2〜Arg5 は述語動詞ごとに該当する意味役割がかなり異なり，直接的な比較はできない．ただし，Arg2 以降のタグの総数は Arg0, Arg1 と比べて極端に少なくなる．
- ArgM-DIR は Directionals（方向付け）であり，EDR での source と goal をまとめたものに該当する．これらの推定精度は PropBank のシステムでは解析精度が極端に低く，提案システム 1 の方がかなり良い精度で解析できている[*34]．
- ArgM-EXT は Extent Markers であり，変化の程度を表す語に付く．EDR でこれに該当しそうなも

[*34] 述語動詞によっては source や goal にあたるものが Arg2〜arg5 に割り当てられることがあるので，一概には判断できない。

のとしては quantity などが考えられるが，この二つの比較では PropBank のシステムの方が良い精度を示している．

- ArgM-LOC は Locatives（場所格）であり，動作の起きた位置を表す語に付く．EDR での place に該当すると思われるが，これらの比較では提案システム 1 の精度はより高性能な Pradhan 2005 のシステムに近い値を示している．
- ArgM-MNR は Manner Markers であり，動作の様子を表す副詞的な役割であることを示す．提案システム 1 の manner を比較すると，PropBank の両システムよりも良い精度で推定できていることが分かる．
- ArgM-TMP は Temporal Markers であり，動作の起きた時間を表す．提案システム 1 の time の推定精度は Hacioglu 2004 と Pradhan 2005 のシステムの中間程度の数値を示している．
- ArgM-MOD と ArgM-NGE はそれぞれ Modals, Negation であり，非常に高い推定精度を示しているが，これらは単に “will”, “may”, “can” などの助動詞や “not”, “No” などの否定語を示しているだけである．

PropBank の意味解析システムはいずれも固有表現（Named Entity）[*35]という意味構造の推定にかなり影響力のある特徴量を用いており，また，PropBank コーパスにおいて提供されている述語動詞の辞書（VerbNet）内での概念分類や，述語動詞の文中における態（Voice）なども既知のものとして利用している．提案システム 1 の解析精度は，一見すると Hacioglu 2004 と同程度，Pradhan 2005 よりは劣っているようにも見えるが，上記のような多分に意味的な追加情報を利用せず，文中の形態素に関する情報と依存関係という構文情報のみを利用していることを考えると，かなり良好な解析性能を示しているといえる．このことから，句構造のようなトップダウンの構文的特徴ではなく，語間の依存関係というボトムアップ的な特徴を用いても，同程度の精度で意味解析をおこなうことができることが分かる．

5.5 参考実験（日本語文への適用）

本論文では，依存構造を英文の意味解析に利用することの利点として，同様の方法で日本語の文についても解析がおこなえることを挙げた．EDR では日本語文，英文ともに同じタグ付け形式で記述されており，上記の意味解析モデルを学習するための教師データと同様のものを日本語文についても容易に作成できる．そこで，日本語文についても，上記の実験と同様の意味解析モデルを SVM で学習させ，意味解析の精度を測定してみた．SVM の特徴量に用いたのは，形態素，読み仮名，品詞，Position，Distance，係り先形態素，係り先読み仮名，係り先品詞である．（提案モデル 1 で用いた先行詞や前置詞といった特徴量は英語に特徴的なものであり，ここでは利用できない．）

表 7 を見てみると，個別には推定の難しい意味タグもあるものの，全体的に見てかなり高い精度で意味タグ付けをおこなえることが分かる．全ての依存関係に対する単純な正解率（Accuracy）は 85.58%（null タグも含む）であり，厳密な比較はできないものの既存研究で紹介した深層格推定の各種モデルと同等かそれ以上の精度で解析できていることが期待できる．実際には日本語の意味解析において利用可能なさまざまな文法的知識があり，これを利用することで解析精度をさらに向上することができる可能性がある．

このように，依存関係という比較的言語普遍で原始的な文法構造を利用することで英文だけでなく日本語文

[*35] 特定の場所名を表す LOC，組織名を表す ORG，人名を表す PERSON など

表 7　参考実験 結果（日本語文）

TagName	Precision	Recall	$F_{\beta=1}$
agent	65.78%	60.91%	63.25
goal	62.05%	71.84%	66.59
implement	49.40%	38.75%	43.43
manner	71.54%	68.34%	69.90
object	76.07%	79.27%	77.63
place	51.06%	47.22%	49.07
possessor	22.69%	11.74%	15.47
purpose	52.54%	44.93%	48.44
source	65.05%	69.43%	67.17
time	69.25%	68.03%	68.63
AVERAGE(weighted)	69.88%	70.00%	69.94

においても同様の手法で高精度の意味解析が可能であることが分かる．現在，日本語文についてはかなり高い精度で依存構造解析をおこなうソフトウェアが利用できるため，こうしたソフトウェアを前処理として利用することで自動的な意味解析器を実現することが可能である．

5.6　解析結果の検討

作成した提案モデル 1 の意味解析器を用い，前処理の依存構造解析器として MINIPAR を利用して，実際に平文入力から意味タグ付き依存構造木を出力する意味解析システムを実装した．ここでは，いくつかの例文を入力してシステムの解析結果を検討する．（簡単のため，学習・推定に用いる特徴量の大部分を省略し，形態素と品詞の依存構造のみ表示）

1. 入力：“He gave me a chocolate.”

   出力：

```
1    He          PRON     2    gave        VT      agent
2    gave        VT       *    ()          C       _
3    me          PRON     2    gave        VT      goal
4    a           ART      5    chocolate   NOUN    _
5    chocolate   NOUN     2    gave        VT      object
6    .           PUNC     2    gave        VT      _
```

   非常に短くて単純な文である．もっとも中心的な述語である “gave” のまわりに agent（“He”），object（“chocolate”），goal（“me”）が登場することが正しく解析できている．（“me” には受益者格の意味タグである beneficiary の方がふさわしいと思われるが，この意味タグは EDR コーパス中での事例数が

極端に少なく，統計学習が非常に困難なため，ここではより一般的な深層格である goal のタグをもって正解としている.）

2. 入力：“The municipal government also hosted three international symposiums on environmental protection last year.” （EDR の学習セット外の文例より）

出力：

```
1    The           ART    3    government   NOUN   _
2    municipal     ADJ    3    government   NOUN   modifier
3    government    NOUN   5    hosted       VT     agent
4    also          ADV    5    hosted       VT     manner
5    hosted        VT     *    ()           C      _
6    three         ADJ    8    symposiums   NOUN   number
7    international ADJ    8    symposiums   NOUN   modifier
8    symposiums    NOUN   5    hosted       VT     object
9    on            PREP   11   protection   NOUN   _
10   environmental ADJ    11   protection   NOUN   _
11   protection    NOUN   8    symposiums   NOUN   modifier
12   last          ADJ    11   protection   NOUN   _
13   year          NOUN   12   last         ADJ    _       ( error!  time --> 5 )
14   .             PUNC   5    hosted       VT     _
```

比較的長い文だが，もっとも代表的な述語は動詞の “hosted” であり，その周りに agent, object などが出現する比較的単純な意味構造の文例である．ほぼ正しい意味タグを推定できているが，“last year” が “hosted” の time であることが発見できていない．これは，依存構造解析で “year” があやまって “last” に係ってしまい，“hosted” との関係性が発見されなかったためである．依存構造解析の結果に意味解析のアルゴリズムが依存しすぎるために起きた誤りと言える．

