第２部　データベースと計算機処理

印欧語根を用いたハイパー英単語辞書

池辺 八洲彦　蔡 東生　大出 真

1. はじめに

自然言語を学習するものにとって、辞書が必要不可欠なツールであることは言うまでもない。これまで辞書は紙という記録媒体に依存していたため、その内容や情報の構造に幾つかの問題や制限があったが、近年の計算関連分野の急速な発展により、CDROM などに代表される大容量の記録媒体と、WWW やリレーショナルデータベースなどの新たな技術を利用することで、紙媒体にとらわれない新たな辞書を開発することが可能になっている。

われわれの研究室では「辞書の改革なくしては語学教育の改革は有り得ない」との立場に立ち、特に英単語学習者にとって役に立つ辞書の形を模索してきており、それらの研究では一貫して、英単語学習を手助けする情報として語源情報である印欧語根に着目している。

本稿では、印欧語根について簡単に説明し、具体的な英単語学習例と印欧語根の判明率から印欧語根情報が英単語学習のために有用であることを示すとともに、インターネット上で既存の Web ブラウザを用いて検索利用できるように試作したハイパー英単語辞書について説明する。

2. 欧祖語と印欧語根

本研究では印欧語根情報を用いる英単語学習法を採用している。この「印欧語根」とは、聞き慣れない言葉であると思う。そこでまず、印欧語根 (Indo-European Roots) と、それに関連した印欧祖語 (Proto-Indo European) について簡単に説明する。

印欧祖語

印欧祖語は、それ自体が現実の言語として歴史上存在した自然言語ではなく、19 世紀からヨーロッパにおいて急激に発展した比較言語学によって、印欧諸語

(Indo-Europeans)に見られる規則的な音韻変化をたどるなどして、理論的な再構成によって得られた“仮想”の言語である。現代英語も印欧諸語の一つであり、印欧祖語から派生したものであるといえる（図１参照）。印欧祖語が“仮想”であると書いた理由は、印欧祖語は少なくとも紀元前数千年ごろの言語とされており、文証されない言語であるからである。

印欧語根

印欧語根は、印欧祖語の個々の語における語幹をいくらかのパターンにまとめたものである。印欧祖語自体が文証されない言語であるので、それを構成していたとされる語根も推定の形であるが、文献［１］において言語学者 Calvert Watkins は印欧語根を 596 種に分類し、その意味論的説明、その印欧語根より由来するとされる現代英語、およびその発展過程の概略を示している。

しかしながら、実際の言語間の派生関係は図１のように単純なものではなく、地理的・経済的な理由などで別の言語文化を持った地域から導入された借入語や、外来語と元あった語との複合語などが存在し、錯綜とした関係が形作られている。そのために、全ての現代英語においてその基となった印欧語根が明らかにされているわけではなく、印欧語根不明の場合がある。

