Maximum Entropy法による人名抽出

執筆者：萩原 裕一

指導教官：新納 浩幸

平成１２年３月１日

目 次

図目次

表目次

第1章　序論

1.1　概要

本論文は、「Maxinum Entropy（ME）法による人名抽出」について述べる。

人名を始めとする固有表現は、新聞記事なのどの文章を理解するためにキーになる要素である。情報抽出の前処理としてこれらの固有表現抽出が書かせない。情報抽出は近年自然言語処理において精力的に研究されている分野である。

固有表現の抽出はクラス分類問題への変換が可能であり、各単語がどのクラスにどの程度の確率で属するかを求めることで行なえる。従来、クラス分類手法として決定木を利用して固有表現抽出を行なった研究があったが、ここでは、データスパースネス問題に効果的であるME法を試みる。

本研究では固有表現は人名のみとする。そして人名抽出にME法を利用できることを確認することが本研究の目的である。またクラス分類の際に、従来設定されていたクラス [NONE] を4つのクラスに細分化する工夫も採り入れる。

1.2　本論文の構成

本論文は、まず情報抽出とは何かについて述べる。そして情報抽出の前処理として必要な、固有表現の定義と本論文の目的である人名表現の定義を示す。その後、本研究で用いたME法の説明し、人名抽出の実験・結果とその考察について述べる。

第2章　情報抽出

本研究でいう情報抽出は、その分野で中心的存在である米国の Message Understanding Conference(MUC) における定義に準ずる。ここでは情報抽出とその際に必要な技術と問題点、応用分野についても述べる。

2.1　情報抽出とは何か

MUC における情報抽出とは、新聞記事などのようなテキストからあらかじめ指定されたイベントや事柄に関する情報を抽出し、その情報をデータベースに入力する、という技術で、近年、自然言語処理においては精力的に研究されている分野である。

図 2.1 に人事異動に関する新聞記事を元にした情報抽出結果を示す。ここでは抽出したい情報は、企業の重役の異動（昇進、降格、退任など）に関する情報であり、抽出したい情報の内容としては、該当者の人名、会社、異動前役職名、異動後役職名、異動発生日というように与えられている。御覧のように、面倒な文章で書かれた人事異動の情報がすっきりとしたデータベース形式で抽出されている。

この情報抽出という技術は、特定の情報を簡単に調べたい時などに非常に役に立つ。例えば、過去 10 年の新聞記事などから、企業の重役の異動に関する情報を得たいという場面を想定してみる。

現在良くある情報検索の技術を利用すると、適当な検索式を作成し、異動に関連した記事を引っ張りだすことはできる。しかし、その結果は膨大であり、実際に必要な情報は 1 つ 1 つの記事を読んでみなければ得ることができない。また、自動要約という技術を利用して、その手間を少なくすることは可能であるが、それでも文章を読まなければいけないことには変わりはない。また、今の技術では、特定の視点に注目したよう要約はまだ開発途上であると言わざるを得ない状態である。これに対し、情報抽出の技術を利用した場合には、表 2.2 のような表形式で、複数の異動の情報を視覚的に一度に見ることができる。表形式になっているので、特に注目したい欄を対象にフィルターをかけることなども可能である。つまり、情報検索結果の記事を 1 記事あたり数分かけて読みながら調べるというのに比べて、非常に高速に必要な情報を得ることができる。

ここでは一例として、重役の人事異動を取り上げたが、対象となる情報は、もちろん、それだけに限らない。一般的に前もって、「抽出したい情報の型」が決め

表 2.1: 情報抽出の例

< 新聞記事 >

ABC 株式会社は 12 日、臨時取締役会で田中一郎社長が代表権のある会長に就任し、山だ次郎副社長が社長に昇格する人事を内定したと発表した。鈴木三朗会長は代表取締役に留まる。3 月 25 日に開く株主総会後の取締役会で正式決定する。田中社長は五期十年社長を務め、年齢も 71 歳と高齢になったため、若返りをはかる。

< 異動イベント情報 >

人名	田中一郎
会社名	ABC 株式会社
異動前役職名	社長
異動後役職名	会長
異動理由	昇格
異動発生日	3 月 25 日

人名	山田次郎
会社名	ABC 株式会社
異動前役職名	副社長
異動後役職名	社長
異動理由	昇格
異動発生日	3 月 25 日

人名	鈴木三朗
会社名	ABC 株式会社
異動前役職名	会長
異動後役職名	代表取締役
異動理由	昇格
異動発生日	3 月 25 日

表 2.2: 情報抽出の結果の例

人名	会社名	異動	異動理由	発生日
田中一郎	ABC 株式会社	社長 → 会長	昇進	1999 年 3 月 25 日
山田次郎	ABC 株式会社	副社長 → 社長	昇進	1999 年 3 月 25 日
鈴木三朗	ABC 株式会社	会長 → 代表取締役	降格	1999 年 3 月 25 日

[固有名詞 | カタカナ文字列 | アルファベット列] “株式会社” → @ 企業 数字 “月” 数字 ” 日” → @ 日時
[“社長” | “副会長” |“会長” | ...] → @ 役職 @ 日時 “,” @ 企業名 “の” @ 人名 @ 役職 “は” @ 役職 “に”” 昇進した” → @ 昇進イベント

図 2.1: パターンの例

られるものなら特にこだわらない。

例えば、合弁企業の情報、新製品の情報、信じ行の情報などの産業会に役に立ちそうな内容や、研究者向けには科学技術論文における技術内容の情報抽出や、医療カルテ、ゲノムといった特定分野のテキストにおける情報（例えば、タンパク質の役割の情報抽出)、また、アイドル歌手の活動の情報抽出、スポーツなどの特定のイベントなどと個人的な利用も含め広く考えることができる。

この技術のポイントは、一般的にテキストを理解するという技術に比べて、あらかじめ抽出したい情報の型が与えられているということにより、実現容易だという点にある。すでに、いくつかのデモシステムは作成されており、実際に商用に使われ始めているシステムもある。

2.2 利用されている技術

先ほど、情報抽出がテキストを理解するよりも容易であると述べた。ここでは実際にどのような点で容易なのか、また逆にシステムを開発したり使用したりする際に制約が生じていないかというようなことについて述べる。

パターンマッチングによる情報抽出

一般にテキスト解析というと、自然言語処理における構文解析や意味解析を行なうと考えられがちであるが、これらの技術はまだ困難な問題が残り、何ら制約のない環境で広く利用可能であるとは言いがたい。それに対して、情報抽出の実用が広く可能になっているのは、これらの難しい技術を使用して深い理解を試みることなく情報抽出と実現できる技術が生まれたと言う点に集約できる。それがパターンマッチングによる情報抽出と言う技術である。

初期の MUC では構文解析などの技術を用いる方式が主流であったが、パターンマッチングの方式が性能的に優れていたため淘汰されてしまった。現在は一部で統計的構文解析などを利用した試みはあるものの、広くは利用されていない。

パターンマッチングは、その情報抽出の対象に関係する文や文の一部にマッチするパターンを図 2.2 のように用意しておいて、それを決まった順に適応し、決定的に情報をつかんでいくと言う技術である。

例えば、図 2.2 の最初のパターンを言葉に直すと、「固有名詞かカタカナ列かアルファベット列の後に、株式会社と言う文字列があった場合にはそれらをまとめて

入力文：2月7日、XYZ株式会社の高橋四郎副社長は社長に昇進した。
↓
2月7日、(@企業）の高橋四郎副社長は社長に昇進した。
↓
（@日時)、(@企業）の（@人名）(@役職）は（@役職）に昇進した。
↓
（@昇進イベント）

図 2.2: パターンマッチングの例

企業と判断する」ということである。@で始まる要素は、何らかのカテゴリを示す。図2.2では、入力文が次第にパターンマッチングにより@のついたカテゴリに変換され、最終的に昇進イベントを得るという様子が描かれている。まず「XYZ株式会社」が最初のパターンにより「@企業」に変換され、その他、日時、人名、役職などが随時変換されていく。そして、最後のパターンを利用して、昇進イベントを得るという流れになる。最後の昇進イベントを得る際には、それぞれの項の内容を記憶し、それぞれの役割の抽出も同時に行なう。これによって表2.1に示したような情報が抽出されることになる。

ここでは、説明のために簡略化してパターンマッチングの流れを示したが、実際にはこれだけでは十分ではない。人名などの抽出はなかなかパターンのみでは困難であり、辞書的な情報との組合せが必要になってくる。また、パターンは注意深く設計しないと、意図しない文字列とマッチングしてしまうことも考えられる。そして、正確に情報を把握するためにはパターンマッチングの順序も重要である。また現実的には子のような1文に対するパターンマッチングだけですべての情報を得られるとは限らない。1つのイベントが複数の文にまたがって書かれていることもある。このような問題を解決するためには照応解析の技術が必要になってくる。これは例えば、表2.1の例で、2つ目の文にある鈴木三朗がABC株式会社の会長であることを理解するためには、その前の文の情報が必要であるということである。このような照応を厳密に行なうには、難しい問題があるが、ある程度までの照応解析は、シンプルなルールで解決することができる。

情報抽出は、これまでに述べたパターンマッチングの技術を応用して、実用かに向けて大きく成長してきた。しかし、このパターンマッチングの枠組から、システムの開発や利用においてある種の制約を生じさせてしまっている。

2.3 技術的な課題

情報抽出においてパターンマッチング技術を利用したことによって生じた最大の課題は、パターンを情報抽出の課題ごとに作成しなければいけないという点である。つまり、人事異動のパターンは人事異動にしか使用できず、新たに企業合弁

に関する情報抽出を行ないたい時にはそのためにパターンを新たに作成しなければいけないということである。課題によって必要なパターンの数は異なるが、ある程度複雑な課題の場合には、少なくとも500から1000近いパターンが必要になってくる。これらのパターンを課題が与えられるごとに1から手作業で作成していくのでは、そのコストが膨大になり、あまり実用的であるとは言い難い。システムがある特定の情報抽出の要望だけを処理するのならばいいが、望むべくは幅広い課題に対処できるようにしたい。この問題に対しては大きく3つの方向で研究が進められている。

1つは、課題の種類に依存しないパターンと、課題に依存したパターンを切りわけ、前者をライブラリとして用意しておくとい言う方法である。たとえば、図2.2の例では重役の昇進という課題以外でも十分に有効であり、課題に依存しないパターンとして利用できる。また、課題に依存するパターンも課題の種類によっては流用できるものもあると思われる。そのような分野依存性を的確に判断しパターン作成の労力を少しでも減らそうというのが1つ目の方針である。

2つ目は、パターンを作成するためのツールを用意し、言語処理や計算機システムなどに詳しくない人でも簡単にパターンを作成できるような環境を用意する方法である。例えば、ニューヨーク大学では例文を元に、簡単にパターンを作成するツールを開発している。パターンのようその一般化や構文的なバリエーションの自動的な作成などをサポートし短時間で特定の課題に対するパターンを作成できるようになっている。

最後には、パターンを自動的に作成するという方法である。基本的には大量の文章を元に、動詞の使われ方や固有表現の出現情報などからパターンを自動的に作成するという取り組みが行なわれている。この分野はまだ研究が開始されたという段階である。

2.4 要素技術の分割

最初に紹介したMUCでは、システムのモジュラリティやポータビリティを向上させることを目的に、情報抽出における要素技術を分割し、それぞれの技術をそれのみで評価しようという試みが行なわれた。この分割は、単に情報抽出の開発に役立っているだけでなく、それぞれの要素技術の問題点を整理したり、また、単独で応用システムに無味込まれたりするのに役立っている。

まずは図2.2の例からも明らかであるが、人名、組織名、時間などの表現の抽出は1つの要素技術として分割できる。この課題はテキストの中にある固有表現を抽出し、種類を認定することが課題とされており、すでに情報検索の分野で応用されている。一例としては、地名とも人名ともとれる表現（例えば、成田やワシントン）を地名として検索したいか、人名として検索したいかをあらかじめ指定しておくことによって、検索ミスを減らそうといった応用である。

また、組織名や人名には課題によらない決まったか他があると考えられる。例えば、組織名なら、所在地、社長名、事業分野などがある。そのような情報の型（テンプレート）を分野によらず決めておいて、その情報を抽出することを目的としたかだいもある。この技術はテンプレート抽出技術と呼ばれている。

最後に、パターンマッチングのところで触れたが、照応解析というのは情報抽出にとって重要な技術である。これも独立した課題として扱うことができるし、この技術は他の自然言語処理応用システムに利用することができる。

2.5 応用分野

情報抽出は技術的には成熟期に手が届いているという状態であり、米国は元より、日本でも製品化や、デモシステムの実例がある。

- 高度情報検索としての応用

  現在の情報検索が持つ問題点を解決していこうという目的が対象である。インターネットを始めとする電子上のテキストが膨大になるにつれ、非常に有望な応用課題となっている。新聞記事や特定の情報リソースからの情報抽出はもとより、電子図書館での応用なども考えられる。

- データベースの（半）自動構築

  世界のデータベースのコンテンツのシェアは米国が断突であり、日本でもその作成が急務であると伝えられている。このようなデータベースの作成は非常に手間のかかる仕事であるが、それを推進するツールとしての利用価値が考えられる。

- 情報抽出のツール

  米国やMUCが盛んであった背景には、米国の軍や情報中央局で、実際に情報抽出の作業を人手で行なっているということが挙げられる。聞くところによると、実際に100人単位の人間が、世界の治安、政治、麻薬などに関する情報を、世界中の新聞や通信者から集め整理し、製作立案に役立てているということである。子のような作業を自動化することがMUCのスポンサーである米軍や情報中央局の意図であり、実際に補助ツールとして使用されているそうである。

  日本の場合には、北朝鮮のミサイル疑惑などの際に、日本の情報収集の重要性が話題になったが、企業戦略における情報収集といった応用も考えられる。

第3章　固有表現

ここでは、固有表現の定義について述べる。本研究はIREXによる固有表現の定義にしたがっているが、抽出の対象は人名のみである。

人名抽出はクラス分類問題として捉えられる。具体的には抽出対象の文の各単語にクラスを割り当てるのだが、人名表現抽出に適用される規則について、固有表現全般に共通の規則と、人名表現固有の規則について説明する。

3.1　固有表現の定義

IREXでは固有表現の種類を、固有名詞的表現として組織名、人名、地名、固有物名、時間表現として日付表現、時刻表現、数値表現の種類として金額表現、割合表現の8種類を定めている。

固有名詞的表現

- 組織名 (ORGANIZATION)

  複数の人間で構成され、共通の目的を持った組織名の名称である。株式会社等の会社、固有の政府や組織、学校、軍、スポーツチーム、国際組織、労働組合、工場、ホテル、空港、病院、教会や何ら化の目的を持ったグループなどもその対象が組織としてのいみで使われている文脈においては組織名とする。