3. 入力：“The company paid five cents a share in April.”（PropBank の文例より）

出力：

```
1      The      ART    2      company  NOUN   _
2      company  NOUN   3      paid     VT     agent
3      paid     VT     *      ()       C      _
4      five     NUM    5      cents    NOUN   number
5      cents    NOUN   3      paid     VT     object
6      a        ART    5      cents    NOUN   _
7      share    NOUN   5      cents    NOUN   unit
8      in       PREP   9      April    NOUN   _
9      April    NOUN   5      cents    NOUN   time   ( error!  ---> 3 )
10     .        PUNC   3      paid     VT     _
```

文の構造自体は単純なものだが，“in April” の係り先を誤っている．意味役割タグの推定では “five cents a share（一株あたり 5 セント）” の表現が若干難しいかと予想したが，単位をあらわす unit タグを正しく推定できている．図 5 の unit タグの解析精度は 56% とあまり高くはないが，EDR コーパスには新聞記事などの文例が多いことから，似たような事例をうまく学習し，汎化できているのではないかと思われる．（ちなみに PropBank ではこの文例に対して Arg0（“The company”），Arg1（“five cents a share”），ArgM-TMP（“in April”）しかタグ付けされていない．）

以上の例は比較的正しく解析ができているが，明らかに動作主体的な主語としての表層格を持つ語を object と誤判定するなどの極端な間違いもまだかなり多い．このような誤りは学習セット中にまったく存在しない単語の周辺で特によく見られることから，依存関係を基本単位とするタグ付けにおいては，依存関係ペアの語そのものが何であるかが推定の精度にかなり大きな影響力を持っていることが予想される．例えば，文例 3 の “in April” は，文例数 10000 以下で学習した試作システムでは time ではなく place とタグ付けされてしまっていた．これは，学習セット中に “April” という単語が登場せず，これが時を表す語であることが一切学習されなかったためである．こうした誤りをなくすためには，できる限り多くの文例を学習に用いることが必要である．

6　実装と評価 2 〜構文解析と意味解析のブートストラップ〜

前章では，文の依存構造を意味解析の際の特徴量として用いるシステムを実装・評価した．依存関係にある語のペアを基本単位とし，形態素列上の前後関係などを特徴量として追加することで意味解析精度の向上が得られることを示した．しかし，意味解析モデルの学習過程においていくつか不合理な点も見受けられた．

例えば，ある形態素が依存先に対して持っている意味役割タグを推定する場合，その形態素の周辺の情報を特徴量に加えることで精度が向上することが分かっているが，前章で示したデータによる学習では，形態素列という字面上の周辺の語しかチェックすることができず，“a” や “the” などの冠詞やカンマなどの記号が多いと周辺の情報が得られにくくなってしまう．また前後二語ずつの固定文脈長というものにもあまり本質的な意味は無いように思われる．

また，ひとつの依存関係ペアを単位として意味タグの推定をおこなっているが，それ以外の依存関係ペアを考慮に入れていないため，依存先の語のまわりにある他の意味構造を一切考慮に入れていないことになり，場合によってはひとつの述語に対してふたつの語が agent タグを持つなど一文一格の原則（p.24）を破ってしまう恐れがある．

これらの点を考慮すると，提案モデル 1 のように解析済みの依存構造を天下り的に意味解析の特徴量に利用することに多少の違和感が感じられる．そもそも語の間の統語的な依存関係は，意味的な依存関係にかなり直接的に関わっており，人間の言語処理過程において「依存構造」という構文の処理と「意味役割構造」という意味の処理が密接に関連している可能性もある．これまでは構文解析→意味解析という風にリニアにおこなっていた処理を相互におこなっていくことで，それぞれの解析の結果を活かしながら精度を向上させられるといったことも考えられる．そこで，構文の処理である依存構造解析と意味の処理である意味役割タグ推定を同時におこなっていく解析方法を考案し，実装してみた．

6.1　実装

ここで実装する意味解析モデルは，品詞付きの形態素列から依存構造と意味タグを交互に解析し，ボトムアップに意味タグ付き依存構造木を作成するというものである（図 38）．

文中の要素についてほとんど何も手がかりの無い状態から依存構造と意味構造を構築していく過程を再現するため，依存構造解析器としては関連研究で紹介した山田らの SVM による依存構造解析アルゴリズム（p.36）をベースとする．このアルゴリズムは入力として品詞付き形態素列のみを想定しており，局所的にひとつずつ依存関係を発見していくボトムアップ的な解析方法であることから，構文と意味の初期的解析過程の再現に向いていると考えた．

今回は，依存構造解析と意味役割タグ推定を同時におこなうシステムとして，もっともシンプルな解析アルゴリズムを実装した．図 39 のように，依存構造解析において Left または Right，つまりいずれかの語が他の語に依存することが決定した際に，その依存関係において係り元の語が係り先の語に対して何らかの意味役割を持つかどうかを推定する．推定の結果，意味役割タグが付与された場合は，係り元の語に意味役割タグがついていることを記録し，依存構造解析の際の特徴量に意味役割タグの種類も加える．つまり，依存構造解析の結果に従って意味解析をおこない，意味解析の結果を後の依存構造解析にも利用する形となる．

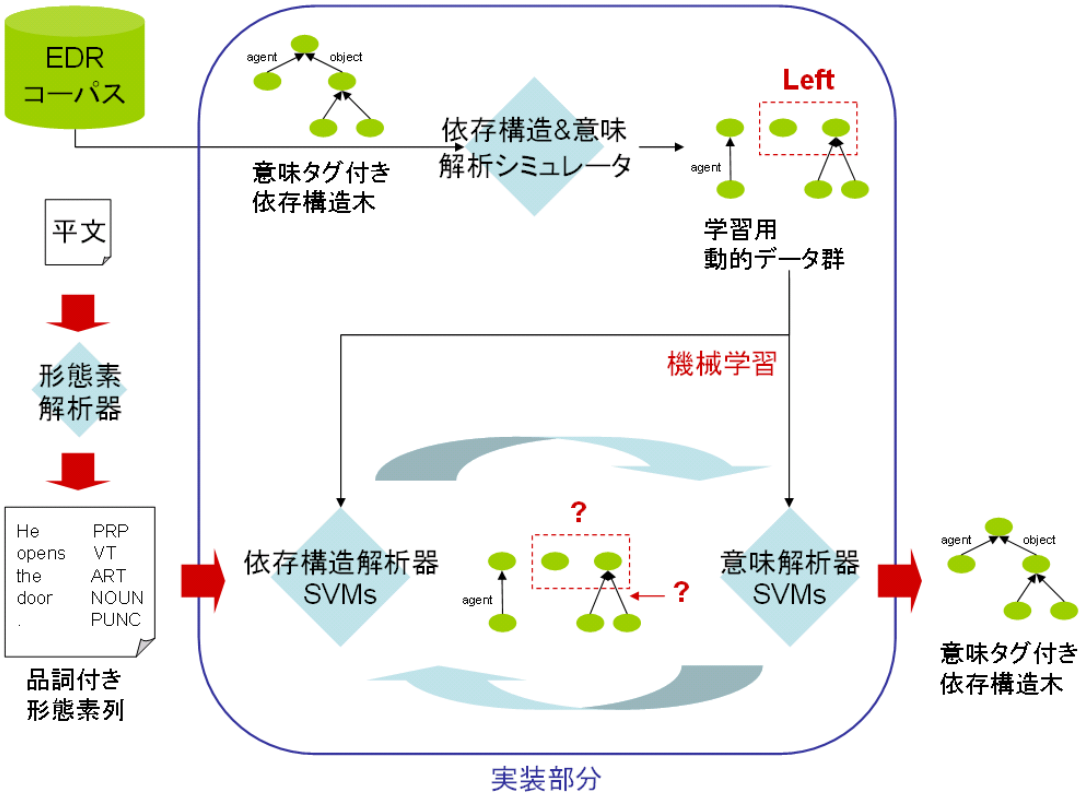
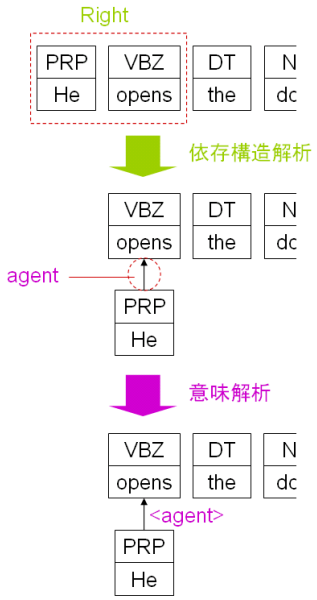