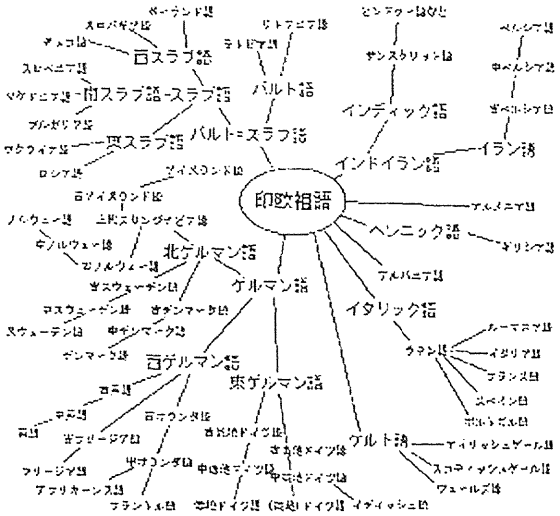

図 1　印欧諸語から派生したとされる諸言語

2.1.　印欧語根に注目する理由

前節で述べたように、言語学者 Calvert Watkins は文献［1］において印欧語根を 596 種に分類している。同辞典の第 1 版から比較すれば、分類の併合・削除によってかなり簡潔化したことになる。もし、英語を外国語とする者が、学習すべき膨大な数の英単語がわずか数百種に分類されるとすれば、英単語学習の面から見てかなりのメリットであろう。また、印欧祖語は数千年も昔の仮想言語であるにもかかわらず、それを構成していた印欧語根の意味・形態が現在の英単語においても生き続けている場合がよく見られ、印欧語根の観点から英単語についての理解を深めることもできる。

したがって、印欧語根を用いる学習の大きなメリットは、

- 同一の語根に由来する英単語（同根語）をまとめて学習できる
- 類義語の意味上の違いや反義語との対照を、印欧語根の観点から考察できる

という従来の学習用辞書や単語集を中心とした学習では難しい学習方法が、比較的容易にできる点にあると我々は考えている。

2.2.　印欧語根を用いた単語学習実例

以下は前節であげた学習の典型例である。

1.　　同根語をまとめて学ぶ

印欧語根[**teua-**]は「ふくれる」という概念を表わす。同根語としては，butter（バター），thousand(千)，thumb(親指)，tomb（墓），tumor（腫瘍）などがある。thousand は hundred がふくらんだものと考えられる。他の各単語も「ふくれる」といった概念を語義に含んでいることが認められる。このように同一の語根に由来する単語をまとめて学ぶことにより，単語間の関係やその基本的な語義概念を容易にイメージすることができ，個別に単語を学ぶより，広く深い理解が得られると考えている。

2．類義語の差異を学ぶ

suspect と doubt は大意において「疑う」という意味をもつ類義語であるが、微妙に使われ方が違う。この差異を印欧語根から推察してみることができる。suspect は印欧語根[**upo**]+ [**spek-**]から由来している。[**upo**]は「～を超えて」といった概念を表し、一方[**spek-**]は「見ること、観察すること」といった概念を表している。ここから「疑いを抱かせるような点があるために気づく」といったニュアンスを持つことがなんとなく推察できる。一方 doubt は印欧語根[**dwo-**]に由来している。印欧語根[**dwo-**]は「2（two)」という概念を表す。ここから「二つの選択肢で迷う」といった doubt のニュアンスを推察できる。これら印欧語根の情報を例文や語義の説明に付け加えることで語彙力をより増やすことに役立つと考えている。

この他にも、歴史的観点から英単語を学ぶことで英語自体に対する興味をより深めるという副次的な効果も期待できると考えている。

3.　単語の印欧語根調査

前節では印欧語根を用いる英単語学習法を紹介したが、もととなる印欧語根が不明である単語も存在するので、どの英単語にもこの学習法が適用できるわけではない。そこで、実際に英単語を集めた場合にどの程度の割合で印欧語根が判明するか（印欧語根の判明率）調査を行い、この学習法の有効性を検証した。またどのような同根語のグループが得られるかについても調査した。

3.1.　印欧語根判明調査方法

英単語の印欧語根調査については、アメリカで最も広く使われているという英語辞典である文献［1］、またその日本における CD-ROM メディア出版物である文献［2］の 596 種の印欧語根分類を利用した。ここで、英単語の印欧語根が判明するとはどういうことか、具体的に説明する。

In tend (¹n-tµnd") *v.* **in tend ed, in tend ing, in tends.** *–tr.* **1.** To have in mind; plan: *We intend to go. They intend going. You intended that she go.* **2.a.** To design for a specific purpose. **b.** To have in mind for a particular use. **3.** To signify or mean. *--intr.* To have a design or purpose in mind. [Middle English *entenden*, from Old French *entendre*, from Latin *intendere* : *in-*, toward; see IN-² + *tendere*, to stretch; see **ten-** below.]　（文献［1］より引用）