- 人名 (PERSON)

  固有の人を指す名前。

- 地名 (LOCATION)

  固有の場所を指す名称。大陸、国名、地域名、都市名、地方名、県名、町名、道路名、住所、駅名、線路名、モニュメント、海洋名、湾、運河、川名、池名、湖名、島、公園、山、砂漠の名前などを含む。

- 固有物名 (ARTIFACT)

  人間の活動によって作られた具体物、抽象物を含む固有の物の名称。

時間表現

表 3.1: クラス割り当て

クラス	説明
PS(Person-Start)	人名の始まりの単語
PM(Person-Middle)	人名の途中の単語
PE(Person-End)	人名の終りの単語
PI(Person-Itself)	それ自身が人名
NS(None-Start)	人名の始まりの単語
NM(None-Middle)	人名の途中の単語
NE(None-End)	人名の終りの単語
NI(None-Itself)	それ自身が人名

- 日付表現 (DATE)、時刻表現 (TIME)

  時間表現では、絶対的な表現 (1999 年 2 月 1 日など) や、起点が明確であり絶対的な時間がわかるような相対的表現 (昨日など) を抽出する。日付表現は、その単位が 24 時間以下であるものを指す。

数値表現

- 金額表現

  金額を表す言葉。

- 割合表現

  割合を表す言葉。

3.2 人名抽出のクラス分類問題への変換

固有表現の抽出はクラス分類問題として捉えることができる。

人名表現の抽出を行なう場合、各単語に表 3.1 のようにクラスを割り当てれば良い。

本研究では固有表現抽出の中でも人名のみを対象としている。よって、人名以外の表現にはクラス NONE を設定している。基本的には各単語に一意にクラスを割り当てることはできない。例えば、クラス PS の次にクラス NS が来たり、クラス NI が来ることはあり得ない。ようするに、人名表現の途中でそのほかの固有表現などの人名ではない表現は出現しないということである。このクラス割当ができれば、人名表現の抽出が可能となる。

3.3 固有表現抽出の際の共通規則

固有表現の抽出では以下の8種類の固有表現の抽出を行なう。

	開始位置タグ	終了位置タグ
* 固有名詞的表現		
* 組織名、政府組織名	<ORGANIZATION>	</ORGANIZATION>
* 人名	<PERSON>	</PERSON>
* 地名	<LOCATION>	</LOCATION>
* 固有物名	<ARTIFACT>	</ARTIFACT>
* 時間表現		
* 日付表現	<DATE>	</DATE>
* 時間表現	<TIME>	</TIME>
* 数値表現		
* 金額表現	<MONEY>	</MONEY>
* 割合表現	<PERCENT>	</PERCENT>

それぞれの固有表現文字列の開始、終了位置に、システムは重複や入れ子のない唯一のタグのペアをふる。もし、表現が重なっている場合は、原則的に長い単位の表現を抽出する。(例：日本銀行では日本を地名として抽出するのではなく、日本銀行全体を組織名として抽出する。) タグの種類は上記の分類に示された通りである。

3.3.1 固有名詞的表現

固有名詞的表現は、組織名、人名、地名、固有物名の固有の対象を示す表現を言う。

固有名詞や固有名詞を含む複合語、その省略形などの形で表現されている。ただし、指示代名詞や普通名詞を利用した照応表現はここでの抽出対象には含まない。(例：それ、当局、同県、委員会)

表現が、例えば、組織名にも地名にも取れるものがある。(例：成田空港問題。成田空港に着陸した。ホワイトハウスの発表。ホワイトハウスに到着した。) その場合は、使用されているコンテクストから、それがその場合にどちらを意味するかを参考に判断する。それでも曖昧なものについてはOPTIONALを利用する。

以下に固有名詞的表現全体に適用される規則を示す。

3.3.2 部分表現

名詞連続や接辞が付いたものでも、その一部に固有名詞的表現を含む場合はそれを抽出する。また、全体として慣用的に普通名詞的に用いられている場合には、そのうちの一部が固有名詞的表現であっても、語源まで逆登らなければいけないような場合や、「言語」の場合、その他曖昧な物についてはOPTIONALとする。表記上地名ではない物(アメリカン、ナポリタン等)はまったく抽出しない。

<ORGANIZATION>ＮＨＫ</ORGANIZATION>番組
<PERSON>クリントン</PERSON>大統領
<LOCATION>日本</LOCATION>市場
<ORGANIZATION>米軍</ORGANIZATION>機
<ORGANIZATION>衆</ORGANIZATION><ORGANIZATION>参</ORGANIZATION>議長
<ORGANIZATION>労働党</ORGANIZATION>員
<ORGANIZATION>沖電気関西研究所</ORGANIZATION>長
<ORGANIZATION>沖電気関西研究所</ORGANIZATION>所長
<ORGANIZATION>製造物責任問題検討会</ORGANIZATION>委員長
<ORGANIZATION>武蔵野音大</ORGANIZATION>大学院
来<LOCATION>日</LOCATION>
訪<LOCATION>米</LOCATION>
<ORGANIZATION>文</ORGANIZATION>相
<ORGANIZATION>農水</ORGANNIZATION>大臣
<PERSON>鈴木</PERSON>家
<LOCATION>フィリピン</LOCATION>人
在<LOCATION>エジプト</LOCATION><LOCATION>日本</LOCATION>企業
<LOCATION>パリ</LOCATION>ジェンヌ
<LOCATION>江戸</LOCATION>っ子
<LOCATION>京</LOCATION>女
<LOCATION>東</LOCATION>男
<OPTIONAL>フランス</OPTIONAL>人形
<OPTIONAL>フランス</OPTIONAL>料理
<OPTIONAL>瀬戸</OPTIONAL>物
<OPTIONAL>唐</OPTIONAL>きび
<OPTIONAL>五右衛門</OPTIONAL>風呂
<OPTIONAL>川崎</OPTIONAL>病
<OPTIONAL>ハンセン</OPTIONA>氏病
オー・デ・<OPTIONAL>コロン</OPTIONAL>
<OPTIONAL>漢</OPTIONAL>字
<OPTIONAL>英</OPTIONAL>語
<OPTIONA>和</OPTIONAL><OPTIONAL>英</OPTIONAL>辞典

アメリカンコーヒー
スパゲッティーナポリタン

3.3.3 助詞「の」、特殊記号

助詞の「の」や特殊記号の「,.・／　(スペース)」等は固有表現を分割し、いずれの固有表現もそれを含まない。ただし、慣用的にそのような物を含む表現や、人名で苗字と名前を記号で継げるような場合には、それを含む。

<ORGANIZATION>日本銀行福岡支店</ORGANIZATION>
<ORGANIZATION>日本銀行</ORGANIZATION>の<LOCATION>福岡</LOCATION>支店
<ORGANIZATION>日本銀行</ORGANIZATION>・<LOCATION>福岡</LOCATION>支店
<LOCATION>大阪</LOCATION>の工場
<ORGANIZATION>京大</ORGANIZATION>の<PERSON>長尾眞</PERSON>氏
<LOCATION>東京</LOCATION>　<LOCATION>銀座</LOCATION>
<ORGANIZATION>影の内閣</ORGANIZATION>
<PERSON>ビル・クリントン</PERON>

3.3.4 連続固有表現

連続固有表現はそれぞれが独立した固有表現で、前のものが、後のもののスーパークラスになっている場合は分割しない。ただし、並列な固有表現が連続している場合にはそれらは分割する。また、部分的な並列表現で分割すると違った意味のものを生んでしまう場合には分割しない。

<ORGANIZATION>日本銀行福岡支店</ORGANIZATION>
<ORGANIZATION>東工大水泳部</ORGANIZATION>
<ORGANIZATION>東京読売巨人軍</ORGANIZATION>
<LOCATION>東京都目黒区大岡山２－１２－１</LOCATION>
<LOCATION>東京銀座</LOCATION>
<LOCATION>米軍立川基地</LOCATION>
<LOCATION>東京</LOCATION><LOCATION>大阪</LOCATION>
<LOCATION>日</LOCATION><LOCATION>韓</LOCATION>両国
<LOCATION>欧</LOCATION><LOCATION>米</LOCATION>
<LOCATION>薩</LOCATION><LOCATION>長</LOCATION>
<ORGANIZATION>衆</ORGANIZATION><ORGANIZATION>参</ORGANIZATION>議長
<LOCATION>チェコスロバキア</LOCATION>
<LOCATION>南北朝鮮</LOCATION>
<LOCATION>中南米</LOCATION>

<ORGANIZATION>衆参議院</ORGANIZATION>

また、正式名称であると分る物はすべてまとめて抽出する。

<ORGANIZATION>在ナイジェリア日本大使館</ORGANIZATION>

3.3.5 省略形、ニックネーム

省略形、ニックネームは固有表現が含まれている場合はすべて抽出する。ただし、省略であっても一般名詞のみを使い照応として使われている場合は抽出しない。

<ORGANIZATION>ＮＲＡ</ORGANIZATION>
<ORGANIZATION>ホワイトハウス</ORGANIZATION>の発表によると
<ORGANIZATION>広島</ORGANIZATION>対<ORGANIZATION>巨人</ORGANIZATION>
<LOCATION>北朝鮮</LOCATION>
<PERSON>きょん２</PERSON>
<PERSON>ミッチー</PERSON>コール
<ORGANIZATION>明治</ORGANIZATION>　(明治生命と文脈で分る場合)
<ORGANIZATION>東武鉄道</ORGANIZATION>... 当鉄道は...
<PERSON>田中<PERSON>教授... 教授は...

3.3.6 入れ子

入れ子の場合は、一番外の固有表現のみを抽出する。

<ORGANIZATION>松下貿易（松貿）株式会社</ORGANIZATION>

以下のような場合には注意が必要である。

<ORGANIZATION>松下貿易株式会社</ORGANIZATION>（<ORGANIZATION>松貿</ORGANIZATION>）

3.3.7 特殊シンボル

かっこ、かぎかっこ、強調のためのシンボルは、それが固有表現の内部にあれば入れるが、外にある場合は入れない。

<PERSON>ビト・”ザ・ゴッドファーザー”・コーレオン</PERSON>
「<PERSON>橋竜</PERSON>」カラー
<ARTIFACT>「平和のためのパートナーシップ」（ＰＦＰ）協定</ARTIFACT>

3.3.8 漠然とした対象

固有名詞的表現であっても、宗教名、特定の対象を示さないグループ名、漠然と複数の組織を示す表現は抽出しない。(特例：組織名として表現されている宗教名はオプショナルとする)

民主リベラル新党
過激派
保守派
<LOCATION>セルビア</LOCATION>人勢力
ＪＲ各社
国営四現業
関係七組合
夏の収穫祭
警察
彼はイスラム原理主義者である。
<OPTIONAL>浄土真宗本願寺派<OPTIONAL>がビデオを発売した。
<OPTIONAL>オウム真理教<OPTIONAL>が訴えを起した。

但し、

<ORGANIZATION>警察庁</ORGANIZATION>
<ORGANIZATION>東京都警視庁</ORGANIZATION>

3.3.9 イベント名、事件名

イベント名は、いづれの固有表現にも当らないとし抽出しない。

<LOCATION>長野</LOCATION>オリンピック
<LOCATION>リオ</LOCATION>・カーニバル
<ORGANIZATION>リクルート</ORGANIZATION>事件発覚
<ORGANIZATION>ロッキード</ORGANIZATION>事件
通常<ORGANIZATION>国会</ORGANIZATION>
<LOCATION>湾岸</LOCATION>戦争

3.3.10 漢字の読み

漢字の名前等の読みが（）内等に記述されている場合、その読みについても抽出する。

<PERSON>有馬朗人</PERSON>氏（<PERSON>ありま・あきと</PERSON>）
<PERSON>京極純一</PERSON>氏（<PERSON>きょうごく・じゅんいち</PERSON>）
<PERSON>長尾眞</PERSON>（<PERSON>まこと</PERSON>）氏

3.3.11 接頭辞

「旧」「新」等の接頭辞は一般に、個体を区別するために付いているので、固有表現に含める。

<LOCATION>旧ユーゴスラビア</LOCATION>
<ORGANIZATION>新田中派</ORGANIZATION>

3.3.12 空想上の対象

空想上の対象でも、それが組織名、人物、地名、固有物名を現わすものであれば、タグ付けする。物語、本、小説、劇、芝居、テレビ、ラジオなどの場合でも同様である。ただし、空想上の対象でも、明かにそのような対象外であるもの (例えば、空想上の動物) はタグ付けしない。その中間である曖昧な対象は OPTIONAL とする。

<PERSON>アリス</PERSON>は不思議な国を旅行します。
<PERSON>寅さん</PERSON>が<LOCATION>富士見山</LOCATION>に出掛けました。
<ORGANIZATION>地球防衛軍</ORGANIZATION>の<PERON>南</PERSON>隊長
ミッキーマウス
みなしごハッチ
<OPTIONAL>サイボーグ００９</OPTIONAL>
<ARTIFACT>ギリシャ神話</ARTIFACT>の<OPTIONAL>ゼウス</OPTIONAL>

3.3.13 仮名

仮名を使用した表現は抽出しない。

仮名：田中太郎氏
仮名：A氏
少年Aは...
企業Bの発売した商品C

3.3.14　一般的な表現

一般的な表現を使用し、文脈を使用しないと特定の対象を認定できない表現は、それを固有名として抽出しない。

総務部
首都
<LOCATION>福岡</LOCATION>支店

ただし、組織名の連続表現で、上下関係の組織名が連続している時などには組み合わせて組織名とする。(３．１．Ｃを参照の事)

<ORGANIZATION>日本銀行福岡支店</ORGANIZATION>

また、慣用的にそのような一般的な表現であるが、それが特定の対象を指している場合や普通名詞の組み合わせなどで固有名を指している場合には抽出する。

<ORGANIZATION>国会</ORGANIZATION>
<ORGANIZATION>影の内閣</ORGANIZATION>

3.3.15　新聞の名前

新聞の名前は、「報じる」「インタビューした」というような動作の主体となっている場合には組織名とし、「に載っている」といった物として暑かわれている場合には固有物名とする。

<ORGANIZATION>毎日新聞</ORGANIZATION>が行なったアンケートによると、
１５日付けの<ARTIFACT>ニューヨークタイムズ</ARTIFACT>によると、

以降は、人名表現固有の規則についてのみ説明する。

3.4 人名抽出の際の規則

3.4.1 役職名、敬称

役職名、敬称などは人名に含めない。

<PERSON>長尾</PERSON>総長
<PERSON>田中</PERSON>教授
プロフェッサー・<PERSON>グリッシュマン</PERSON>
<PERSON>石川</PERSON>氏