図 38　実装 2：システム概要

利点として考えられるのは，以下の点である．

- 隣接する形態素どうしを順番に見てボトムアップ的に依存関係を発見していくため，本来依存関係にある形態素動詞が配列内で隣接した時点で意味役割関係の有無を推定することができる．すなわち，字面上の周辺情報ではなく，依存関係の部分木というチャンク単位での周辺の情報を見て意味解析をおこなうことができる．
- それまでの意味タグ推定結果を次の解析で動的に利用するため，例えば「係り先になる語に agent の意味役割を持つ語が既に係っているのならば次は他の意味役割だろう」といった一文一格の原則を暗にデータが表現することになる（図 40）．これは，依存関係をバラバラに扱っていた前章の解析モデルでは表現できなかった知識を追加していることになる．
- 意味役割タグが特徴量に追加されていくことで，それまで解析して判明してきた意味構造が後の依存構造の推定にも有利に利用できる可能性がある．（ただし，意味タグ推定の誤りが悪影響を与える恐れも当然ある．）



Input Sentence: $c_1, c_2, \ldots, c_i, \ldots, c_n$
Initialize:
　$i = 1; s = c_1, c_2, \ldots, c_i, \ldots, c_n$
　no_construction = *true*
Start: while *true* do begin
　if $i == |s|$ then
　　if no_construction == *true* then break;
　　no_construction = *ture*
　　$i = 1$;
　else
　　x = *get_contextual_features*(*s*, *i*);
　　y = *classify*(*model.dep*, x);
　　if *y* == **Left** or **Right** then
　　　z = *estimate_Semantic_Role*(*model.sem*, *x*, *y*);
　　　$s = construction(s, i, y, z)$;
　　　no_construction = *false*;
　　else if *y* == **Shift** then
　　　$i = i + 1$;
　　end;
　end;
end;

図 39　依存構造解析と意味役割タグ推定のブートストラップ

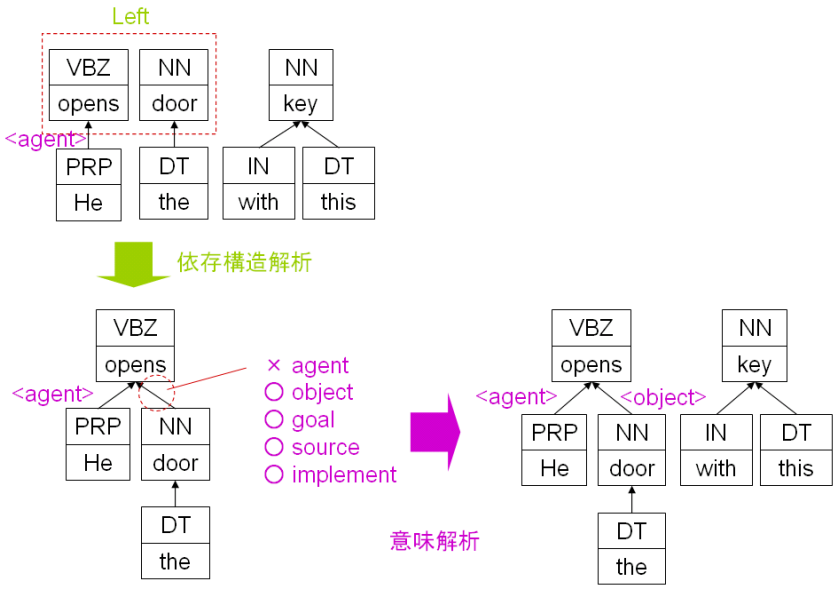


図 40　意味タグ推定結果の動的な利用

6.2　予備実験

まず準備として，山田 [17][18] らが提案する SVM による依存構造解析器と同じものを作成した[*36]．本論文では意味解析のために EDR コーパスを採用しているが，山田らは Penn Treebank の構成素木を依存構造

[*36] 山田らは 2004 年の論文 [19] で SVM による操作の種類に “Wait” などを加えて 6 種類にまで細分化し係り先精度を 90.6% まで向上させたことを報告しているが，今回実装したモデルは操作が 3 種類のもっともシンプルなモデルである．

木に変換したもので解析器の学習・評価をおこなっている．統計的解析モデルの解析精度はコーパスの種類に大きく影響を受けるので，EDR コーパスでの解析精度を山田らの結果と直接比較することはできない．そこで，作成した依存構造解析器の解析性能を試すため，山田らが精度評価をおこなったのと同じ Treebank のデータ（依存構造木に変換）でも精度評価をおこなった．EDR コーパス，Treebank のデータでそれぞれ依存構造解析のみをおこなった際の精度を表 8 に示す．

表 8　予備実験 結果（依存構造解析のみ）
EDR : 約 37000 文で学習，約 10000 文でテスト
Treebank : 39832 文で学習，2416 文でテスト

	EDR	Treebank	
	my model	my model	YAMADA 2003
Dependency Accuracy	79.16%	89.50%	90.03%
Complete Accuracy	23.01%	36.67%	38.40%

今回作成した解析モデル（my model）の解析精度を見てみると，同じモデルでも EDR コーパスのデータで学習した方がかなり悪い結果になっている．一方，同じ my model による Treebank の解析精度が山田らの報告にかなり近い数字を示していることから考えると，EDR コーパスはその依存構造木の記述スタイルの影響から，解析の難しいデータになっているということが見て取れる[*37] ．

6.3　評価実験

予備実験で，山田らの提案する依存構造解析器と同程度の性能の解析器を作成できたことを確認した．次に，この解析器を使って提案する依存構造・意味構造同時解析モデルを作成し，精度を評価した（ここで作成したモデルを「提案モデル 2」と呼ぶ）．アルゴリズムは図 39 の右側の擬似コードに示したとおり，山田らのアルゴリズムに意味役割タグ推定の処理 $estimate\_Semantic\_Role()$ を追加しただけである．