図 2　英単語の記述例

図 2 は英単語 intend を文献［1］で引いたときの記述である。‘［…］’で囲まれた部分は語源情報を示している。この場合、英単語 intend は中英語の *entenden*、古フランス語の *entendre*、ラテン語の *intendere* などから派生したことが書かれているが、最後に**[ten-]**という太字の文字列がある。これが intend の由来となる印欧語根である。印欧語根についての記述を調べると、図 3 のように書かれている。**[ten-]**の説明を見ると、この印欧語根と intend の関係について述べられている。

ところで、英単語 intend にはまだ印欧語根が含まれている。それは、intend の語根情報の中で、“see（～を見よ）”により示されている**[IN-²]**の記述から導かれる印欧語根**[en]**である。

このように、一つの英単語が複合語として印欧語根を複数個持っている場合、その複合語の要素である接頭辞・語幹・接尾辞の記述を調べて、由来する全ての印欧語根を明らかにする必要がある。

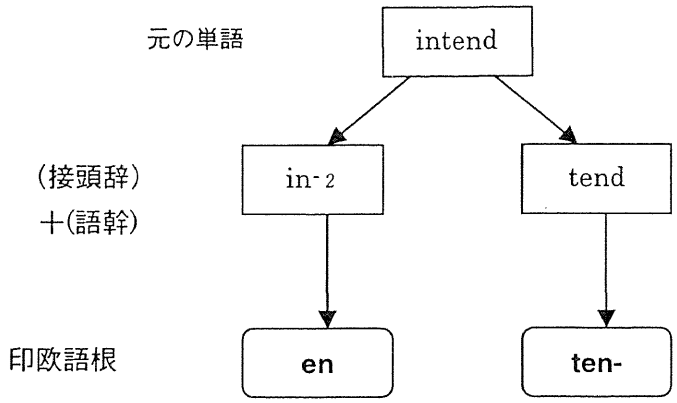

図 3　intend に由来する印欧語根

3.2.　調査結果

まずは、辞書のデータを収集する過程で用いた各単語集や学習参考書における由来する印欧語根の判明率の調査結果を表 1 にまとめた。

ここでいう由来する印欧語根の判明は、少なくとも一つの印欧語根に由来することが判明した英単語の場合を指す。

対象によって多少ばらつきがあるが、全体においては約 80％弱の割合で判明しており、この結果は印欧語根による分類に則った学習法が、十分実用性のあるものであるといえる。

次に、全体での印欧語根出現度数について調べた調査結果を表 2 に示す。

ちなみに，出現回数の一番多い[per1]は非常に広い意味を持つ印欧語根で，基本的には「前に」「…を経て」を表す前置詞の意味を持つ。その他に「in front of, before, early, first, chief, toward, against, near, at, around」のような広い意味を表す。

出現頻度上位 10 位までの印欧語根が，出現総数の約 4 分の 1 以上をしめ，20 位までで全体の 3 分の 1 以上，55 位までで全体の半分を占める事が判明した。印欧語根は 596 種に分けられているが，その数が学習者の負担になるようであれば上位に来るような印欧語根から優先的に学ぶようにすることで学習効率が上昇することを期待できるであろう。

単語集	単語数	判明率
資料[1]の印欧語根解説より important Derivatives	3959	(100.0)
VIS-ED NO.1 (Visual Education assn. 刊)	1000	74.0
VIS-ED NO.2 (Visual Education assn. 刊)	998	72.0
How to Prepare for the GRE (Barron's Educational Series, Inc 刊)	3737	73.1
How to Prepare for TOEFL (Barron's Educational Series, Inc 刊)	634	69.1
英単語頻出案内 (桐原書店 刊)	1526	85.1
英単語連想記憶術 (青春出版社 刊)	1706	77.4
試験に出る英単語 (青春出版社 刊)	1628	80.8
英単語ターゲット 1900 (旺文社 刊)	2143	78.6
速読英単語①必修編 (増進会出版社 刊)	1877	81.9
英検準一級参考書 (旺文社 刊)	557	76.3
全体(重複を除く)	9777	76.5