役職名がそのまま特定の人を表わし、文脈的にもその表現が役職ではなく人を表わしているような場合はOPTIONALとする。襲名する名前については節3.4.4を参照の事。

<OPTIONAL>天皇</OPTIONAL>
<OPTIONAL>ローマ法王</OPTIONAL>
<OPTIONAL>ダライラマ</OPTIONAL>

ただし、一般に役職名として使用される表現で照応として使われている場合にはOPTIONALとしない。

首相
大統領

3.4.2 賞名

固有の賞などに名前が使用されていても、人名とはしない。賞は固有物名とする。

<ARTIFACT>ノーベル賞</ARTIFACT>
<ARTIFACT>芥川賞</ARTIFACT>

3.4.3 肩書が付いた慣用的表現

人名に肩書が付いて慣用的に表現されている人名はOPTIONALとする。慣用的がどうかの判断は正解作成者の判断とする。

<OPTIONAL>清少納言</OPTIONAL>
<OPTIONAL>紫式部</OPTIONAL>
<OPTIONAL>虞美人</OPTIONAL>
<PERSON>エリザベス</PERSON>女王

3.4.4 襲名する名前

襲名する名前は人名とする。また、固有のエンティティーを示す数字などの表現が付いた場合はそれも含めて人名とする。

<PERSON>木村庄之助</PERSON>
横綱<PERSON>若ノ花</PERSON>
<PERSON>第十四代木村庄之助</PERSON>
<PERSON>先代若ノ花</PERSON>
<PERSON>本因坊</PERSON>

第4章　ME法

各単語がどの各クラスにどのくらいの確率で計算できれば人名抽出が行なえる。この計算は機械的学習手法を利用すれば可能であるが、本研究ではデータスパースネス問題に効果的であることが知られているME法を利用する。

ME法は，事象 t と h が同時に出現する頻度 $O(t,h)$ を訓練データとして，条件付き確率 $P(t|h)$ を推定するアルゴリズムである．ME法では確率 $P(t|h)$ を以下の式によって計算する．

$$P(t|h) = \frac{\prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)}}{\sum_t \prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)}}$$

ここで $f_i(t,h)$ は素性と呼ばれる 1 か 0 を返す関数であり，F はその総数，α_i は素性パラメータである．ME法は各素性が既知データ中に現われた割合は未知データも含む全データ中においても変化しないという制約をおく．これは推定すべき確率分布 $P(t|h)$ による素性 f_i の期待値と，既知データにおける確率分布 $\tilde{P}(t,h)$ による素性 f_i の期待値が等しいことを示している．ME法はこの制約をできるだけ満たすように各素性パラメータ α_i を推定する．具体的に α_i は GIS (Generalized Iterative Scaling) アルゴリズムにより求められる．

4.1　ME法とは

統計的自然言語処理において、曖昧性解消のための統計情報を学習する際にしばしば問題となるのはデータスパースネス問題である。データスパースネス問題とは、統計情報を学習するためのデータ量が十分にない時に発生する問題である。例えば、ある品詞 l から 単語 w の生成確率 $p(x|l)$ を学習する際に、訓練データに一度も現れない単語xについては、その生成確率p(x—l)は0として学習される。したがって、単語の生成確率 $p(x|l)$ を基に曖昧性解消のためのスコアを計算する場合、正解となる解析結果に単語 x が含まれる時には、そのスコアは低く見積もられる。このように、訓練データに一度も現れない事象や低頻度でしか現れない事象に対しては、それらの事象に対する統計情報を正しく学習できないことがある。このようなデータスパースネス問題は、統計情報の学習に要するパラメータ

の数が多ければ多いほど、また統計情報の学習に用いるデータ量が少なければ少ないほど発生しやすい。

このような訓練データに一度も現れない事象や低頻度の事象に対して、その統計情報を推測するのがスムージングと呼ばれる手法である。例えば、先ほどの例においては、訓練データに一度も現れない単語 x に対してある低い出現頻度 γ を与えることにより、生成確率 $p(x|l)$ は 0 ではなく、小さい確率値を持つようになる。統計的自然言語処理においては、データスパースネス問題に対処するさまざまなスムージング手法が提案されている。

このようなスムージング手法の中でも、近年注目されているのが ME 法である。ME 法とは、訓練データから条件付確率確率 $P(t|h)$ を推定するアルゴリズムである。この ME 法は自然言語処理に最も適したアルゴリズムであると考えられ、実際に自然言語処理に応用した研究例も報告されている。

- 形態素解析に応用した研究
- 構文解析に応用した研究
- PP-atachment 問題に応用した研究
- 機械翻訳に応用した研究
- 文境界の認定に応用した研究
- bi-gram 確率からの n-gram 確率の推定に応用した研究
- 下位範疇化フレーム n-gram 確率の推定に応用した研究
- 対訳単語対の抽出に応用した研究

ME 法を自然言語処理に応用する際に問題となるのは、確率も出るの推定に要する計算量が非常に多いということである。このことは、語彙的従属関係を反映する確率モデルなど、推定パラメータ数の多い確率モデルを ME 法を用いて推定する際の障害となっている。語彙的従属関係は構文解析の曖昧性解消に大きく貢献する統計情報である。ところが、語彙的従属関係は単語の共起関係に関する統計情報であり、また単語の数は非常に多いため、推定パラメータの数も膨大になる。したがって、語彙的従属関係を現在利用可能な言語資源から直接学習することはほとんど不可能に近く、何らかのスムージングを行なわなければならない。このとき。語彙的従属関係の学習に、自然言語処理に適した特性を持つと考えられる ME 法を適用しようとしても、確率モデルの推定に非常に多くの計算量を要するために一般に困難である。これを行なうためには、ME 法による確率モデルの推定に要する計算量を抑制する必要がある。よって語彙的従属関係のようなパラメータ数の多い統計情報のスムージングに ME 法をを適用することができるように、ME 法による確率推定アルゴリズムを効率化しなければならない。

4.2 ME法

ME法とは、事象 t と h が同時に出現する頻度 $O(t,h)$ を訓練データとして、条件付確率 $P(t|h)$ で表される確率モデルを推定するアルゴリズムである。ここで、h は履歴事象 (history) と呼ばれ、条件付確率の前提となる事象である。一方 t は目標事象 (target) と呼ばれ、確率モデルが予測する事象である。以下、ME法の原理について簡単に説明する。

ME法では、すべての事象の組 (t,h) について、確率 $P(t|h)$ の値を式 (4.1) によって計算する。

$$P(t|h) = \frac{\prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)}}{\sum_t \prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)}} \tag{4.1}$$

ここで $f_i(t,h)$ は素性 (feature) と呼ばれ、目標事象、履歴事象の組に対して1または0を返す任意の関数であり、また F はこのような素性の集合である。一方 α_i は素性パラメータと呼ばれるものである。

次に、素性の期待値 $E(f_i)$、$\hat{E}(f_i)$ を式 (4.2)、(4.3) のように定義する。

$$E(f_i) = \sum_{t,h} P(t,h) f_i(t,h) \simeq \sum_{t,h} \hat{P}(h) P(t|h) f_i(t,h) \tag{4.2}$$

$$\hat{E} f_i = \sum_{t,h} \hat{P}(t,h) f_i(t,h) \tag{4.3}$$

$$\text{ただし、} \hat{P}(t,h) = \frac{O(t,h)}{\sum_{t,h} O(t,h)} \tag{4.4}$$

$$\hat{P}(t,h) = \sum_t \hat{P}(t,h) \tag{4.5}$$

$E(f_i)$ は、ME法によって推定された確率モデル $P(t|h)$ において、素性 f_i が事象 (t,h) に対して1を返す期待値を表している。一方 $\hat{E} f_i$ は、訓練データによって最尤推定された確率モデル $\hat{P}(t,h)$ において、素性 f_i が事象 (t,h) に ついて1を返す期待値を表している。言い換えれば、$E(f_i)$ とは素性 f_i が1を返す事象の推定された確率モデルにおける同時確率 $P(t,h)^2$ の総和であり、$\hat{E}(f_i))$ とは素性 f_i が1を返す事象の訓練データにおける同時確率 $\hat{P}(t,h)$ の総和である。

確率モデルの推定は、

- 素性に関する式 (4.6) の制約を満たしつつ、

$$\forall f_i \in \quad FE(f_i) = \hat{E}(f_i) \tag{4.6}$$

- $P(t|h)$ のエントロピー $H(P)$ が最大となるように、

$$H(P) = -\sum_h \hat{P}(h) \sum_t P(t|h) log P(t|h) \tag{4.7}$$

素性パラメータ α_i を反復推定することにより行なわれる。式(4.6)で表される素性に関する制約とは、訓練データにおける素性の期待値と確率モデルにおける素性の期待値が等しいと言う制約である。一方、式(4.7)に示す確率モデルのエントロピー HP が最大になるということは、推定される条件付確率 $P(t|h)$ が一様分布に近くなることを意味する。すなわち、ME法による確率モデルの推定とは、ある素性が1を返す事象集合についてはその各事象の同時確率 $P(t,h)$ の和を訓練データのそれに近付け、かつ確率分布が一様分布になるべく近くなるように、各素性のパラメータ α_i を推定することである。このような素性パラメータ α_i の反復推定を行なうアルゴリズムの一つがGISアルゴリズムである。

このことを具体例を用いて説明する。ME法によって推定される確率アルゴリズムとして、ある動詞 v のヲ格の格要素として名詞 n が現れる確率 $P(n|[s(v,を)])$ を考える。単端のため、推定する確率モデルの目標事象を名詞 n 、履歴事象を動詞 v とし、推定する確率モデルを式(4.8)のように記述する。

$$P(n|N[s(v,を)]) = P(n|v) \tag{4.8}$$

この確率モデルの推定に用いる素性の例を式(4.9)に挙げる。

$$ftr(n,v) = \begin{cases} 1 & if n \in C_{食物} \quad and\ v = 食べる \\ 0 & otherwise \end{cases} \tag{4.9}$$

ここで、$C_{食物}$ は"食物"を表す名詞のクラス名詞の意味クラスであり、"$n \in C_{食物}$"は n が意味クラス $C_{食物}$ に属することを意味する。

もし、訓練データにおいて「食べる」という動詞のヲ格として"食物"を表す名詞が頻繁に出現していれば、"食物"を表す名詞 $n \in C_{食物}$ の出現確率が高くなるように確率モデルも推定される。これに加えて、確率モデルはエントロピーが最大に、すなわち一様分布になるべく近くなるように推定される。このため、"食物"表す名詞の出現確率、及びそれ以外の名詞の出現確率はすべて等しくなるよう推定される。

式(4.9)の素性は「食べる」という特定の履歴事象の確率分布を求めるものであったが、複数の履歴事象の確率分布を決める式(4.10)ような素性も記述できる。

$$ftr(n,v) = \begin{cases} 1 & if n \in C_{食物} \quad and\ v \in C_{食べる} \\ 0 & otherwise \end{cases} \tag{4.10}$$

この素性は、"食べる"という行為を表す動詞の意味クラス $C_{食べる}$ に属する動詞 v が与えられた時に、そのヲ格に"食物"表す名詞の意味クラス $C_{食物}$ に属する名詞が出現する事象の確率和が訓練データに等しくなるように確率モデルを推定する働きをする。

従って、訓練データにおいて $C_{食べる}$ に属する動詞のヲ格に $C_{食物}$ に属する名詞が多く出現しているなら、それらの重なりあっている事象の確率も高くなるように確率モデルが推定される。

以上で述べたように、素性は事象の組 (t,h) に対して1または0を返す関数であり、訓練事象全体における部分集合を限定し、その部分集合に属する事象の推定された確率モデルにおける確率和が訓練データにおける確率和と一致しなければならないという制約として働く。このとき、素性が事象の組 (t,h) に対して1または0を返す条件を記述する際には、目標事象に対する条件と履歴事象に対する条件を独立に記述しなければならない。ただし、式(4.10)のように、両者の論理積を条件としても構わない。

ME法は、以下のような優れた特徴を持つ。

- 目標事象、履歴事象の両方を抽象化した素性を取り扱うことができる

  目標事象、履歴事象のどちらか片方を抽象化した素性(例えば素性(4.9))しか用いることができない場合には、確率モデルの推定パラメータの数が多くなり効率が悪い。例えば、式(4.10)のような素性を目標事象のみしか抽象化しない素性を用いて表すと以下のようになる。

$$\forall v' \in C_{\text{食べる}}\, ftr(n,v) = \begin{cases} 1 & if n \in C_{\text{食物}} \quad and\ v = \text{食べる} \\ 0 & otherwise \end{cases} \tag{4.11}$$

  すなわち、$C_{\text{食べる}}$ に属する v' の数だけ素性を用意しなければならないが、これは推定する素性パラメータ α_i の数が多くなるために効率が悪い。ところが、ME法においては、素性(4.10)のような目標事象・履歴事象の両方を抽象化する素性を用いることができるので、推定する素性パラメータの数も1つで良い。

- 素性が1を返す事象集合の大きさに制限がない

  これは、素性が1を返す事象の集合として任意の目標事象と履歴事象に対する条件を指定できるということである。例えば、$C_{\text{食物}}$ よりも抽象度の大きい $C_{\text{物体}}$ のような意味クラスを用いて、式(4.12)のような素性を F の要素としても構わない。

$$ftr(n,v) = \begin{cases} 1 & if n \in C_{\text{物体}} \quad and\ v = \text{作る} \\ 0 & otherwise \end{cases} \tag{4.12}$$

  ここで重要なのは、さまざまな大きさの事象集合に対して1を返す素性が混在しても構わないという点である。これにより、「食べる」という動詞については抽象化レベルの低い $C_{\text{食物}}$ という意味クラスを用いた素性(4.9)を使い、「作る」という動詞については抽象化レベルの高い $C_{\text{物体}}$ という意味クラスを用いた素性(4.12)を使うこともできる。このように、素性が1を返す事象集合の大きさを任意に変えることにより、より柔軟なスムージングが可能となる。

- 素性が1を返す事象の集合に重なりがあっても良い

  例えば、素性 f_a と f_b が1を返す事象集合に重なりがあっても、ME法はそれぞれの素性が式(4.6)の制約を満たすように確率モデルを推定することができる。このことは、自然言語処理への応用に適した特性の一つであると考えられる。なぜなら式(4.9)や素性(4.10)のように意味クラスを用いて素性を定義する場合、さまざまな視点による意味クラスを同時に利用することができるからである。例えば、「学校」という単語は、建造物という視点から見れば $C_{\text{建物}}$ という意味クラスに属し、組織という点から見れば $C_{\text{組織}}$ というクラスに属している。このようなさまざまな視点から見た意味クラスを同時に利用する場合には、それらの素性が1を返す事象集合に重なり(例えば、単語「学校」は両方の意味クラスに属する)ことができるが、ME法はこのような場合でも確率モデルを推定することが可能である。