意味役割タグ推定の精度を評価するにあたって，比較対象としては文の依存構造をコーパスから抽出して特徴量に利用した「実装と評価 1」の「依存構造利用モデル」を用いる（「提案モデル 1」は先行詞などの情報を特徴量に追加しているためやや有利になると思われるため）．また，「実装と評価 1」のモデルは既に依存構造解析が正しくおこなわれているデータを用いているため，このモデルと提案モデル 2 を比較するにあたっては，提案モデル 2 の側でも依存構造解析が正しくおこなえている場合を想定しなければ平等な評価ができない．提案モデル 2 のプログラム実装では依存構造のみ教師データを与えて解析することも可能であるため，ここでの比較においては依存構造は全て正しく解析できたとした場合の精度を比べることにする．依存構造利用モデルと提案モデル 2 の意味役割タグ解析精度を表 9 に示す．目立った改善ではないが，若干の精度向上が見られた．依存構造利用モデルは，形態素ごとの特徴量として「原形」を用いているが，提案モデル 2 ではこれは用いていない．形態素ごとの「原形」は，その形態素を限られたコーパスの中で判断するために非常に有用

[*37] 前述のように，前置詞等の係る位置が通常の依存構造と違っており，より広い文脈長が必要になるのではないかと思われる．また EDR コーパスの品詞タグは Penn Treebank よりも粗く，名詞，動詞といった大雑把な分類しか記述されていない．

表 9　評価実験 結果（提案モデル 2 意味タグ推定）
依存構造利用モデル（左）との比較

	依存構造利用モデル			提案モデル 2		
TagName	Precision	Recall	$F_{\beta=1}$	Precision	Recall	$F_{\beta=1}$
agent	72.02%	75.78%	73.85	75.29%	79.24%	77.22
goal	52.87%	40.89%	46.11	53.09%	45.83%	49.19
implement	39.39%	15.12%	21.85	29.87%	16.13%	20.95
manner	59.06%	62.81%	60.88	59.11%	55.67%	57.34
object	66.57%	74.59%	70.36	73.02%	73.93%	73.47
place	54.24%	37.65%	44.44	53.21%	42.29%	47.12
possessor	37.14%	25.49%	30.23	44.10%	24.69%	31.66
purpose	65.12%	43.75%	52.34	55.26%	37.61%	44.76
source	53.13%	34.00%	41.46	61.63%	57.37%	59.42
time	66.09%	63.19%	64.05	65.35%	57.36%	60.91
AVERAGE(weighted)	64.22%	63.88%	64.05	67.57%	64.18%	65.83

な情報であるため[*38]，これを用いずとも同等以上の解析精度が得られたことから，意味役割タグ推定を依存構造解析と同時におこなうことで解析精度の向上が得られる可能性があることが分かる．

依存構造の正解を与えず，全ての依存構造解析と意味役割タグ推定を自動でおこなった場合の意味役割タグ推定精度は表 10，依存構造解析精度は表 11 の右のようになった．

依存構造解析にも誤りが混入しながら意味役割タグの推定をおこなっていくので，依存構造がすべて正しく解析できた場合と比べるとかなり意味解析精度が低下していることが分かる．それでも加重平均で 60 弱の $F_{\beta=1}$ 値は維持しており，agent, object といった主要なタグでは 70 以上の精度で解析ができている．意味解析にかなりの手がかりを与える句構造に関する知識を一切用いず，また固有表現などの付加的な情報も利用することなく，形態素と品詞だけを手がかりにおこなった意味解析としては，かなり良好な結果が出ているといえる．

さらに，表 11 を見ると分かるように，依存構造解析の正解率は単独でおこなったときよりもかなり向上している．前述のように EDR コーパスの依存構造木は Treebank のそれと比べて解析が困難になっているはずだが，このコーパス上でこれだけの精度向上が認められたということは，意味解析を同時におこなうことでより頑強な解析をできるようになったと言える．意味役割タグ推定の結果にもかなりの誤りが混入しているはずだが，その悪影響を含めても，意味的なタグが付加されていることが構文的な解析の助けになっていることが分かる．すなわち，初期的な構文の解析過程において，構文の解析と意味の解析が相互に影響を及ぼしあい，それぞれの精度を高めていくブートストラップが起きていることが示唆される．

[*38] SVM による解析モデルで扱えるのは単語の “ID” であり，例えば “word” と “words” はまったく別のものとして扱われる．これらの間に単数形・複数形という関係があることをデータとして明示するためには，原形のデータを必ず形態素とセットで用意しなければならない．

表 10　評価実験 結果（提案モデル 2 意味タグ推定・自動解析）

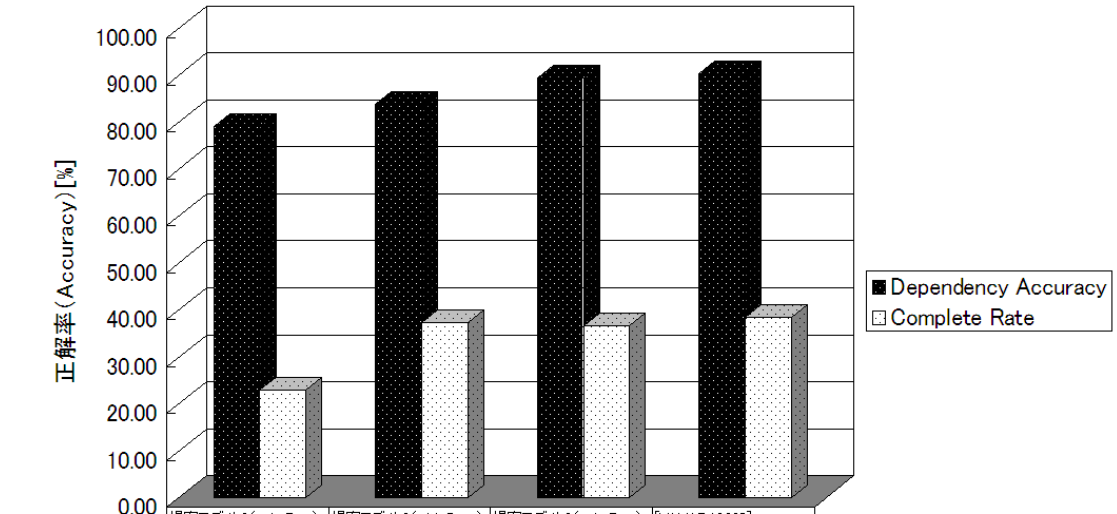
TagName	Precision	Recall	$F_{\beta=1}$
agent	72.95%	76.08%	74.48
and	64.005%	63.37%	63.71
condition	35.73%	29.02%	32.02
goal	46.77%	40.30%	43.30
implement	27.50%	15.43%	19.77
manner	55.22%	52.44%	53.79
number	72.07%	72.83%	72.45
object	65.23%	66.34%	65.78
or	55.70%	54.61%	55.15
place	47.17%	36.69%	41.27
possessor	42.38%	22.87%	29.71
purpose	43.88%	27.31%	33.67
scene	30.44%	22.41%	25.81
source	55.53%	49.77%	52.49
time	59.64%	50.33%	54.59
time-from	54.32%	38.94%	45.36
time-to	41.94%	27.08%	32.91
AVERAGE(weighted)	60.89%	57.15%	58.96

表 11　評価実験 結果（提案モデル 2 依存構造解析・自動解析）：
依存構造解析のみおこなった場合（左）との比較

	only Dependency Analysis	with Semantic Analysis
Dependency Accuracy	79.16%	83.88%
Complete Accuracy	23.01%	37.25%