表 1 単語集別印欧語根判明率

順位	表記	累積出現割合(%)
1	[per1]	3.79
2	[kom]	7.48
3	[en]	10.89
4	[ne]	13.86
5	[re-]	16.70
10	[wer-2]	27.75
55	[al-1]	35.75
164	[wen-1],etc.	50.30

表 2 印欧語根の出現頻度

4.　辞書をデータベース化する意義

従来の学習用辞書の欠点を分析すると、内容的欠点(扱っている情報の質と量)と構造的欠点(辞書が紙を媒体として構築されている)の2種類に大別できる。辞書をデータベース化することにより、

- 多量、多種類の情報を扱える
- 複雑な検索要求に応えられる

という利点が生じ、またハイパーテキストの特徴であるリンクを利用することで、紙媒体の辞書に較べ格段に便利になる。

今回は、前節で述べた学習法による英単語の語義の理解に重点を置きデータベース化した。対象とする辞書要素は、見出し語の由来する印欧語根、語義の簡単な説明、類義語比較、接頭辞や語幹などの情報である。

5. ハイパー辞書の製作と公開

当研究室での英単語学習ハイパー辞書の研究は、実際に英単語学習において利用されることを最終目的としており、“できる限り幅広い人々に利用される媒体”として、WWW上で公開する方法を用いている。

5.1.　インターフェース

自然言語学習用辞書であることから、当然コンピューターにあまり馴染みのない人々から利用される場合も考えなければならない。そのため、辞書におけるインターフェースは非常に重要であり、コンピューターに慣れている人はもちろん、そうでない人にも分かりやすく操作しやすいインターフェースについて考える必要がある。

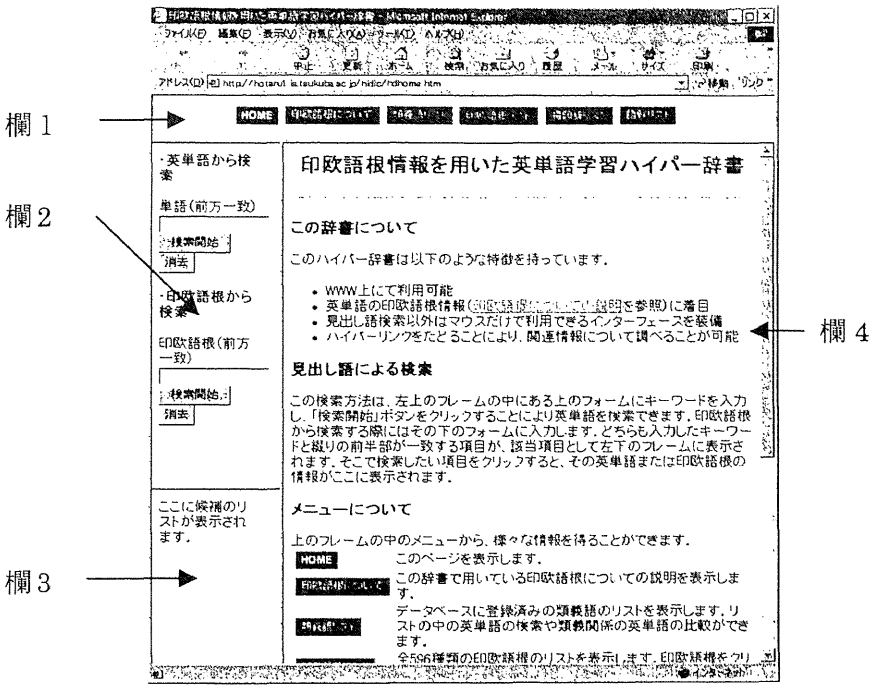

図 4　辞書のスタート画面

欄 1 は、様々な情報を欄 4 に表示させるリンクを持つメニューのためのフレームである。

欄 2 は、英単語の綴りか印欧語根の表記による見出し語検索の際にキーワードを入力するためのフレームである。

欄 3 は、欄 1 で入力したキーワードから始まる英単語または印欧語根の候補が表示されるフレームであり、その綴りをクリックすると欄 4 にその英単語についての情報が表示される。