  素性が1を返す事象集合に重なりがない場合には、例えば式(4.13)のような簡単な式を用いて確率モデルを推定することによって非常に少ない計算量でスムージングをおこなうことができる。

$$P(t,h) = \frac{\sum_{\forall t_i,h_j \ f(t_i,hj)=1} \hat{P}(t_i,h_j)}{N} \tag{4.13}$$

ただし、N は $f(t_i,h_j)=1$ となる事象の組 (t_i,h_j) の数

  式(4.13)は、素性 f が1を返す事象集合 (t_i,h_j) の訓練データにおける確率 $\hat{P}(t_i,h_j)$ の平均を $P(t,h)$ とすることを意味する。しかしながら、このようにして推定される確率モデルとしては、素性が1を返す事象集合に重なりがないという条件を満たさなければならないために比較的単純なものに限定される。これに対して、ME法はより一般的な確率モデル、すなわち素性が1を返す事象集合に重なりがあるような確率モデルを推定することができる。

このように、ME法は優れた特徴を持っているが、以下に述べるような問題も存在する。まず、第一に、確率モデルの推定に用いるGISアルゴリズムの計算量が非常に多いということである。これはME法を実際に自然言語処理に応用する際の障害となる。もうひとつは、確率モデルの推定に用いる素性集合 F をどのように決定すれば良いのかということである。本節で述べたME方の原理から推察できるように、推定される確率モデルの性質は素性集合 F に大きく依存する。この F を決定する方法としては、まず素性集合 F を決定する方法としては、まず素性集合の候補 S を作り、その中の1つの素性を採用した時の確率モデル全体のエントロピーの変化量を調べ、その変化量の最も大きい素性を確率モデルの推定に有効であると判断して F に加え、これを繰り返す素性選択アルゴリズムと呼ばれる方法が提案されている。しかしながら、この方法は、1つの素性を F に加え

る度に、すべての素性候補についてそれを素性として採用した時の確率モデル全体のエントロピーの変化量を計算し、またGISアルゴリズムによって確率モデルの推定をやり直すため、素性選択に要するする計算量が非常に多いという問題点がある。

4.3 GISアルゴリズムの効率化

本節では、訓練共起データ $O(t,h)$ と素性集合 F をもとに、各素性のパラメータ α_i を反復推定するGISアルゴリズムの概要と、それを効率化する手法について述べる。

GISアルゴリズムの概要を以下に示す。

GISアルゴリズム

1. 訓練データにおける素性の期待値 $\hat{E}f_i$ を計算する。
2. すべての素性パラメータの初期値 $\alpha_i^{(0)}$ を1とする。
3. 式(4.1)より、与えられたパラメータ $\alpha_i^{(n)}$ における $P^{(n)}(t|h)$ を計算する。
4. 式(4.2)より、確率モデルにおける素性の期待値 $E^{(n)}(f_i)$ を計算する。
5. α_i を以下のように更新する。

$$\alpha_i^{(n+1)} = \alpha_i^{(n)} \left[\frac{\hat{E}(f_i)}{E^{(n)}(f_i)} \right]^{\frac{1}{C}} \tag{4.14}$$

6. α_i が収束するまで3～5を繰り返す

式(4.14)における C は補間定数と呼ばれる定数である。GISアルゴリズムは、収束条件として、あらゆる目標事象・履歴事象の組 (t,h) に対して、各素性 f_i の返す値の和、すなわち (t,h) に対して1を返す素性の数は等しくなければならないという制約を持つ。

$$\forall t, \forall h \quad \sum_{i=1}^{|F|} f_i(t,h) = C \tag{4.15}$$

ところが、通常は式(4.15)の制約は満たされていないため、以下のような補間定数 C と補間素性 f_c を導入する。

$$C = \max_{t,h} \sum_{i=1}^{|F|} f_i(t,h) \tag{4.16}$$

$$f_c(t,h) = C - \sum_{i=1}^{|F|} f_i(t,h) \tag{4.17}$$

補間素性 f_c を素性集合 F に加えることにより式 (4.15) の制約式は満たされ、GIS アルゴリズムも必ず収束する。

GIS アルゴリズムの 1 回の反復に要する計算量は $O(O|T|H)$ である。ただし、T は目標事象の集合、H は履歴事象の集合、F は素性の集合である。例えば、節 4.2 において、動詞 v が与えられたときに、そのヲ格に現れる名詞 n の出現する確率を与える確率モデル $P(n|v)$ を例として挙げた。この確率モデルを GIS アルゴリズムを用いて推定する場合、日本語における動詞の数は約 10,000 ($= |H|$)、名詞の数は約 200,000($= |T|$) 程度であるので、1 回の反復推定に要する計算量でもかなり大きい。

次に、GIS アルゴリズムを効率化する 2 つの手法を以下に示す。

- ある履歴事象 h について 1 を返す素性の集合を F_h とする。履歴事象 h_1 と h_2 において、F_{h_1} と F_{h_2} が等しいなら、式 (4.1) より確率分布 $P(t|h_1)$ と $P(t|h_2)$ は全く同一となる。

$$\forall t \in T P(t|h_1) = P(t|h_2) \tag{4.18}$$

  したがって、GIS アルゴリズムの Step.3 の $P(t|h)$ の計算において、F_h が等しい履歴事象については、その中の 1 つの履歴事象 h_1 について $P(t|h_1)$ を計算すれば、その他の履歴事象については計算を省略できる。

- ある事象 (t_1, h) と (t_2, h) において、これらに対して 1 を返す素性の集合が全く同じならば、式 (4.19) に示すように、式 (4.1) の分子の項は全く同じになるはずである。

$$\prod_{i=1}^{|F|} \alpha_i^{f_i(t_1,h)} \prod_{i=1}^{|F|} \alpha_i^{f_i(t_2,h)} = \tag{4.19}$$

  したがって、GIS アルゴリズムの Step.3 の $P(t, h)$ の計算において、先ほどと同様に同じ値となる項の計算を省略することができる。

以上の手法により、計算量がどの程度削減されるかについては素性集合 F に依存する。一般に素性集合 F の要素数が少なければ少ないほど、計算量も大幅に縮小される。

4.4 素性選択アルゴリズムの効率化

P4.2 で述べたように、ME 法によって推定される確率モデルの品質は素正集合 F に大きく依存する。したがって素正集合 F をどのように決定するかは、ME 法

における最も重要な問題である。この F を決定する方法としては素性選択アルゴリズムと呼ばれるものが提案され、また実際にこれを応用した研究もいくつか報告されている。素性選択アルゴリズムとは、与えられた素性の候補の集合 S から確率モデルに採用するべき素性集合 F を選択するアルゴリズムである。これは、S の中から確率モデルのログ尤度 (log likelihood) を最も増大させる素性を1つ選択し、それを F に追加するといった操作を繰り返すことによって行なう。ログ尤度とは、式 (4.20) で定義され、エントロピーと同じく確率モデルが一様分布にどれだけ近いか（ログ尤度が低くなればなるほど確率モデルは一様分布に近付く）を表す指標である。

$$L(P) = \sum_{t,h} \hat{P}(t,h) log P(t|h) \tag{4.20}$$

素性選択アルゴリズムの詳細を以下にまとめる。

素性選択アルゴリズム

1. F を空集合とする。
2. 素性候補の集合 S の各要素 f_j について f_j を F に加えた時の f_j のパラメータ α_j を求める。これは GIS アルゴリズムにより反復推定する。
3. f_j F に加えた時のログ尤度の増分 $\Delta L(f_j)$ を計算する。$\Delta L(f_j)$ の最も大きい素性を1個選択し、それを確率モデルの素性として新たに F 追加するに
4. 新しい F における各素性のパラメータ α_i を GIS アルゴリズムを用いて再推定する。
5. 素性を加えた時の確率モデルのエントロピーの変化量がある閾値 T_{ent} 以下になるまで 2．～ 4．を繰り返す

この素性選択アルゴリズムは最小記述長原理 (Minimum Description Length Priciple、以下 MDL 原理) による素性選択とほぼ一致する。MDL 原理においては、確率モデルの良さを表す評価尺度としてモデル記述長とデータ記述長を用いる。モデル記述長は推定された確率モデルの複雑さを表し、データ記述長は推定された確率モデルと訓練データにおける確率モデルとの近さを表す。そして、このモデル記述長とデータ記述長の和が最小となるように、すなわちなるべく単純なモデルでかつ訓練データにおける確率モデルに近い確率分布を持つように、確率モデルに与える素性を決定する。素性選択アルゴリズムにおいて、素性を1個増やすことは確率モデルを複雑にし、MDL 原理におけるデータ記述長を増やすことに対応する。また、その時に確率モデルのログ尤度が大きくなることは、MDL 原理におけるデータ記述長を減らすことに対応する。したがって、素性を1個増やした時のログ尤度の増分が最大となる素性を選択することは、MDL 原理におけるモデル記述長とデータ記述長の和が最小となる確率モデルを選択することに等しい。

MDL 原理を自然言語処理に応用した例としては、動詞の格フレームを学習する

研究や単語のクラスタリングを行なった研究などがある。これらの研究においては、素性が1を返す事象集合に重なりがないことを前提にしている。これに対し、ME 法における素性選択アルゴリズムは、素性が1を返す事象の集合に重なりがあるような一般的なモデルにも適用することが可能である。この素性選択アルゴリズムは、すべての祖性候補について、それを素性として採用した時のログ尤度を求めるために素性パラメータを GIS アルゴリズムによって反復推定し、またログ尤度の増分が最も大きい素性を素性集合 F 加えるたびに GIS アルゴリズムによって確率モデル全体の推定をやり直すため、計算量が非常に多いという問題がある。そこで、素性選択アルゴリズムを効率化する4つの手法を提案する。

- 素性選択アルゴリズムの Step.2 において、素性候補の集合 S の中で $\Delta L(f_i)$ の値が最も大きい素性を f_1 、二番目に大きい素性を f_2 とする。このとき、まず f_1 のみを F に加えて確率モデルの推定をやり直してから (Step.4)、再び素性候補の集合 S の中の各素性のログ尤度の増分 (これを $\Delta L'(f_j)$ とする) を計算する。

  ここで、f_1, f_2 が条件 (4.21) を満たす場合を考える。

$$T_{f_i} \cap T_{f_2} = \emptyset \quad and H_{f_1} \cap H_{f_2} = \emptyset \tag{4.21}$$

  式 (4.21) において、T_f は素性 f が1を返す目標事象の集合であり、H_f は素性 f が1を返す履歴事象の集合である。ここでは、式 (4.21) の条件を満たす2つの素性 f_1 f_2 は互いに独立であると定義する。f_1 f_2 が互いに独立であることは、2つの素性が1を返す事象集合に重なりがないことを意味する。

  f_1 と f_2 が互いに独立な場合、すなわち全く異なる事象に対する確率分布を決定する素性である場合は、まず F f_1 を追加してから確率モデルを推定し、f_2 に対するログ尤度の増分 $\Delta L'(f_2)$ を計算しても、その値は $\Delta L(f_2)$ と変わらない。すなわち $\Delta L(f_2) \simeq \Delta L'(f_2)$ であると予想される。したがって、f_1 の次に追加される素性として f_2 が選ばれる可能性が高い、そこで F に素性を1個ずつ追加する代わりに、ΔL の値を大きく、かつ互いに独立である上位 N_f 個の素性を F に同時に追加するように、素性選択アルゴリズムを修正する。これにより、素性選択に要する時間を約 $1/N_f$ に短縮できる。

- 素性選択アルゴリズムの Step.2 においては、素性候補の集合 S に含まれるすべての要素 f_i について $\{\alpha_i\}], a_j$ のすべてのパラメータを GIS アルゴリズムを用いて再推定している。ここで、$\{\alpha_i\}$ はすでに選択された素性集合 F の中の各素性に対応したパラメータであり、a_j はこれから素性として選択するべき素性候補の集合 S の一つの要素 f_j に対応したパラメータをあらわす。ところが、これは GIS アルゴリズムによる反復推定に要する計算量が多

いために効率が悪い。そこで、素性パラメータ $\{\alpha_i\}$ の値は変化させず、a_j のみを再推定するように修正したGISアルゴリズムを適用することによって a_j の値を近似推定する。この修正されたGISアルゴリズムの詳細を以下に示す。

1. $\alpha_i^0 = 1$ とする。
2. $P^{(n)}(t|h)$ を計算する。

$$P^{(n)}(t|h) = \frac{\prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)} \times \alpha_j^{(n)f_j(t,h)}}{\sum_t \prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)} \times \alpha_j^{(n)f_j(t,h)}} \tag{4.22}$$

3. $E^{(n)}(f_j)$ を計算する。
4. α_j の値を更新する。

$$\alpha_j^{(n+1)} = \alpha_j^{(n)} \left[\frac{\hat{E}(f_j)}{E^{(n)}(f_j)} \right]^{\frac{1}{C}} \tag{4.23}$$

5. α_j の値が収束するまで繰り返す。

素性集合 F の要素としてすでに選択された素性に対応したパラメータ $\{\alpha_i\}$ については、式(4.23)に示した反復推定に要する計算量を省略することにより、パラメータ推定に要する計算量を大幅に削減することができる。

- α_j のみ推定するように修正したGISアルゴリズムにおいては、式(4.22)に示した $P^{(n)}(t|h)$ の計算は以下のようになる。

$$\begin{aligned} P^{(n)}(t|h) &= \frac{\prod_{i=1}^{|F|} \alpha_j^{f_i(t,h)} \times \alpha_j^{(n)f_j(t,h)}}{\sum_t \prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)} \times \alpha_j^{(n)f_j(t,h)}} \\ &= \frac{\prod_{i=1}^{|F|} \alpha_j^{f_i(t,h)} \times \alpha_j^{(n)f_j(t,h)}}{D(\overline{T}_{f_j}) + D(T_{f_j}) \times \alpha_j^{(n)}} \end{aligned} \tag{4.24}$$

$$D(T_{f_j}) = \sum_{t \in T_{f_j}} \prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)} \tag{4.25}$$

$$D(\overline{T}_{f_j}) = \sum_{t \in T_{f_j}} \prod_{i=1}^{|F|} \alpha_i^{f_i(t,h)} \tag{4.26}$$