6.4　参考実験（PropBank への適用）

本論文では依存関係を基礎とした意味解析タスクを設定し，依存構造に密着した意味関係タグを推定するシステムを作成したが，既存の意味解析システムとの比較をおこなうため，既存研究で主に利用されている PropBank の述語-項構造の意味役割タグの推定タスクにも提案手法を適用して実験をおこなった．これは関連研究（p.34）で紹介した Hacioglu の実験とほぼ同様のものであり，PropBank の構成素木を依存構造木に変換してしまう段階で一部の意味役割タグが消失してしまうことから厳密な比較のできるものではないが，Hacioglu の報告していない個別のタグごとの解析精度を大雑把に把握するために実施した．



	提案モデル2(only Dep.)	提案モデル2(with Sem.)	提案モデル2(only Dep.)	[YAMADA2003]
	EDR		PennTreebank	
■Dependency Accuracy	79.16	83.88	89.50	90.30
□Complete Rate	23.01	37.25	36.67	38.40

図 41　依存構造解析の正解率の比較

CoNLL Shared Task 2004 のデータ（学習データ約 9000 文，テストデータ約 1700 文）を依存構造に変換し，これを提案モデル 2 を用いて学習・テストした結果を表 12 に示す．この結果をみると，Arg0, Arg1 などの解析結果はかなり高い数値を示している．それ以降のタグはテストセット中に極端に数が少ないことからばらつきが大きいが，それでも一様に高い精度を示しており，数値だけ見れば CoNLL Shared Task 2004 の他のシステム（p.52 表 6）にも勝っている．しかも，提案モデル 2 の方法では特定の動詞が述語であることをデータ中で明示しておらず，述語動詞の原形を手がかりに利用する CoNLL Shared Task の各種システムよりもかなり少ない情報量のみでの解析となっている．

ただし，前述のとおり学習・テストに用いたデータはそもそも依存構造木への変換の際に一定量の意味役割タグが失われており，もとのデータでの解析精度と同じ尺度で比べることはできない．ここで学習・推定できたのは依存構造にマッチした意味役割関係である．依存構造に直接マッチしていない意味役割関係は推定できないため，その点では既存の句構造に基づく意味解析システムに劣っているといえる．これは，提案モデルの意味解析アルゴリズムが依存構造という構文的構造に密着しすぎていることによるものである．そのため表 12 の結果は，EDR 以外のコーパスにおいても，依存構造にマッチした部分の意味役割タグは提案手法によって既存手法以上の精度で解析が可能なことを示しているといえる．

表 12　参考実験 結果（提案モデル 2 PropBank 意味タグ推定）

TagName	Precision	Recall	$F_{\beta=1}$
Arg0	90.70%	90.90%	90.80
Arg1	86.09%	88.12%	87.09
Arg2	64.66%	58.37%	61.35
Arg3	72.06%	35.25%	47.34
Arg4	76.19%	53.33%	62.75
Arg5	0.00%	0.00%	0.00
ArgM-ADV	65.52%	51.15%	57.45
ArgM-CAU	94.12%	68.09%	79.01
ArgM-DIR	45.45%	16.95%	24.69
ArgM-DIS	72.65%	58.62%	64.89
ArgM-EXT	100.00%	50.00%	66.67
ArgM-LOC	58.98%	69.23%	63.71
ArgM-MNR	53.52%	45.78%	49.35
ArgM-MOD	0.00%	0.00%	0.00
ArgM-NEG	80.00%	72.73%	76.19
ArgM-PNC	57.81%	41.57%	48.37
ArgM-PRD	0.00%	0.00%	0.00
ArgM-TMP	73.47%	75.66%	74.55
AVERAGE（weighted）	79.46%	77.39%	78.08

7 結論

7.1 結論

本論文では，コーパス作成のコストや言語普遍的なデータ記述の観点から整備の難しい構成素木ではなく，よりプリミティブな構文情報である依存構造木を用いて意味役割タグ付けをおこなう意味解析タスクを提案した．既存研究の意味解析タスクよりも細粒度の意味役割のタグ付けをおこなうシステムを開発し，解析精度については，句構造文法の知識をフルに用いた意味解析システムには若干及ばないもののこれに近い解析精度を簡単な特徴量のみから学習したモデルで実現できることを示した．また日本語文のデータに対してもほぼ同様のモデルを作成し，既存研究と同等かそれ以上の精度を達成した．このことから，言語普遍性の高い構文情報である依存構造を用いることで言語依存性が低く自動解析に繋げやすい意味解析システムを構築できることを示した．

また，依存構造と意味構造の相互の関連性を考慮し，依存構造木を作成しながら意味役割タグを同時に推定していくブートストラップ型の解析手法を提案し，解析システムの性能を評価した．その結果，意味解析については劇的な改善は見られないものの若干の精度向上が確認でき，また依存構造解析については正解率の向上が確認できた．この結果から，プリミティブな構文構造である依存構造の処理と意味役割構造の処理とが相互に重要な影響を及ぼしあっている可能性を示した．

7.2 今後の課題

・意味解析システムの精度向上について

本論文で提案した意味解析システムは，他のコーパスの意味役割タグ付けをおこなった既存研究のシステムと直接比較できないものの，ほぼ同程度の性能の意味解析をおこなうことができた．しかし，句構造に基づく構成素木をフルに利用したシステムには及ばず，さらなる精度向上のための工夫が必要である．これには，既存研究で用いている固有表現タグなどの追加特徴量を加えることで比較的容易に意味役割タグの推定精度を向上させることができると思われる．実際に大量の文書に意味解析を施すシステムを作成する場合は，これらの有効な特徴量のなかから現実に利用可能なものを吟味し，利用するべきである．

また，意味解析システムの学習アルゴリズムに関しても検討が必要である．今回用いた Support Vector Machine は既に既存研究で意味解析への有効性が認められており，作成した意味解析システムの高い精度もこの機械学習アルゴリズムの汎化能力に負うところが大きいと思われるが，このアルゴリズムには教師データからの学習に非常に時間がかかる欠点がある．モデルの学習にさまざまな特徴量を工夫すると SVM の取り扱うベクトルの次元が極端に大きくなり，またコーパスのうち学習に用いる文例数を増やすと最適化問題の計算時間が指数的に増えてしまう．一般的に自然言語処理システムの機械学習に用いられるコーパスの規模は Penn Treebank の 40000〜50000 文程度であり，今回の実験でも EDR コーパスのうちの同程度の文例数を用いたが，実際には EDR コーパスには 120000 文強の文例数があり，これを全て用いて意味解析モデルを SVM で学習させるのは，現実的には不可能と言っていい．SVM によるテキストデータの学習では単語の “ID” を扱っており，単語の変化形などはもとの単語とまったく別のものとして扱われるため，その変化形の出現頻度が低いとその単語がそもそも持っている性質が十分にデータ中に表現できない．また，今回提案した依存関係ベースの意味解析は語と語のペアを基本単位として扱うため単語そのものの性質が非常に重要であ

り，十分な数の語の用例がなければ解析がうまく進まない．以上の問題から，意味解析システムの精度向上のためにはさらに大量の文例数を現実時間内に学習できるアルゴリズムの選択が必要になってくると思われる．