欄 4 は、この英単語学習ハイパー辞書の主要な情報を表示するフレームである。

5.2.　仕組み

今回製作したハイパー英単語辞書は、WWW サーバとリレーショナルデータベースシステムを仲立ちする CGI プログラムを作成することによって、インターネット上で既存の Web ブラウザを用いて検索できるようになっている。また

これにより、ハイパーテキストの特性を生かした辞書の機能が実現されている。

5.3.　機能

英単語ハイパー辞書では以下の機能を備えている。

1．　見出し語検索

英単語の綴り、もしくは印欧語根の表記から情報を検索する。ユーザが入力したキーワードと綴りが前方で一致する項目を該当項目として表示し、その該当項目をクリックするとその情報を表示する。

2．　メニューによる検索

欄1のメニューには6つのリンクがあり、左から、ハイパー辞書の使い方などを表示するリンク（「HOME」と書かれたボタン）、印欧語根についての説明を表示するリンク（「印欧語根について」）、データベースに登録されている類義語のリストを表示するリンク（「類義語リスト」）、596種の印欧語根のリストを表示するリンク（「印欧語根リスト」）、接頭辞のリストを表示するリンク（「接頭辞リスト」）、語幹のリストを表示するリンク（「語幹リスト」）である。図4の欄4に表示されている情報は、この欄1の一番左のリンクである「HOME」をクリックすることで、いつでも参照することができる。

3．　見出し語の情報の一部をクリックすることにより、その語句に対して検索を行う。

英単語の情報には、その単語の接頭辞、語幹および由来する印欧語根の説明があり、その中の数多くのリンク部分をクリックすることで、関連情報を表示することができる。

開発環境は表3の通りである。Access と Visual Basic は親和性が高くこの程度の規模のデータベースシステムならば比較的容易に構築することができる。ただ将来大規模に公開する場合は、より性能が良く、信頼度の高い環境に移行する必要がある。

OS	Windows NT Workstation 4.0
RDBMS	Access97
CGI プログラミング	Visual Basic 5.0
Web サーバ	Peer Web Services 2.0

表 3　開発環境

なお、今回作成した英単語学習ハイパー辞書は

http://hotaru1.is.tsukuba.ac.jp/hidic/hdhome.htm

において公開している。

6. まとめと今後の展開

本稿では、英単語学習法の一つとして、印欧語根を用いることを提案し、その有効性を検証調査した。また将来の本格的なハイパー辞書データベースの公開に向けて、現在まで収集したデータをWWW上で検索できるシステムの構築を行った。

今後の展開としては、学習用辞書としての実用性をより高めるために、フリーなデータソースを利用した語彙数の拡大、印欧語根以外の語源情報や例文などデータベース自体の拡張、そして本格的な公開に備えた計算機環境の整備が挙げられる。

参考文献

[1] The American Heritage Dictionary of The English Language, Third Edition : Houghton Mifflin Company, 1992.

[2] アメリカン・ヘリテージ トーキング英英辞典 第三版: 住友金属工業株式会社 オープンシステム事業室, 1994.

[3]「印欧語根情報を用いた英単語ハイパー辞書に関する研究」: 吉村 信吾 著, 平成8年度 筑波大学大学院理工学研究科修士論文, 1997.

[4]「英単語学習参考書を対象としたハイパー辞書の研究」: 飯田 浩隆 著, 平成10年度 筑波大学第三学群情報学類学士論文, 1999.