ここで、T_{f_j} は f_j が1を返す目標事象の集合であり、$\overline{T}_{f_j}$ は f_j が1を返さない目標事象の集合である。すなわち、$T_{f_j} \cup \overline{T}_{f_j} = T$ である。$D(T_{f_j}), D(\overline{T}_{f_j})$ の値は α_j の値に全く依存しないので、これらは α_j の推定の第1回目の反復時のみに計算すれば、第2回目以降の反復時には計算を省略できる。

- S 中の 2 つの素性候補 f_1 と f_2 について、それらが 1 を返す目標事象の集合が等しい場合、$(T_{f_1} = T_{f_2})$ を考える。このとき f_1 、 f_2 のパラメータ推定において、式 (4.24) の項 $D(T_{f_j})$ 、$D(\overline{T}_{f_j})$ の 値は全く等しい。したがって Δ を計算する素性候補の順序を T_{f_j} によってそーとすることにより、一部の素性の候補については項 $D(T_{f_j})$ 、$D(\overline{T}_{f_j})$ の計算を完全に省略することができる。

第5章　人名抽出実験

5.1　人名抽出の概要

ここでは人名抽出が一体どのような流れで行なわれるかを以下に示す。

1. テキストへのタグつけ
   訓練元データを作成するために、テキスト中の人名表現にタグつけをする
2. 形態素解析
   形態素解析によってテキストを単語に分割すると同時に、各単語の品詞を得る。
3. 単語へのクラス割り当て
   1と2によって作成されたデータを照合し、各単語へクラスを割り当てる。
4. f_i：素性関数の定義
   ME法で必要なパラメータである素性関数を定義する。
5. $O(t,h)$：頻度データの作成
   実際の訓練データとなる頻度データを作成する。
6. α_i：素性パラメータの算出
   GISアルゴリズムにより素性パラメータを算出する。
7. ME法による $P(t|h)$ の算出
   ME法により条件付確率 $P(t|h)$ を算出し、各単語の各クラスに属する確率を算出する。
8. ビタビアルゴリズムによるクラス推定
   ビタビアルゴリズムによりクラス列を推定する。

5.2　テキストへのタグつけ

まず訓練元のデータを作成するために、テキスト中の人名表現にタグつけを行なう。このタグの形式にはSGML形式でタグ付けをした。図5.2にタグ付けしたテキストの一部を示す。

今回使用したテキストには毎日新聞1995年1週間分(約4000行)を使用した。タグ付けにはP14の固有表現抽出の規則に従い、人名表現にタグつけを行なった。

「私の大切にしたいのは／その国の大きさでも繁栄でもない／その国はごく小さくていいし／すこしは武器らしいものを持つが／誰（だれ）も使おうとしない」
あれあれ、これからの日本のあるべき姿かなと思ったら、古代中国の哲学者、<PERSON>老子</PERSON>の言葉だった。「そこに住む人はみんな／生きることと死ぬことを大切にするから／船や車で遠くとびだしたりしない」というくだりもある
詩人、<PERSON>加島祥造</PERSON>さんの訳した「<PERSON>老子</PERSON>」の一節だ。数年前、旅行先で何気なく英訳本を手にしたのが老子との出会いという。加島さんは信州・伊那谷の山荘にこもり、十数冊の英訳をもとに翻訳を進めた。古めかしい「老子」が驚くほど若返った。今日的になった「あくまでも頑張る軍隊は全滅する／木も、堅く突立ったものは風に折れる／しなやかで、柔らかで／弱くて繊細なものこそ／上に位置を占めて／花を咲かせるべきなのだ」という文章もある。日本という国も少々突っ張りすぎた

図 5.1: タグ付きテキスト

5.3 形態素解析とクラス設定

先ほどの人名表現にタグ付けしたテキストからタグを取り去り、形態素解析し、単語に分割する。形態素解析にはJUMANを使用した。

この形態素解析したデータとSGMLタグ付のデータと照合することにより、形態素解析によって分割された各単語にクラスを割り当てる。以下に各単語にクラスを割り当てられたテキストを示す。人名表現にはPERSON、その他の表現にはNONEクラスが割り当てられている。この段階ではクラスの細分化は行なっていない。図5.2に各単語にクラスを割り当てたデータの一部を示す。

本研究ではタグ付のテキストからタグを取り去り、形態素解析されたテキストと照合することによってクラスの割り当てを行なうプログラムを作成した。これによってSGML付きのテキストを訓練元のデータとして利用できるようになった。

次に、図5.2のデータから、前後の単語情報と人名表現に関すると思われる品詞の分類とクラスの細分化を行なう。

本研究ではME法の履歴事象として単語とその文脈データを使用するため、前後の単語情報と品詞情報が必要である。表5.1に作成されたデータを示す。

図5.2のデータを調べてみると、人名表現の前、それ自身、後の単語の品詞には、名詞、未定義語、特殊、接頭辞、接尾辞の品詞が多く使用されていることがわかった。さらに区別するために上記の5品詞を分類した。表5.2に各品詞の分類したものを示す。

NONE 中国 ちゅうごく 中国 名詞 6 普通名詞 1 * 0 * 0
NONE の の の 助詞 9 接続助詞 3 * 0 * 0
NONE 哲学者 てつがくしゃ 哲学者 名詞 6 普通名詞 1 * 0 * 0
NONE 、 、 、 特殊 1 読点 2 * 0 * 0
PERSON 老 ろう 老 名詞 6 普通名詞 1 * 0 * 0
PERSON 子 こ 子 名詞 6 普通名詞 1 * 0 * 0
NONE の の の 助詞 9 接続助詞 3 * 0 * 0
NONE 言葉 ことば 言葉 名詞 6 普通名詞 1 * 0 * 0
NONE だった だった だ 判定詞 4 * 0 判定詞 25 ダ列タ形 8
NONE 。 。 。 特殊 1 句点 1 * 0 * 0

図 5.2: 各単語へのクラス割り当て

表 5.1: 品詞分類とクラス細分化

前の原型	前の品詞	現在の原型	次の原型	次の原型	次の品詞	クラス
古代	名1	中国	名1	の	助詞	NM
中国	名1	の	助詞	哲学者	名1	NM
の	助詞	哲学者	名1	、	特1	NM
哲学者	名1	、	特1	老	名1	NE
、	特1	老	名1	子	名1	PS
老	名1	子	名1	の	助詞	PE
子	名1	の	助詞	言葉	名1	NS
の	助詞	言葉	名1	だ	判定詞	NM
言葉	名1	だ	判定詞	。	特1	NM
だ	判定詞	。	特1	「	特1	NM

表 5.2: 品詞の小分類

名詞				
人名	普通名詞	地名	数詞	それ以外のもの
名1	名2	名3	名4	名0

未定義語		
カタカナ	その他	それ以外のもの
未1	未2	未0

特殊	
記号	それ以外のもの
特1	特0

接頭辞	
名詞接頭辞	それ以外のもの
接頭1	接頭0

接尾辞	
名詞性名詞接尾辞	それ以外のもの
接尾1	接尾0

表 5.3: 素性関数

f_i	前の原型	前の品詞	今の原型	今の原型	次の原型	次の品詞	クラス
f_1	*	*	*	特1	*	*	NE
f_2	*	*	*	*	*	名2	NE
f_3	*	名2	*	*	*	*	NI
f_4	*	*	*	*	*	特1	NI
f_5	*	*	*	*	*	名1	NM
f_6	*	*	*	名1	*	*	NM
f_7	*	*	*	*	*	名1	NS
f_8	*	名2	*	*	*	*	NS
f_9	*	名1	*	*	*	*	PE
f_{10}	*	*	*	*	*	特1	PE
f_{11}	*	特1	*	*	*	*	PI
f_{12}	*	*	*	名2	*	*	PI
f_{13}	*	*	*	名1	*	*	PM
f_{14}	*	*	*	*	*	名1	PM
f_{15}	*	特1	*	*	*	*	PS
f_{16}	*	*	*	名2	*	*	PS

5.4 f_i:素性関数

本研究では16種類の素性関数を定義した。
表5.3に素性関数の一覧を示す。
例として一番目の素性関数 f_1 を見てみる。

$$f_1 = \begin{cases} 1 & if \text{ 今の単語の品詞} = \text{特1} \quad and \text{ クラス} = NE \\ 0 & otherwise \end{cases}$$

これは、前の単語の原型が任意、前の単語の品詞が任意、今の単語の原型が任意、前の単語の品詞が特1、次の単語の原型が任意、次の単語の品詞が任意で、かつクラスがNEであるとき、1を返す、という意味である。表5.3はこれらを表形式にしたものである。
これにより、訓練データとなる頻度データ $O(t,h)$ を求めることができる。

図 5.3: テスト文

「不動産情報・登記・税制・評価システム協議会」が発足－－ネットで情報を無償提供

◇不動産情報、参考価格や抵当権者も－－今月中旬、実現へ

これまで不透明と指摘されてきた不動産情報を、官民が協力してインターネットで無償

提供する「不動産情報・登記・税制・評価システム協議会」（会長、寺村信行・元国税庁

長官）が今月中旬に発足することになった。全国の弁護士、不動産鑑定士、一級建築士、

公認会計士、税理士など不動産に関する業務を行う９業種の専門家（総数１０４万人）を

順次、会員登録し、協議会が妥当と判断した参考価格を提供する。最終的に年間４０万件

といわれる不動産取引情報すべてを対象とする予定だ。

5.5 α_i:素性パラメータの算出

求めた頻度データ $O(t,h)$ から、GIS アルゴリズムによって素性パラメータを算出する。

本研究では訓練用のテキストから、頻度データを求め、素性パラメータを算出した。以下に α_i を示す。

α_i = { 7.79, 7.23, 0.53,1, 385.70, 338.62, 5.29, 5.26, 3.28, 2.84, 4.86, 3.02, 1.63, 1.52, 3.92, 3.53 }

5.6 $P(t|h)i$ の算出と人名抽出

以上より，単語 a のクラスが C となる確率 $P(C|a)$ が求まる．テスト文 111 文に対する実験の結果，表 5.4 のような結果を得た．テスト文の一部を図 5.3 に示す。

5.7 ビタビアルゴリズムによるクラス推定

入力文の各単語にクラスを付与するが，一意にクラスを付与した場合，現実には不可能なクラス付け生じることもある．たとえば，ある単語が人名の始まりのクラス (PS) を与えられ，次の単語が人名以外の終わるクラス (NE) を与えられた

表 5.4: 各クラスに属する確率

単語	NS	NM	NE	NI	PS	PM	PE	PI
「	0.286955	0.286883	0.286838	0.027865	0.027865	0.027865	0.027865	0.027865
不動産	0.041881	0.431081	0.004067	0.004067	0.041892	0.431070	0.004067	0.041875
情報	0.007600	0.078243	0.007600	0.007600	0.007600	0.805535	0.078224	0.007600
・	0.286961	0.286889	0.027865	0.027865	0.027865	0.027865	0.286823	0.027865
登記	0.008177	0.084189	0.008177	0.008177	0.008177	0.866748	0.008177	0.008177
・	0.286961	0.286889	0.027865	0.027865	0.027865	0.027865	0.286823	0.027865
税制	0.008177	0.084189	0.008177	0.008177	0.008177	0.866748	0.008177	0.008177
・	0.286961	0.286889	0.027865	0.027865	0.027865	0.027865	0.286823	0.027865
評価	0.008177	0.084189	0.008177	0.008177	0.008177	0.866748	0.008177	0.008177
システム	0.286961	0.286889	0.027865	0.027865	0.027865	0.027865	0.286823	0.027865
協議会	0.007600	0.078243	0.007600	0.007600	0.007600	0.805534	0.078225	0.007600
」	0.037612	0.037612	0.387178	0.037612	0.037612	0.037612	0.387149	0.037612
が	0.227878	0.227821	0.022128	0.022128	0.227941	0.022128	0.022128	0.227848
発足	0.008177	0.084189	0.008177	0.008177	0.008177	0.866748	0.008177	0.008177
ー	0.057826	0.057826	0.057826	0.057826	0.057826	0.057826	0.595215	0.057826
ー	0.125000	0.125000	0.125000	0.125000	0.125000	0.125000	0.125000	0.125000
ネット	0.125000	0.125000	0.125000	0.125000	0.125000	0.125000	0.125000	0.125000
で	0.387240	0.387143	0.037603	0.037603	0.037603	0.037603	0.037603	0.037603
情報	0.008177	0.084189	0.008177	0.008177	0.008177	0.866748	0.008177	0.008177
を	0.286961	0.286889	0.027865	0.027865	0.027865	0.027865	0.286823	0.027865
無償	0.045307	0.466353	0.004400	0.004400	0.004400	0.466341	0.004400	0.004400
提供	0.007600	0.078243	0.007600	0.007600	0.007600	0.805535	0.078224	0.007600

表 5.5: 人名の自動抽出の結果

人名の数	144 件
人名と判断したもの	406 件
抽出に成功した数	17 件

としたら，そのクラス列に対する人名抽出は不可能である．

このような事態を避けるために，入力文中の単語 a に対して，a がクラス C に属する確率 $P(C|a)$ を求めることにする．

入力文が $a_1a_2\cdots a_n$ （各 a_iは単語）の場合，ME 法により，$P(C|a_j)$ が求まる．単語 a_j に与えるクラスが C_j とすると，人名抽出は，クラス C_j と C_{j+1} が接続可能という条件のもとで，以下の値が最も大きくなるような，C_j の列を求めることに対応する．

$$\sum_{j=1}^{n} P(C_j|a_j)$$

これは一般に Viterbi アルゴリズムによって求めることができる．

5.8 実験結果

表5.4のデータから、ビタビアルゴリズムを用いてクラス列を一意に付与し、人名部分に対応する単語列を抽出する。図5.4にクラスを割り当てたデータの一部を示す。

ここで、単語のとなりにある数字は、1から8まで順にクラスNS(無関係の始まり)、クラスNM(無関係の途中)、クラスNE(無関係の終わり)、クラスNI(それ自身が無関係)、クラスPS(人名の始まり)、クラスPM(人名の途中)、クラスPE(人名の終わり)、クラスPI(それ自身が人名)というようになっている。

人名抽出実験の結果を表5.5に示す。表をみればわかるように，ほとんど人名表現を抽出できていないことがわかる。図5.8に抽出に成功した個所の一部を示す。

「 1
不動産 2
情報 2
・ 2
登記 2
・ 2
税制 2
・ 2
評価 2
システム 2
協議会 2
」 2
が 2
発足 2
ー 2
ー 2
ネット 2
で 2
情報 2
を 2
無償 2
提供 3

4
これ 5
まで 7
不透明だ 1
と 2
指摘 2
する 2
れる 2
くる 2
不動産 2
情報 2
を 2
、 2
官民 2
が 2
協力 2

図 5.4: ビタビアルゴリズムによるクラス割り当て

評価　2
システム　2
協議会　2
」　2
（　2
会長　2
、　3
寺村　5
信行　7
・　1
元　2
国税庁　2
長官　2
）　2
が　2
今月　2
中旬　2
に　2
発足　2
する　2
こと　2
に　2
なる　2