・依存構造を意味解析に用いることに関して

本論文の提案では，統語的な依存関係が意味的な依存関係と一致していることを仮定し，統語的依存関係上で意味役割タグを推定した．これは，自然言語処理の意味解析問題で推定する意味役割タグがかなり多くの場合統語的依存関係上に付与されていることを根拠としている．実際，EDR コーパス ではほとんどの深層格タグが統語的依存関係上に付与されている．しかし，実際の文の意味構造ではひとつの語が複数の語に対して異なる意味的関係性を持っている場合があり，必ずしも統語的依存構造のみに従うものではない．このため文書構造化のための標準タグ付け体系として紹介した GDA でも，ひとつの語が依存構造に縛られずに複数の意味を持つことを許すための柔軟性を意識して設計がされている．

今回提案した意味解析モデルでは，EDR コーパスのように依存関係と意味関係が一致しているものでなければ付与することができない．これは，「実装と評価 2」で示した構文解析と意味解析のファーストステップ的な解析法としては非常に合理性のあるものだが，より高次の意味関係を解析することはできない．不定詞の意味上の主語など，直接の依存関係のない語句の間にある意味関係を解析するためには，句構造文法のように部分的要素と文全体との関係を整理できる文法表現のもつ特長もやはり必要になると思われる．

部分と部分の関係を強調する依存文法と部分と全体の関係を強調する構成文法は相対立する文法表現であるが，よりプリミティブで言語普遍的な構造と思われる依存構造からグローバルな構造を表現する句構造へ派生する仕組みが発見されれば，言語普遍的な文構造の表現手段となり，ひいては計算機による文の意味構造の解析にも大きな進展を与えることになるものと考える．

・タグ付き言語資源を普及させるために

本論文では，文書への意味構造の付与を容易にし構造化文書の普及を促すことを将来的な目的として意味解析システムを構築した．コーパスを整備しやすく標準的なタグ付け体系に準拠しやすい依存構造をベースにして意味解析のアルゴリズムを作ることで，意味解析の研究成果をタグ付き文書の普及に繋げやすくしようと考えた．本研究でも解析結果を GDA 形式の XML ファイルに変換するツールを作成し，成果を実際に文書の構造化に利用可能な形で整備するための第一歩としようとしている．GDA で取り扱う意味構造には他にも文同士の関係を表す「修辞構造」などがあるが，この意味構造に関する研究は本論文で取り上げた意味役割（深層格）推定とは別に独自の発展を遂げている．現在，GDA に関する研究では修辞構造を利用した要約などの応用研究が盛んだが，最も細粒度の意味解析である深層格を用いる応用の可能性はまだ十分に検討されていない．深層格推定の成果を GDA における既存の研究と組み合わせ，多粒度の意味構造を記述することができるようになれば，自動要約などの既存研究を超えるキラーアプリケーションの手がかりになるかもしれない．現段階よりも人々の期待にこたえられる言語処理の応用可能性を示すことができれば，文書データに対するタグ付け・構造化のインセンティブとなり，文書データのさらなる有効利用のための正のフィードバックループを形成することができるようになると考えられる．

付録 A　Semantic Tagging Tool の実装

意味構造を表すタグによって構造化された文書の普及を促すためにはすでに存在する大量の未構造化文書を自動解析し，標準的なタギングフォーマットで出力できなければならない．こうした意味解析をおこなう解析器の多くは解析済みコーパスを学習することで解析精度を高めており，一定量の質の高いコーパスが利用可能になって初めて作成できる．逆に，タグ付け作業者の負担を減らすためには意味解析器による自動解析の結果を仮タグ付け結果として利用できることが重要であり，自然言語処理のためのコーパスの整備と機械学習による解析器との関係は互いの成果を利用しあう相補関係にある．

今回作成した意味解析システムも，解析精度等の面で不十分な点を残しながらも文の構文解析・意味解析を自動的におこない，解析済みデータを出力して人手でのタグ付けの助けとなることができる．そこで，作成した意味解析モデルを他の形態素解析器等と組み合わせ，タグ付け支援ツールとして実装した．

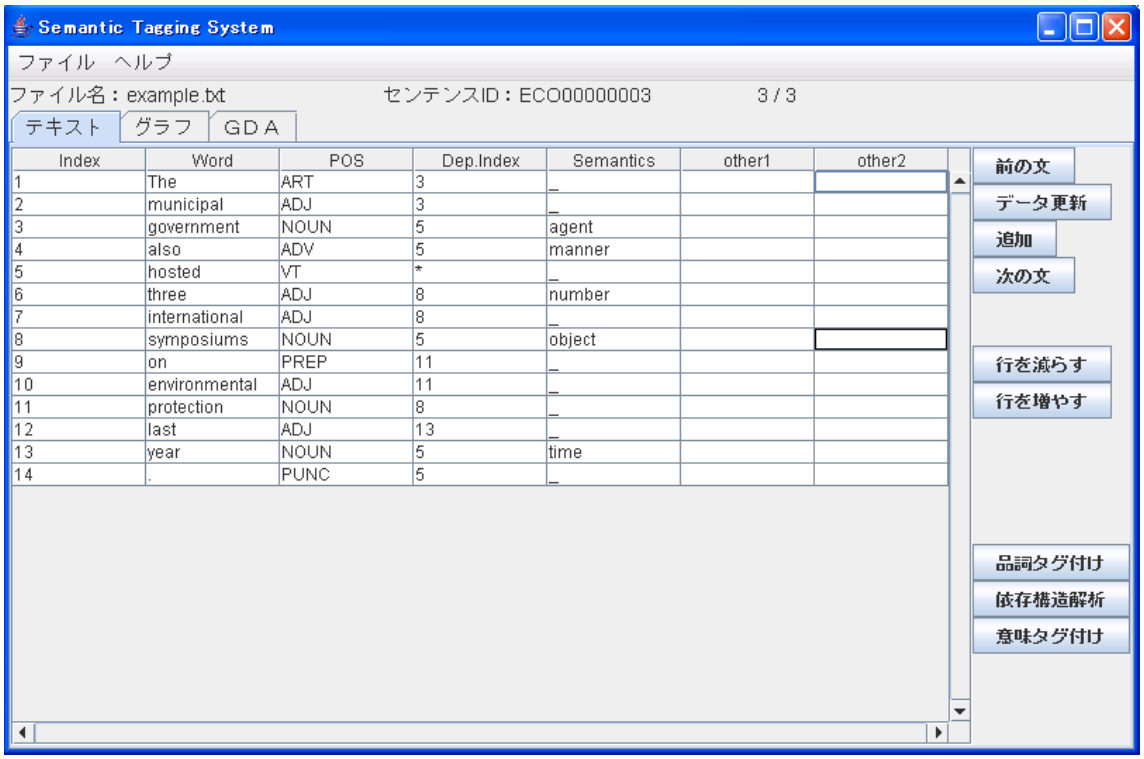


図 42　タグ付け支援ツール：シンプルなテーブル表示

形態素の抽出は既存の形態素解析器を利用し（今回は TreeTagger を利用した），形態素列としてテーブルに表示した文例に対し，提案モデル 2 による意味解析システムを用いて品詞タグ付け → 依存構造解析 → 意味タグ付け を実行できるテーブル表示の画面をメインとし，解析結果を意味タグ付き依存構造木としてＧＵＩで表示するグラフ画面，GDA のタグ付け基準に準拠した XML にフォーマット変換して表示する GDA 画面を用意した．また，実際のコーパス作成では大量の文をひとつずつ解析，タグ付けをしていく必要があるため，ファイル内の文例を大量に読み込み，作業をしながら文例を移動できるようにしている．