図 5.5: 抽出成功例

第6章　考察

- 失敗の原因

  P4.2にあるように、推定される確率モデルの性質は素性集合 F に大きく依存する。今回の実験では素性集合を頻度によって決めたが、良い結果を得ることはできなかった。これは素性集合がクラスを特徴づける素性の集合として適切でなかった。

  そしてGISアルゴリズムによる素性パラメータの算出には予想以上の時間がかかった。これにより素性関数がクラスを特徴づける素性の集合として適切であるかどうかの判断が難しいことも原因の一つであると考えられる。

  素性集合 F をどのように決定するかは、ME法における最も重要な問題である。この素性集合を決定する方法として素性選択アルゴリズムがあるが、本研究においては未実装である。素性選択アルゴリズムも結局はGISアルゴリズムに依存しているため、計算量が非常に大きくなってしまうからである。

  以上の理由により今回、ME法を利用した人名抽出は良い結果を残すことができなかった。

- 今後の課題今後の課題としては、クラスを特徴づける素性関数を定義することである。しかし、素性が適切であるかどうかというのは実際にGISアルゴリズムを使用し確率モデルの推定を行うまでは判断が難しい。場合によってはGISアルゴリズムに代わるアルゴリズムを採用することも考慮しなければならない。

第7章　結論

本研究ではME法を利用した人名表現抽出を行った。しかし、ME法が人名表現抽出に有効かであるかは検証することができなかった。

次の課題は、素性関数を決定し直し、素性選択アルゴリズムを実装することによって抽出結果の向上を図ることである。

謝辞

本研究の遂行及び論文の作成において多大な御助言及び御指導を賜った新納 浩幸教官 (茨城大学工学部システム工学科) に深い感謝の意を表します。

関連図書

[1] 関根聡：“テキストからの情報抽出 -文書からの特定の情報を抜き出す-” “IPSJ Magazine Vol.40 No.4 Apr.1999”

[2] IREX Homepage: http://cs.nyu.edu/cs/proteus/irex/

[3] 新納浩幸：“拡張文字ベースの HMM を利用した固有名詞抽出” IREX ワークショップ予稿集,1999

[4] Satoshi Sekine,Ralph Grishman,and Hiroyuki Shinnnou : “A Decision Tree Method for finding and classifying Names in Japa nese Texts”

付 録A　クラス割り当てプログラム

```
/* findtag.h */

#ifndef __FINDTAG_H__
#define __FINDTAG_H__


/* include files */
#include <stdio.h>
#include <string.h>
#include <assert.h>


typedef struct tag_taga_t {
  char *p; /* 開始タグの文字位置 */
  char *e; /* 終了タグの文字位置 */
  /*  Organization,Person,Location,Artifact,Date,Time,percent */
  /*  のいずれかが入る。ただしすべて大文字 */
  char *attrib;

} taga_t;

/* prototype function */
void findtag(const char *, const char *);
char *cpinst(char *,char *);
int tagcount(char *);
int tagsort(taga_t *);

/* global */
#define MAX_TAGARR_SIZR 100
int tagarr_size = 0;
taga_t tagarr[MAX_TAGARR_SIZR];
```

```
#endif /* __FINDTAG_H__ */

/* -------------- findtag.c ---------------------  */
/* 形態素解析して分割された単語に PERSON タグをつける */
/* ---------------------------------------------- */

#include "findtag.h"

int main(int argc, char *argv[])
{
  FILE *key_file, *juman_file;
  char key_file_buf[4048], juman_file_buf[1024], out_buf[1024],class[30];
  char *p, *str, *key_buf_pos, *token,*token2,*token3,*token4, *key_buf_end_pos, *
  long filesize;
  int array_pos, tag_num, tagst=0, len, i;
  int flag_tagnon = 0;

  if(NULL == (key_file = fopen(argv[1],"rt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }
  if(NULL == (juman_file = fopen(argv[2],"rt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }

  while(NULL != fgets(key_file_buf, 4048, key_file)){
    flag_tagnon = 1;
    tag_num = tagcount(key_file_buf);
    if(tag_num != 0){
      tagarr_size = 0;/* 格納される場所をクリア */
      findtag(key_file_buf, "ORGANIZAION");
      findtag(key_file_buf, "PERSON");
      findtag(key_file_buf, "LOCATION");
      findtag(key_file_buf, "ARTIFACT");
      findtag(key_file_buf, "ORGANIZATION");
      findtag(key_file_buf, "DATE");
      findtag(key_file_buf, "TIME");
```

```
        findtag(key_file_buf, "PERCENT");
        findtag(key_file_buf, "MONEY");
#ifdef DEBUG
        if(tag_num != tagarr_size) {
  fprintf(stderr, "見つけたタグの数と格納したタグの数が一致しません. \n");
  abort();
        }
#endif /* DEBUG */
        tagsort(tagarr);
        flag_tagnon = 0;
      }

      /*    printf("%d-%s", tag_num,key_file_buf);*/

      array_pos = 0;
      key_buf_pos = key_file_buf;
      key_buf_end_pos = key_buf_pos + strlen(key_file_buf);

      while(key_buf_pos <= key_buf_end_pos){
        if(NULL == fgets(juman_file_buf, 1024, juman_file))
  break;

        len=0;/* タグつけ個所の長さをクリア */

        strcpy(out_buf,juman_file_buf);
        token = strtok(juman_file_buf, " ");

        if(token == NULL) {
  fprintf(stderr, "can't get juman token\n");
  abort();
        }

        if(tag_num == 0 || flag_tagnon) {
  strcpy(class, "NONE");
  if((strcmp(token, "EOS")) != 0) { /* EOSの場合は何もしない */
    key_buf_pos += strlen(token);
    /*    printf("%s %s\n", class, token);*/
    printf("%s %s", class, out_buf);
  }
```

```
else
  printf("%s %s\n", class, token);
continue;
    }

    if(array_pos >= tagarr_size) {
fprintf(stderr, "overflow at tagarr accsess.\n");
fprintf(stderr, "tagarr size is %d.\n", tagarr_size);
fprintf(stderr, "accsess value is %d.\n", array_pos);
abort();
    }


    /* タグ処理 */

    /* case 開始タグ */
    if(tagarr[array_pos].p == key_buf_pos){
key_buf_pos = strstr(key_buf_pos,">") + 1;
strcpy(class,tagarr[array_pos].attrib);
tagst++;

/* タグつけ個所が token より短い? */
tmp=key_buf_pos;
while(tmp++ != tagarr[array_pos].e) /* タグつけ個所の長さ */
  len++;
if(strlen(token) > len){
  key_buf_pos=strstr(tagarr[array_pos].e,">")+1-len;
  sprintf(class,"%s",tagarr[array_pos].attrib);
  /*    sprintf(class,"%s-fixed-",tagarr[array_pos].attrib);*/
  tagst=0;
  if(++array_pos >= tagarr_size) { /* これ以上タグがない場合は
              タグの処理をかまさない */
    array_pos = tagarr_size - 1;
    flag_tagnon = 1;
  }

}

    }
```

```
    /* case 終了タグ */
    else if(tagarr[array_pos].e == key_buf_pos){
key_buf_pos = strstr(key_buf_pos,">") + 1;
/* strcpy(class,tagarr[i].attrib); */
strcpy(class, "NONE");
tagst = 0;
if(++array_pos >= tagarr_size) { /* これ以上タグがない場合は
            タグの処理をかまさない */
  array_pos = tagarr_size - 1;
  flag_tagnon = 1;
}
else if(key_buf_pos == tagarr[array_pos].p){
  /* これでは範囲外アクセスだおー */
  key_buf_pos = strstr(key_buf_pos,">") + 1;
  strcpy(class,tagarr[array_pos].attrib);
  tagst++;
}
    }


    /* タグ開始チェックに洩れてタグ領域にアクセスしてしまった場合 */
    else if(tagst == 0 && key_buf_pos > tagarr[array_pos].p &&
            key_buf_pos < tagarr[array_pos].e){

key_buf_pos = strstr(tagarr[array_pos].p,">") + 1;
if((key_buf_pos=strstr(key_buf_pos,token)) > tagarr[array_pos].e){
  sprintf(class,"%s",tagarr[array_pos].attrib);
  /*    sprintf(class,"Tyep1A:%s",tagarr[array_pos].attrib); */
  array_pos++;
}
else{
  tagst++;
  sprintf(class,"%s",tagarr[array_pos].attrib);
  /*    sprintf(class,"Tyep1B:%s",tagarr[array_pos].attrib);*/
}
    }

    /* 行き過ぎてタグの外へいってしまった場合 */
    else if(tagst >= 1 && key_buf_pos > tagarr[array_pos].e){
```

```
    strcpy(class,"Type2");
    tagst = 0;
    key_buf_pos = strstr(key_buf_pos,token);

    if(++array_pos >= tagarr_size) { /* これ以上タグがない場合は
                タグの処理をかまさない */
      array_pos = tagarr_size - 1;
      flag_tagnon = 1;
    }
        }

        else{
    if(tagst == 0)
      strcpy(class, "NONE");
        }

        key_buf_pos += strlen(token);
        printf("%s %s", class, out_buf);

      }  /* while(key_buf_pos <= key_buf_end_pos){ */
    }/* while(NULL != fgets(key_file_buf, 2024, key_file)){ */

    fclose(key_file);
    fclose(juman_file);
    return 0;
}


void findtag(const char *string, const char *key)
{
    char stag[20],etag[20];
    char *pos,*str, *epos;

#ifdef DEBUG
    assert(string != NULL);
    assert(key != NULL);
    /* fprintf(stderr, "In findtag...\n");
       fprintf(stderr, "string:%s\n", string);
       fprintf(stderr, "key:%s\n", key); */
```

```
#endif /* DEBUG */

  str = string;
  sprintf(stag,"<%s>",key);
  sprintf(etag,"</%s>",key);

  while((pos=strstr(str,stag)) != NULL){
    /*printf("tag-[%s] found at %p\n",key,pos);*/
#ifdef DEBUG
    if(tagarr_size == MAX_TAGARR_SIZR) {
      fprintf(stderr, "tagarr overflow\n");
      abort();
    }
#endif /* DEUBG */
    tagarr[tagarr_size].attrib = key;
    tagarr[tagarr_size].p = pos;
    epos = strstr(pos, etag);
    tagarr[tagarr_size].e = epos;
    str = epos;
    ++tagarr_size;
  }

} /* findtag */


int tagsort(taga_t *tagp)
{
  int i, j;
  taga_t tmp;

  for(i = 0; i < tagarr_size; i++){
    for(j = i+1; j < tagarr_size; j++){
      if(tagarr[i].p > tagarr[j].p){
  tmp.p=tagarr[i].p;
  tmp.attrib=tagarr[i].attrib;
  tmp.e=tagarr[i].e;
  tagarr[i].p=tagarr[j].p;
  tagarr[i].attrib=tagarr[j].attrib;
  tagarr[i].e=tagarr[j].e;
```

```
  tagarr[j].p=tmp.p;
  tagarr[j].attrib=tmp.attrib;
  tagarr[j].e=tmp.e;
      }
    }
  }
  return 0;
}

int tagcount(char *strings)
{
  int num=0;
  char *p;
  p=strings;
  while(NULL != (p=strstr(p,"<ORGANIZATION>"))){
    p+=strlen("<ORGANIZATION>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<PERSON>"))){
    p+=strlen("<PERSON>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<LOCATION>"))){
    p+=strlen("<LOCATION>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<ARTIFACT>"))){
    p+=strlen("<ARTIFACT>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<DATE>"))){
    p+=strlen("<DATE>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<TIME>"))){
    p+=strlen("<TIME>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<PERCENT>"))){
```

```
    p+=strlen("<PERCENT>");num++;
  }
  p=strings;
  while(NULL != (p=strstr(p,"<MONEY>"))){
    p+=strlen("<MONEY>");num++;
  }

  return num;
}
```

付 録B　クラス割り当て・品詞分類プログラム

```
/* class.h */

#ifndef __CLASS_H__
#define __CLASS_H__

#include <stdio.h>
#include <string.h>

typedef struct hinshi_hinshi_t {
  char prev_org[64]; /* 前の単語の現形 */
  char prev_type[16]; /* 前の単語の品詞 */
  char curr_org[64]; /* その単語の現形 */
  char curr_type[16]; /* その単語の品詞 */
  char next_org[64]; /* 次の単語の現形 */
  char next_type[16]; /*  次の単語の品詞*/
} hinshi_t;

int check_detail(char *);
char *detail_n(char *);
char *detail_u(char *);
char *detail_s(char *);
char *detail_h(char *);
char *detail_t(char *);

#endif /* __CLASS_H__ */

/* -------------- class.c --------------------------- */
/* クラスの細分類と品詞の細分類し前後の単語を出力する */
/* ------------------------------------------------- */
```

```
#include "class.h"

int main(int argc, char *argv[])
{
  FILE *in_file;
  long count=0;
  int detail_id=0;
  char buf1[128],buf2[128],key[4],class[2][10], detail[64];
  hinshi_t line;

  /* 前の単語の原型と品詞を設定(最初) */
  strcpy(line.prev_org,"S");
  strcpy(line.prev_type,"S");

  if(NULL == (in_file = fopen(argv[1],"rt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }

  printf("P-Org,P-Type,C-Org,C-Typ,tN-Org,N-Type,Class\n");
  if(NULL == fgets(buf1,128,in_file)){ /* 最初に読んでおく */
    printf("error");
    return 0;
  }

  while(NULL != fgets(buf2,128,in_file)){
    /* 現在のクラス,現形,品詞を変数にいれる */
    sscanf(buf1,"%s%*s%*s%s%s%*s%s",class[0],line.curr_org,
           line.curr_type,detail);

    /* 人名に関わるものかどうかチェックし細分類 */
    switch (detail_id= check_detail(line.curr_type)){
    case 1:strcpy(line.curr_type , detail_n(detail));break;
    case 2:strcpy(line.curr_type , detail_u(detail));break;
    case 3:strcpy(line.curr_type , detail_s(detail));break;
    case 4:strcpy(line.curr_type , detail_h(detail));break;
    case 5:strcpy(line.curr_type , detail_t(detail));break;
    }
```

```
  /* クラスの最初の文字を抜き出す */
  key[0]=class[0][0];

  if(NULL != strstr(buf2,"EOS")){ /* EOSはスキップ */
    if(NULL == fgets(buf2,128,in_file)){ /* EOFだったら */
strcpy(line.next_org,"E"); /* 次の単語の現形と品詞に'E'を設定 */
strcpy(line.next_type,"E");
if(count > 1) /* クラスの細分化 */
  key[1]='E';
else
  key[1]='I';
key[2]='\0';
printf("%s\t%s\t%s\t%s\t%s\t%s\t%s\n",line.prev_org,line.prev_type,
        line.curr_org,line.curr_type,line.next_org,line.next_type,key);
break;
    }
  }

  sscanf(buf2,"%s%*s%*s%s%s%*s%s",class[1],line.next_org,
              line.next_type,detail);
  switch (detail_id = check_detail(line.next_type)){
  case 1:strcpy(line.next_type , detail_n(detail));break;
  case 2:strcpy(line.next_type , detail_u(detail));break;
  case 3:strcpy(line.next_type , detail_s(detail));break;
  case 4:strcpy(line.curr_type , detail_h(detail));break;
  case 5:strcpy(line.curr_type , detail_t(detail));break;
  }
  /* クラス細分化:現在のクラスが次のクラスが等しければ */
  if(strcmp(class[0],class[1]) == 0){
    count++;
    if(count == 1) /* はじめて同じになった場合 */
key[1]='S'; /* クラスXの始まりXS */
    if(count > 1 )/* 連続して同じだった場合 */
key[1]='M'; /* クラスXの途中XM */
  }
  else{
    if(count == 0)
key[1]='I'; /* クラスX自身 XI */
    if(count >= 1)
```

```
    key[1]='E'; /* クラスXの終りXE */
        count = 0;
      }
      key[2]='\0';
      printf("%s\t%s\t%s\t%s\t%s\t%s\t%s\n",line.prev_org,
              line.prev_type,line.curr_org,line.curr_type,
              line.next_org,line.next_type,key);

      /* 現在の行,単語の現形,品詞を前の行,単語の現形,品詞にする */
      strcpy(buf1,buf2);
      strcpy(class[0],class[1]);
      strcpy(line.prev_org,line.curr_org);
      strcpy(line.prev_type,line.curr_type);
    }
    fclose(in_file);
    return 0;
}

int check_detail(char *hinsi)
{
    if(strcmp("名詞",hinsi) == 0)
      return 1;
    else if(strcmp("未定義語",hinsi) == 0)
      return 2;
    else if(strcmp("特殊",hinsi) == 0)
      return 3;
    else if(strcmp("接頭辞",hinsi) == 0)
      return 4;
    else if(strcmp("接尾辞",hinsi) == 0)
      return 5;
    else return 0;
}

char *detail_n(char *hinsi)
{
    char *dest;
    if(strcmp("人名",hinsi))
      strcpy(dest,"名1");
    else if(strcmp("普通名詞",hinsi))
```

```
    strcpy(dest,"名2");
  else if(strcmp("地名",hinsi))
    strcpy(dest,"名3");
  else if(strcmp("数詞",hinsi))
    strcpy(dest,"名4");
  else
    strcpy(dest,"名0");
  return dest;
}

char *detail_u(char *hinsi)
{
  char *dest;
  if(strcmp("カタカナ",hinsi))
    strcpy(dest,"未1");
  else if(strcmp("その他",hinsi))
    strcpy(dest,"未2");
  else
    strcpy(dest,"未0");
  return dest;
}

char *detail_s(char *hinsi)
{
  char *dest;
  if(strcmp("記号",hinsi))
    strcpy(dest,"特1");
  else
    strcpy(dest,"特0");
  return dest;
}

char *detail_h(char *hinsi)
{
  char *dest;
  if(strcmp("名詞接頭辞",hinsi))
    strcpy(dest,"接頭1");
  else
    strcpy(dest,"接頭0");
```

```
  return dest;
}

char *detail_t(char *hinsi)
{
  char *dest;
  if(strcmp("名詞性名詞接尾辞",hinsi))
    strcpy(dest,"接尾 1");
  else
    strcpy(dest,"接尾 0");
  return dest;
}
```

付 録C　頻度データ生成プログラム

```
/* sosei.h */

#ifndef __SOSEI_H__
#define __SOSEI_H__

#include <stdio.h>
#include <string.h>

typedef struct hinshi_hinshi_t {
  char prev_org[64]; /* 前の単語の現形 */
  char prev_type[16]; /* 前の単語の品詞 */
  char curr_org[64]; /* その単語の現形 */
  char curr_type[16]; /* その単語の品詞 */
  char next_org[64]; /* 次の単語の現形 */
  char next_type[16]; /*  次の単語の品詞*/
  char class[4]; /* その単語のクラス */
} hinshi_t;


/* 素性関数のチェック arg1:入力データ arg2:素性データ */
int soseichk(hinshi_t ,hinshi_t );

#endif /* __SOSEI_H__ */

/* ---------------------- sosei.c --------------------------- */
/* 素性関数を読み込みテキストから頻度データを生成するプログラム */
/* ---------------------------------------------------------- */

#include "sosei.h"
#define MAXFUNC 100
```

```
int main(int argc,char *argv[])
{
  FILE *class_file,*sosei_func,*sosei_hind,*sosei_out;
  hinshi_t data, sosei[MAXFUNC];

  char buf[1024],name[100];
  int i=0,sosei_size,class_id=0;
  long num=0,hind=0;

  if(NULL == (class_file = fopen(argv[1],"rt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }

  if(NULL == (sosei_func = fopen(argv[2],"rt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }
  strcpy(name,argv[1]);
  strcat(name,".hnd");
  if(NULL == (sosei_hind = fopen(name,"wt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }

  strcpy(name,argv[2]);
  strcat(name,".out");
  if(NULL == (sosei_out = fopen(name,"wt"))){
    fprintf(stderr,"cannot open file.");
    exit(1);
  }



  printf("reading\n");
  /* 素性関数の読み込み */
  while(NULL != fgets(buf,1024,sosei_func)){
    printf("%s\n",buf);
    sscanf(buf,"%s%s%s%s%s%s%s",sosei[i].prev_org,sosei[i].prev_type,
```

```
            sosei[i].curr_org,sosei[i].curr_type,sosei[i].next_org,
            sosei[i].next_type,sosei[i].class);

    i++;
    printf("読みましたー\n");
  }
  sosei_size=i;
  printf("素性関数の数は%dですー\n",sosei_size);
  while(NULL != fgets(buf,1024,class_file)){
    num++;
    sscanf(buf,"%s%s%s%s%s%s%s",data.prev_org,data.prev_type,
           data.curr_org,data.curr_type,data.next_org,
           data.next_type,data.class);
    for(i=0;i<sosei_size;i++){
      if(soseichk(data,sosei[i]) == 0){
        if(strcmp(data.class,"NS") == 0)class_id=1;
        if(strcmp(data.class,"NM") == 0)class_id=2;
        if(strcmp(data.class,"NE") == 0)class_id=3;
        if(strcmp(data.class,"NI") == 0)class_id=4;
        if(strcmp(data.class,"PS") == 0)class_id=5;
        if(strcmp(data.class,"PM") == 0)class_id=6;
        if(strcmp(data.class,"PE") == 0)class_id=7;
        if(strcmp(data.class,"PI") == 0)class_id=8;

        /* 素性書出し filename:[sosei_func].out */
        /* データ番号,クラス,素性関数番号 */

        fprintf(sosei_out,"%ld %d %d 1\n",num,class_id,i+1);
        hind++;

      }
    }

    /* 素性の頻度情報の書出し データ番号,class,素性関数番号,頻度 */
    /* 素性の頻度情報の書出し filename:argv[2].hnd */
    fprintf(sosei_hind,"%ld %d %d\n",num,class_id,hind);

    hind=0;
  }
```

```
  fclose(class_file);
  fclose(sosei_func);
  fclose(sosei_hind);
  fclose(sosei_out);
  printf("\nEnd.\n");
  return 0;
}

int soseichk(hinshi_t sample,hinshi_t func)
{

  if(strcmp(func.prev_org,"*") != 0)
    if(strcmp(sample.prev_org,func.prev_org) != 0)
      return 1;

  if(strcmp(func.prev_type,"*") != 0)
    if(strcmp(sample.prev_type,func.prev_type) != 0)
      return 2;

  if(strcmp(func.curr_org,"*") != 0)
    if(strcmp(sample.curr_org,func.curr_org) != 0)
      return 3;

  if(strcmp(func.curr_type,"*") != 0)
    if(strcmp(sample.curr_type,func.curr_type) != 0)
      return 4;


  if(strcmp(func.next_org,"*") != 0)
    if(strcmp(sample.next_org,func.next_org) != 0)
      return 5;

  if(strcmp(func.next_type,"*") != 0)
    if(strcmp(sample.next_type,func.next_type) != 0)
      return 6;

  if(strcmp(func.class,"*") != 0)
    if(strcmp(sample.class,func.class) != 0)
      return 7;
```

```
    return 0;
}
```

付 録D　GISによる素性パラメータ推定プログラム

```
/* ---------------------- gis.c --------------------------- */
/* GISアルゴリズムにより素性パラメータを推定するプログラム    */
/* ------------------------------------------------------- */

#include <stdio.h>
#include <ndbm.h>
#include <fcntl.h>
#include <math.h>

#define CLASS         8
#define DATASIZE 178968
#define FTSIZE       17

#define Delta    0.0000055876
#define Com      2.0

DBM *tdata1,*tdata2;

main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  char line[128];
  int  cn,i,loop,r,cl,ft;
  short int CN[DATASIZE+1];
  extern DBM *tdata1,*tdata2;
  datum  key, val;

  double lam[FTSIZE],newlam[FTSIZE],phatI[FTSIZE],poriI[FTSIZE],new;
```

```
  init_lam(lam);

  tdata1 = dbm_open("tdata1",O_RDONLY, 0640);  /* 素性関数 が 1 を返す
データ */
  tdata2 = dbm_open("tdata2",O_RDONLY, 0640);  /* 補完素性 が 1 を返す
データ */

  if((f1 = fopen("cn.out","r")) == NULL) quit("cn.out ファイルが開けな
い");
  cn = 1;
  while(fgets(line,128,f1) != NULL) {
     for(i = 0;line[i] != ' ';i++);
     cl = line[i+1] - '0';
     if ((cl >= 1) && (cl <= 8)){
        CN[cn] = cl;
     } else {
        quit("cn.out がおかしい");
     }
     cn++;
  }
  if((r = fclose(f1)) == -1) quit("ファイル１が閉じれない");
  printf("cn.out file の読み込み終り\n");

  getPhatI(phatI,CN);

  for(ft = 1;ft <= FTSIZE;ft++) {
       if (poriI[ft-1] == 0) {
          new = 1.0;
       } else {
          new = phatI[ft-1]/poriI[ft-1];
       }
       newlam[ft-1] = (1/2.0)*log(new);
  }
  for(ft = 1;ft <= FTSIZE;ft++) {
       lam[ft-1] = exp(newlam[ft-1]);
  }

  loop = 0;
  while(loop < 200) {
```

```
        printf("-- %d -----------------\n",loop);
        print_lam(lam);

        getPoriI(poriI,lam);

        printf("-- poriI -----------------\n");
        print_lam(poriI);

        for(ft = 1;ft <= FTSIZE;ft++) {
            if (poriI[ft-1] == 0) {
               new = 1.0;
            } else {
               new = phatI[ft-1]/poriI[ft-1];
            }
            newlam[ft-1] = log(lam[ft-1]) + (1/2.0)*log(new);
        }
        for(ft = 1;ft <= FTSIZE;ft++) {
            new = exp(newlam[ft-1]);
            lam[ft-1] = exp(newlam[ft-1]);
        }

        loop++;
    }
    printf("-- %d -----------------\n",loop);
    print_lam(lam);
    dbm_close(tdata1);
    dbm_close(tdata2);
}


/*** core ***************/

void getPhatI(double phatI[],short int CN[])
{
    int ft,cl,h,v;
    double mean;

    for(ft = 1;ft <= FTSIZE;ft++) {
        mean = 0.0;
```

```
        for(cl = 1;cl <= CLASS ;cl++) {
            for(h = 1;h <= DATASIZE;h++) {
                if (cl == CN[h]) {
                   if (v = read_tdata(ft,cl,h)) {
                      mean += Delta*((1.0)*v);
                   }
                }
            }
        }
        phatI[ft-1] = mean;
    }
}

void getPoriI(double poriI[],double lam[])
{
  int ft,h,v,cl;
  double mean,pori;
  double getPori(int,int,double []);

  for(ft = 1;ft <= FTSIZE;ft++) {
     mean = 0.0;
     for(cl = 1;cl <= CLASS;cl++) {
        for(h = 1;h <= DATASIZE;h++) {
              if (v = read_tdata(ft,cl,h)) {
                 pori = getPori(h,cl,lam);
                 mean += pori*((1.0)*v);
              }
        }
     }
     poriI[ft-1] = mean;
  }
}


double getPori(int h,int cl,double lam[])
{
  double pconth;
  double aimP(int,int,double []);
```

```
  pconth = aimP(cl,h,lam);
  return Delta*pconth;
}

double aimP(int cl,int h,double lam[])
{
  int cl2,ft,v;
  double z,val,w;

  z = 0.0;
  for(cl2 = 1;cl2 <= CLASS;cl2++) {
    val = 0.0;
    for(ft = 1;ft <= FTSIZE;ft++) {
      if (v = read_tdata(ft,cl2,h)) {
        val += ((1.0)*v)*log(lam[ft-1]);
      }
    }
    if (cl2 == cl) w = exp(val);
    z += exp(val);
  }
  return w/z;
}


int read_tdata(int ft,int cl,int h)
{
  short int a;

  if (ft == 17) {
     a = read_dbm2(ft,cl,h);
     if (a != -1)  {
       return a;
     } else {
       return 2;
     }
  } else {
     return read_dbm1(ft,cl,h);
  }
}
```

```
int read_dbm1(int ft,int cl,int h)
{
  extern DBM *tdata1;

  char num[10],line[20];
  int i,size;
  datum  key, val;

  make_key(line,ft,cl,h);

  key.dptr = line;
  key.dsize = char_size(line);

  val = dbm_fetch(tdata1,key);
  size = val.dsize;
  for(i=0; i < size; i++) num[i] = (val.dptr)[i];
  num[i] = NULL;
  return atoi(num);
}

int read_dbm2(int ft,int cl,int h)
{
  extern DBM *tdata2;

  char num[10],line[20];
  int i,size;
  datum key, val;

  make_key(line,ft,cl,h);

  key.dptr = line;
  key.dsize = char_size(line);

  val = dbm_fetch(tdata2,key);
  size = val.dsize;
  for(i=0; i < size; i++) num[i] = (val.dptr)[i];
  num[i] = NULL;
  return atoi(num);
```

```
}

void make_key(char line[],int ft,int cl,int h)
{
  int pos,div,a,k;
  char tmp[20];

  pos = 0;
  div = h;
  while(div != 0) {
     a = div%10;
     tmp[pos] = a + '0'; pos++;
     div = div/10;
  }
  tmp[pos] = '.';  pos++;
  div = cl;
  while(div != 0) {
     a = div%10;
     tmp[pos] = a + '0'; pos++;
     div = div/10;
  }

  tmp[pos] = '.';  pos++;
  div = ft;
  while(div != 0) {
     a = div%10;
     tmp[pos] = a + '0'; pos++;
     div = div/10;
  }
  line[pos] = NULL;
  for(k = 0;k <= pos - 1;k++) line[k] = tmp[pos - 1 - k];
}

/**** tools ****************/

void init_lam(double lam[])
{
  int i;
```

```
  for(i = 0;i < FTSIZE;i++) {
     lam[i] = 1.0;
  }
}

void quit(char *s)
{
  printf(s); putchar('\n');
  exit(1);
}

int char_size(char *p)
{
  int i;
  for(i=0;p[i] != NULL;i++);
  return i;
}

void print_lam(double lam[])
{
  int i;

  for(i = 0;i < FTSIZE;i++) {
     printf("%d  -->  %f\n",i+1,lam[i]);
  }
}

void chop(char *p)
{
  int i;
  for(i=0;p[i] != NULL;i++);
  if (i > 0) p[i-1] = NULL;
}

/*-------------------------------------------------

construct ndbm
```