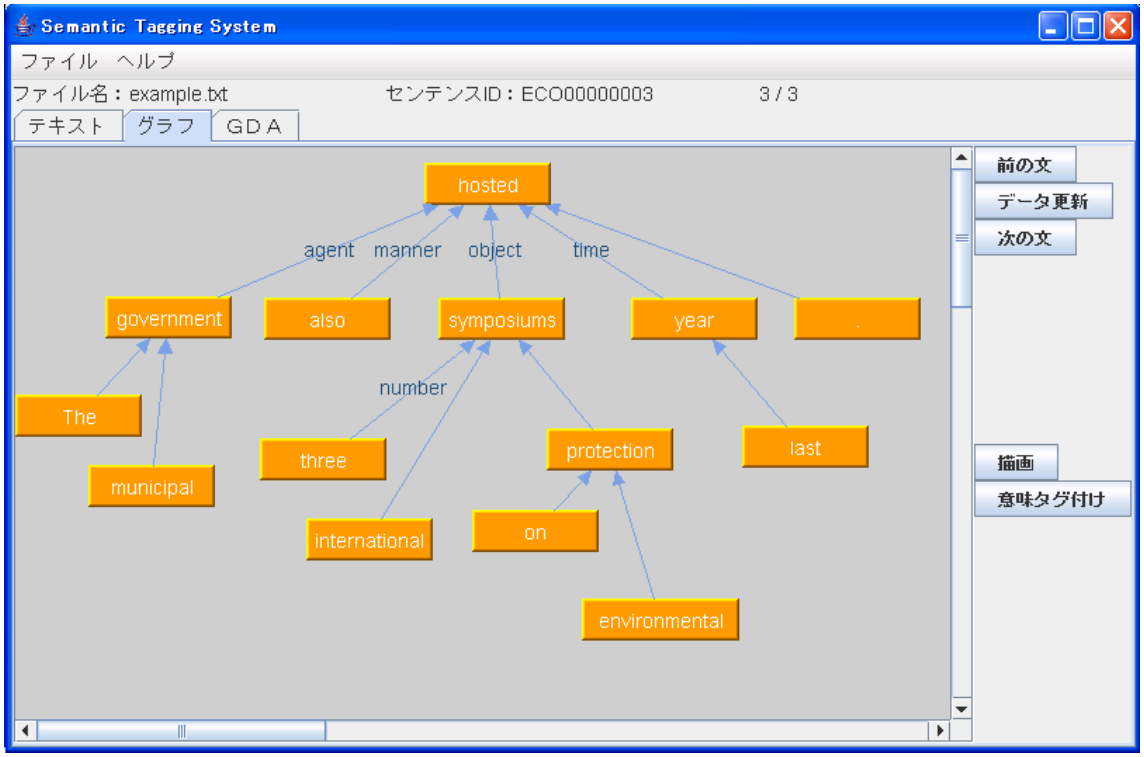


図 43　タグ付け支援ツール：ＧＵＩによる依存構造木の表示

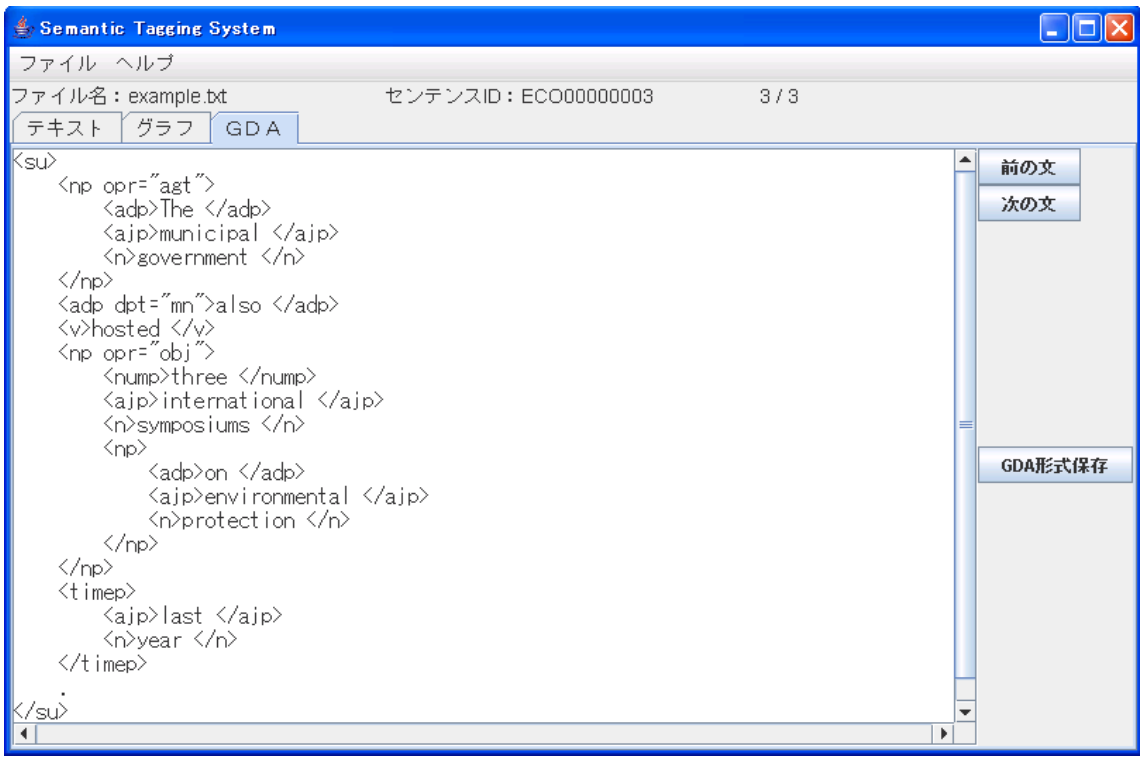


図 44　タグ付け支援ツール：GDA スタイル XML フォーマットへの変換

付録 B　EDR コーパスの概念関係子タグ

本論文で意味解析システムを構築する際に学習データに利用した EDR コーパス中に定義されている意味役割（正確には概念関係子）タグのうち，主要なものの示す意味関係は以下のとおりである．

概念関係子名	意味
agent	有意志動作を引き起こす主体
object	動作・変化の影響を受ける対象
implement	有意志動作における道具・手段
material	材料または構成要素
source	事象の主体または対象の最初の位置
goal	事象の主体または対象の最後の位置
place	事象の成立する場所
scene	事象の成立する場所
basis	比較の基準
manner	動作・変化のやり方
time	事象の起こる時間
time-from	事象の始まる時間
time-to	事象の終わる時間
quantity	物・動作・変化の量
modifier	修飾関係
number	数
and	概念間の連結関係
or	概念間の選択関係
condition	事象・事実の条件関係
purpoes	目的
possessor	所有関係
beneficiary	受益者・被害者
unit	単位

参考文献

[1] Daniel Sleator and Davy Temperley; “Parsing English with a Link Grammar”: Carnegie Mellon University Computer Science technical report CMU-CS-91-196, 1991.10

[2] Colin F. Baker, Charles J. Fillmore, John B. Lowe; “The Berkley FrameNet Project”: Proceedings of COLING/ACL, pp.86-90, 1998

[3] 黒田 航, 高梨 克也, 竹内 和弘, 井佐原 均; “複層意味フレーム分析の紹介”: The 19th Annual Conference of the Japanese Society for Artificial Intelligence, 2005

[4] Daniel Gildea, Daniel Jurafsky: “Automatic Labeling of Semantic Roles”; Computational Linguistics 2002, 28(3):245-288, 2002