GIS のプログラムで利用する素性関数が 1 を返すデータを

ndbm 形式でデータベースにするプログラム

```
-----------------------------------------------------*/

#include <stdio.h>
#include <db1/ndbm.h>
#include <fcntl.h>

main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  int  r,count;
  char line[256],*p;
  DBM *mydb;
  datum  key,content;

  if(argc != 3) quit("引数の数が違う"); /* prog datafile outfile*/
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイル１が開けない");
  mydb = dbm_open(argv[2],(O_RDWR|O_CREAT), 0640);
  count = 1;
  while(fgets(line,256,f1) != NULL) {
     if ((count % 1000) == 0) printf("%d 行終り\n",count);
     key.dptr = strtok(line," ");
     key.dsize = strlen(key.dptr);
     p = strtok(NULL,"\n");
     content.dptr = p;
     content.dsize = strlen(p);
     if (dbm_store(mydb,key,content,DBM_INSERT) < 0) quit("登録できな
い");
     count++;
  }
  if((r = fclose(f1)) == -1) quit("ファイル１が閉じれない");
  dbm_close(mydb);
}

void chop(char *p)
{
  int i;
  i = strlen(p);
```

```
  if (i > 0) p[i-1] = NULL;
}

void quit(char *s)
{
  printf(s); putchar('\n');
  exit(1);
}
```

付 録E　確率付与プログラム

```
/* pth.h */

#ifndef __PTH_H__
#define __PTH_H__

#include <stdio.h>
#include <string.h>
#include <math.h>

#define SOSEI_SIZE 17
#define CLASS_SIZE 8

typedef struct hinshi_hinshi_t {
  char prev_org[64]; /* 前の単語の現形 */
  char prev_type[16]; /* 前の単語の品詞 */
  char curr_org[64]; /* その単語の現形 */
  char curr_type[16]; /* その単語の品詞 */
  char next_org[64]; /* 次の単語の現形 */
  char next_type[16]; /*  次の単語の品詞*/
  char class[4];
} hinshi_t;

typedef struct sosei_sosei_t {
   char prev_org[8]; /* 前の単語の現形 */
   char prev_type[8]; /* 前の単語の品詞 */
   char curr_org[8]; /* その単語の現形 */
   char curr_type[8]; /* その単語の品詞 */
   char next_org[8]; /* 次の単語の現形 */
   char next_type[8]; /*  次の単語の品詞*/
   char class[4];
} sosei_t;
```

```
/* 素性関数のチェック arg1:入力データ arg2:素性関数 */
int soseichk(hinshi_t ,sosei_t);
char *set_class(int);

/* データより確率P(t^h)を出す */
float pth(hinshi_t);
#endif /* __PTH_H__ */

/* ---------------------- pth.c --------------------------- */
/* 単語列からME法により各クラスに属する確率を与えるプログラム */
/* ------------------------------------------------------- */

#include "pth.h"

int main(int argc,char *argv[])
{
  /* データファイル:kekka.class 補完素性ファイル:kekka.hokan */
  /* 頻度ファイル:kekka.?.hnd */
  /* 補完素性は存在しない番号は頻度は2 */

  FILE *class_file,*juman_file;
  int i,j,maxsosei,sosei_result=0,class_num;
  char tango[32];
  hinshi_t in_data;

  if(NULL == (class_file = fopen(argv[1],"rt"))){
    printf("error.cannot open classed-file\n.");
    exit(1);
  }

  if(NULL == (juman_file = fopen(argv[2],"rt"))){
    printf("error.cannot open juman-file\n.");
    exit(1);
  }


  while(EOF!= fscanf(juman_file,"%s %*s %*s %*s %*s %*s %*s %*s %*s %*s
                     %*s",tango)){
```

```
    if(strncmp("EOS",tango,3) == 0){
      printf("%s\n",tango);
      continue;
    }
    fscanf(class_file,"%s%s%s%s%s%s",in_data.prev_org,in_data.prev_type,
    in_data.curr_org,in_data.curr_type,in_data.next_org,in_data.next_type);
    printf("%s",in_data.curr_org);
    pth(in_data);
    printf("\n");
  }
  return 0;

}

float pth(hinshi_t data)
{
  int i,j,k,class_id;
  float hokan=0,func=0,tmp=0,hind=0;
  float alpha[SOSEI_SIZE],ta=1,zta=0,tfact=1;

  sosei_t sosei[SOSEI_SIZE];

  float f[SOSEI_SIZE];



  strcpy(sosei[0].prev_org,"*");
  strcpy(sosei[0].prev_type,"*");
  strcpy(sosei[0].curr_org,"*");
  strcpy(sosei[0].curr_type,"特1");
  strcpy(sosei[0].next_org,"*");
  strcpy(sosei[0].next_type,"*");
  strcpy(sosei[0].class,"NE");

  strcpy(sosei[1].prev_org,"*");
  strcpy(sosei[1].prev_type,"*");
  strcpy(sosei[1].curr_org,"*");
  strcpy(sosei[1].curr_type,"*");
  strcpy(sosei[1].next_org,"*");
```

```
strcpy(sosei[1].next_type,"名2");
strcpy(sosei[1].class,"NE");

strcpy(sosei[2].prev_org,"*");
strcpy(sosei[2].prev_type,"名2");
strcpy(sosei[2].curr_org,"*");
strcpy(sosei[2].curr_type,"*");
strcpy(sosei[2].next_org,"*");
strcpy(sosei[2].next_type,"*");
strcpy(sosei[2].class,"NI");

strcpy(sosei[3].prev_org,"*");
strcpy(sosei[3].prev_type,"*");
strcpy(sosei[3].curr_org,"*");
strcpy(sosei[3].curr_type,"*");
strcpy(sosei[3].next_org,"*");
strcpy(sosei[3].next_type,"特1");
strcpy(sosei[3].class,"NI");

strcpy(sosei[4].prev_org,"*");
strcpy(sosei[4].prev_type,"*");
strcpy(sosei[4].curr_org,"*");
strcpy(sosei[4].curr_type,"*");
strcpy(sosei[4].next_org,"*");
strcpy(sosei[4].next_type,"名1");
strcpy(sosei[4].class,"NM");

strcpy(sosei[5].prev_org,"*");
strcpy(sosei[5].prev_type,"*");
strcpy(sosei[5].curr_org,"*");
strcpy(sosei[5].curr_type,"名1");
strcpy(sosei[5].next_org,"*");
strcpy(sosei[5].next_type,"*");
strcpy(sosei[5].class,"NM");

strcpy(sosei[6].prev_org,"*");
strcpy(sosei[6].prev_type,"*");
strcpy(sosei[6].curr_org,"*");
strcpy(sosei[6].curr_type,"*");
```

```
strcpy(sosei[6].next_org,"*");
strcpy(sosei[6].next_type,"名1");
strcpy(sosei[6].class,"NS");

strcpy(sosei[7].prev_org,"*");
strcpy(sosei[7].prev_type,"名2");
strcpy(sosei[7].curr_org,"*");
strcpy(sosei[7].curr_type,"*");
strcpy(sosei[7].next_org,"*");
strcpy(sosei[7].next_type,"*");
strcpy(sosei[7].class,"NS");

strcpy(sosei[8].prev_org,"*");
strcpy(sosei[8].prev_type,"名1");
strcpy(sosei[8].curr_org,"*");
strcpy(sosei[8].curr_type,"*");
strcpy(sosei[8].next_org,"*");
strcpy(sosei[8].next_type,"*");
strcpy(sosei[8].class,"PE");

strcpy(sosei[9].prev_org,"*");
strcpy(sosei[9].prev_type,"*");
strcpy(sosei[9].curr_org,"*");
strcpy(sosei[9].curr_type,"*");
strcpy(sosei[9].next_org,"*");
strcpy(sosei[9].next_type,"特1");
strcpy(sosei[9].class,"PE");

strcpy(sosei[10].prev_org,"*");
strcpy(sosei[10].prev_type,"特1");
strcpy(sosei[10].curr_org,"*");
strcpy(sosei[10].curr_type,"*");
strcpy(sosei[10].next_org,"*");
strcpy(sosei[10].next_type,"*");
strcpy(sosei[10].class,"PI");

strcpy(sosei[11].prev_org,"*");
strcpy(sosei[11].prev_type,"*");
strcpy(sosei[11].curr_org,"*");
```

```
strcpy(sosei[11].curr_type,"名2");
strcpy(sosei[11].next_org,"*");
strcpy(sosei[11].next_type,"*");
strcpy(sosei[11].class,"PI");

strcpy(sosei[12].prev_org,"*");
strcpy(sosei[12].prev_type,"*");
strcpy(sosei[12].curr_org,"*");
strcpy(sosei[12].curr_type,"名1");
strcpy(sosei[12].next_org,"*");
strcpy(sosei[12].next_type,"*");
strcpy(sosei[12].class,"PM");

strcpy(sosei[13].prev_org,"*");
strcpy(sosei[13].prev_type,"*");
strcpy(sosei[13].curr_org,"*");
strcpy(sosei[13].curr_type,"名1");
strcpy(sosei[13].next_org,"*");
strcpy(sosei[13].next_type,"*");
strcpy(sosei[13].class,"PM");

strcpy(sosei[14].prev_org,"*");
strcpy(sosei[14].prev_type,"特1");
strcpy(sosei[14].curr_org,"*");
strcpy(sosei[14].curr_type,"*");
strcpy(sosei[14].next_org,"*");
strcpy(sosei[14].next_type,"*");
strcpy(sosei[14].class,"PS");

strcpy(sosei[15].prev_org,"*");
strcpy(sosei[15].prev_type,"*");
strcpy(sosei[15].curr_org,"*");
strcpy(sosei[15].curr_type,"名2");
strcpy(sosei[15].next_org,"*");
strcpy(sosei[15].next_type,"*");
strcpy(sosei[15].class,"PS");

  alpha[0] =   7.79;
  alpha[1] =   7.23;
```

```
  alpha[2] =   0.53;
  alpha[3] =   1;
  alpha[4] =   385.70;
  alpha[5] =   338.62;
  alpha[6] =   5.29;
  alpha[7] =   5.26;
  alpha[8] =   3.28;
  alpha[9] =    2.84;
  alpha[10] =   4.86;
  alpha[11] =   3.02;
  alpha[12] =   1.63;
  alpha[13] =   1.52;
  alpha[14] =   3.92;
  alpha[15] =   3.53;

for(i=0;i<CLASS_SIZE;i++){
  tfact = 1;hind = 0;
  strcpy(data.class,set_class(i+1));
  /*    printf("クラス:%s\n",data.class);*/
  for(j=0;j<SOSEI_SIZE-1;j++){
    /*      printf("素性%d:クラス:%s",j+1,sosei[j].class);*/
    func = soseichk(data,sosei[j]);
    /*      printf("func:%f ",func);*/
    hind += func;
    tfact *= pow(alpha[j],func);
    /*      printf("fact:%f\n",tfact);*/
  }
  /*    printf("頻度は%f",hind);*/
  hokan=pow(alpha[SOSEI_SIZE-1],2-hind);
  /*    printf("補完は%f\n",hokan);*/
  /*    tfact *= hokan;*/
  zta += tfact;
  /*    printf("tfact=%f:zta=%f\n",tfact,zta);*/
}

for(i=0;i<CLASS_SIZE;i++){
  tfact = 1;hind = 0;
  strcpy(data.class,set_class(i+1));
  for(j=0;j<SOSEI_SIZE-1;j++){
```

```
        func = soseichk(data,sosei[j]);
        hind += func;
        tfact = tfact*pow(alpha[j],func);
    }
    hokan=pow(alpha[SOSEI_SIZE-1],2-hind);
    /*    tfact *= hokan;*/

    printf(" %f",tfact/zta);

    /*    printf("%d-ta=%f zta=%f : %f\n",i+1,tfact,zta,tfact/zta); */
  }
  return 0.1;
}

int soseichk(hinshi_t sample,sosei_t func)
{

  if(strcmp(func.prev_org,"*") != 0)
    if(strcmp(sample.prev_org,func.prev_org) != 0)
      return 0;

  if(strcmp(func.prev_type,"*") != 0)
    if(strcmp(sample.prev_type,func.prev_type) != 0)
      return 0;

  if(strcmp(func.curr_org,"*") != 0)
    if(strcmp(sample.curr_org,func.curr_org) != 0)
      return 0;

  if(strcmp(func.curr_type,"*") != 0)
    if(strcmp(sample.curr_type,func.curr_type) != 0)
      return 0;

  if(strcmp(func.next_org,"*") != 0)
    if(strcmp(sample.next_org,func.next_org) != 0)
      return 0;

  if(strcmp(func.next_type,"*") != 0)
    if(strcmp(sample.next_type,func.next_type) != 0)
```

```
        return 0;

    /*  if(strcmp(func.class,"*") != 0) */
      if(strcmp(sample.class,func.class) != 0)
        return 0;

    return 1;
}

char *set_class(int type)
{
    if(type == 1)
      return "NS";
    else if(type == 2)
      return "NM";

    else if(type == 3)
      return "NE";

    else if(type == 4)
      return "NI";

    else if(type == 5)
      return "PS";

    else if(type == 6)
      return "PM";

    else if(type == 7)
      return "PE";

    else if(type == 8)
      return "PI";

    return NULL;
}
```