[5] Namhee Kwon, Michael Fleischman, Eduard Hovy; “Frame-Net Based Semantic Parsing Using Maximum Entropy Models”: Proceedings of 21st International Conference on Computational Linguistics, 2004

[6] Lei Shi, Rada Mihalcea; “An Algorithm for Open Text Semantic Parsing”: Proceedings of the Robust Methods in Analysis of Natural Language Data, 2004.8

[7] Nobukazu Shibui and Akito Sakurai; “FrameNet-Based Shallow Semantic Parsing with a POS Tagger”: Proceedings of Joint Workshop of Vietnamese Society of AI, SIGKBS-JSAI, ICS-IPSJ and IEICE-SIGAI on Active Mining, 2004.12

[8] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky; “Shallow Semantic Parsing Using Support Vector Machines”: Proceedings of HLT-NAACL 2004

[9] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky; “Support Vector Learning for Semantic Argument Classification”: Machine Learning Journal, Volume 60, Issue 1-3, pp.11-39, 2005.9

[10] Kadri Hacioglu, Wayne Ward; “Target Word Detection and Semantic Role Chunking Using Support Vector Machines”; Proceedings of HLT-NACCL 2003, Edmonton, Alberta, Canada, pp. 25-27, 2003.5.

[11] Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H.Martin, Daniel Jurafsky; “Semantic Role Labeling by Tagging Syntactic Chunks”: Conference on Computational Natural Language Learning 2004, 2004.3

[12] Kadri Hacioglu; “Semantic Role Labeling using Dependency Trees”: Computational Linguistics 2004, 2004.8

[13] Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky; “Semantic Role Chunking Combining Complementary Syntactic Views”: Conference on Computational Natural Language Learning 2005, 2005.6

[14] Tomohiro Mitsumori, Masaki Murata, Yasushi Fukuda, Kouichi Doi, Hirohumi Doi: “Semantic Role Labeling Using Support Vector Machines”: Conference on Computational Natural Language Learning 2005, 2005.6

[15] Dekang Lin; “DEPENDENCY-BASED EVALUATION OF MINIPAR”: Workshop on the Evaluation of Parsing Systems, 1998

[16] Dekang Lin, Patrick Pantel; “Discovery of Inference Rules for Question Answering”: Natural Language Engineering, 7 (4):pp.343-360, 2002

[17] 山田 寛康，松本 裕治; “Support Vector Machine を用いた英語依存構造解析”: 自然言語処理 152-8, 2002.11

[18] Hiroyasu Yamada, Yuji Matsumoto; “Statistical dependency analysis with Support Vector Machines”: International Workshop on Parsing Technologies, pp.195-206, 2003.4

[19] 山田 寛康，松本 裕治; “Support Vector Machine を用いた決定性上昇型依存構造解析”: 情報処理学会論文誌 Vol.45 No.10, 2004.10

[20] 橋田浩一; “GDA：意味的修飾に基づく多用途の知的コンテンツ”: 人工知能学会誌, 13:528-535, 1998

[21] Masashi Shimbo, Hiroyasu Yamada, Yuji Matsumoto; “Using Syntactic Dependency Information for Calssification of Technical Terms”: Proceedings of the 7th Pacific Rim Knowledge Acquisition Workshop, pp.131-143, 2002.8

[22] 新保 仁，松本 裕治，山田 寛康; “利用者からの要求を考慮したテキストデータからの知識抽出”: 人工知能学会誌 20 巻 2 号, 2005.3

[23] 綾 聡平，松尾 豊，岡崎 直観，橋田 浩一，石塚 満; “修辞構造のアノテーションに基づく要約生成”: 人工知能学会論文誌 20 巻 3 号 B, 2005

[24] 森本憲悟，鬼城渉，藤本裕，下村芳樹，吉岡真治，武田英明; “Universal Abduction Studio の開発（第 7 報） -深層格と構造類似性を用いた知識マッチング-”: 2005 年精密工学会春季大会学術講演会講演論文集, 2005

[25] 村上 裕人，谷澤 嘉和，韓 東力，原田 実; “意味解析による自由記述アンケートの自動分類システム AQUA”: 情報処理学会第 66 回全国大会 2004.3

[26] 野口 貴，韓 東力，原田 実; “反復語句・必須格・文間深層格を考慮した要約システム ABISYS”: 情報処理学会第 66 回全国大会, 2004.3

[27] 原田 実，田淵 和幸，大野 博之; “日本語意味解析システム SAGE の高速化・高精度化とコーパスによる精度評価”: 情報処理学会論文誌, 2002.9

[28] 井村 裕，沓掛 俊樹，佐藤 直美，原田 実; “意味解析システム SAGE の Web 化と連体・使役・受身における解析精度向上”: 自然言語処理 153-4 2003.1

[29] 小山 正太，乾 伸雄，小谷 善行; “「名詞と表層格」パターンに対する深層格対応の推測”: 情報言語処理 154-22, 2003.3

[30] 渋木 英潔，荒木 健治，栃内 香次; “一文一格の原理と深層格選好に基づいた日本語深層格規則の自動獲得手法”: 情報処理学会研究報告, 2003-NL-156(3)

[31] 栗田多喜男; “サポートベクターマシン入門”: http://www.neurosci.aist.go.jp/

[32] 池田和司; “神経回路網への展開 サポートベクトルマシンの学習曲線”: 数理科学 No.489, 2004.3

[33] Nello Cristianini, John Shawe-Taylor 著，大北 剛 訳; 『サポートベクターマシン入門』: 共立出版, 2005.3

[34] 辻井 潤一 著，北 研二 編; 『確率的言語モデル』: 東京大学出版会, 1999

[35] チャールズ・J. フィルモア 著，田中 春美，船城 道雄 訳; 『格文法の原理 : 言語の意味と構造』: 三省堂, 1975.8

[36] 児玉徳美; 『依存文法の研究』: 研究者出版, 1987.3

[37] 児玉徳美; 『意味論と対象と方法』: くろしお出版, 2002.12

謝辞

開放環境科学専攻教授の櫻井彰人先生には，本研究を進めるにあたり数々のご助言をいただき，研究を深めていくのに多大なるお力をお借りすることができました．また学部四年次よりご指導をしていただき，本研究を進めるために必要な力を身に付けるのにさまざまな体験をさせていただきました．心より感謝申し上げます．

開放環境科学専攻教授の山口高平先生には，異なる分野のさまざまな研究に触れる機会を作っていただき，さらに見聞を広げることができました．深く感謝いたします．

開放環境科学専攻助手の飯島正先生には，中間発表で的確なご助言をいただき，研究をさらに深めるきっかけを与えてくださいました．ありがとうございました．

開放環境科学専攻助手の篠沢佳久先生には，実験方法や計算機の扱い方にいたるまでさまざまな場面で助けていただきました．心から感謝いたします．

櫻井・山口研究室の博士課程の先輩方には，豊富な経験と知識を頼りにさせていただきました．また後輩の方々には，共に学び，研究を助けていただきました．本当にありがとうございます．

共に研究を進めた修士二年生の皆さんとは，お互いに異なる研究を進めながらも教えあい，励ましあうことができました．ありがとうございます．

最後に，論文執筆を支えてくれた姉と妹，本研究をおこなう機会を与えてくれた両親に心から感謝します．ありがとうざいました．

2006 年 2 月 3 日