[5] 英語の辞書と語源 : 今里 智晃, 土家 典生 著, 大修館書店, 1984.

Composing vectorial representation for noun phrase using neural networks

TAKAHASHI Naoto
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba
305-8568 JAPAN
ntakahas@etl.go.jp

Abstract

Miikkulainen extended backpropagation to modify not only link weights but also input patterns given to neural networks. He trained his neural network with this extended backpropagation algorithm, called FGREP, to learn case-role assignment. The simulation result showed that FGREP generated vectorial word representations that reflect the word usage in the training corpus.

Takahashi and Motoki modified the output format of FGREP to accelerate and stabilise the learning of the neural network. They used the same training corpus as in Miikkulainen's work and showed their superiority over the original FGREP.

In this paper, we add two neural networks to theirs so that noun phrases of the form 'adjective + noun', in addition to simple nouns, are accepted as syntactic constituents in input sentences. One of the added neural network is *the composer network*; it composes the vectorial representation of a noun phrase from the vectorial representations of the constituents of that noun phrase. The other is *the decomposer network*; it assures that all constituents' informations are reflected in the composed noun phrase representation.

We also describe two prominent results of our experiment with these three neural networks: 1) Some regularities were observed between the representation of a noun and the representation of a noun phrase containing that noun. The relationship between the two representations implied that adjectives work as mapping functions from noun representations to noun phrase representations; 2) The result of principal component analysis showed that the information of the adjective in a noun phrase was encoded as a principal component. In other words, it is possible to know which adjective is used in a noun phrase by checking its vectorial representation.

1 Introduction

For natural language processing with computers, it is necessary to represent words in a form that reflects their meaning. Widely used representations include semantic

features and classification IDs defined in a thesaurus. These informations are given from human to system as *a priori* knowledge.

There are, however, another approach to define word representations: processing sample sentences statistically to generate data-oriented word representations. Hindle's work [1] is a famous example of relatively traditional statistical approaches.

One way to process data statistically is to use neural networks. For example, Miikkulainen and Dyer succeeded in generating vectorial representations for words based on an artificial corpus [2, 3]. They used a neural network with extended backpropagation, called FGREP (Forming Global Representations with Extended backPropagation), to modify input patterns in addition to link weights in the neural network. The principal of FGREP is as follows.

In a layered neural network, the error signal for a non-output unit is given by the following equation [4]:

$$\delta_i = f_i'(net_i) \sum_j \delta_j w_{ij}, \tag{1}$$

where net_i, f_i' and δ_i are the total input, the derivative of the activation function and the error signal of unit i; also w_{ij} represents the weight of the link connecting unit i and unit j. Here, unit i and unit j belong to different layers and unit i's layer is closer to the input layer than unit j's is.

Input units pass the received signal to the next layer without modifying it. In other words, input units can be regarded as units having the identity function $f(x) = x$ as the activation function.

Since the derivative of the identity function is one, the error signal equation for input units becomes:

$$\delta_i = \sum_j \delta_j w_{ij}. \tag{2}$$

Thus, input patterns shall be updated according to the next equation:

$$\Delta r_{ci} = \eta \delta_i, \tag{3}$$

where η is the learning rate and r_{ci} is the i-th element of the input vector c.

Figure 1 is the diagram of the feed-forward version of FGREP network. The network is trained to fill each case-role assembly at the output layer with the appropriate word representation, which is given to the input layer. Then the error signal at the output layer is propagated backwardly through the network to update link weights and input word representations. Each word representation has a random vector as its initial value, but the representations of similarly used words gradually get closer as the training proceeds. As a result, word representations form self-organised clusters.

Takahashi and Motoki changed the target pattern of the FGREP network from sequence of word representations to sequence of case-roles [5]. This modification accelerated learning speed and resolved FGREP's potential problems including:

- all word representations may converge at the same vector, and
- learning can be unstable because it modifies target patterns.

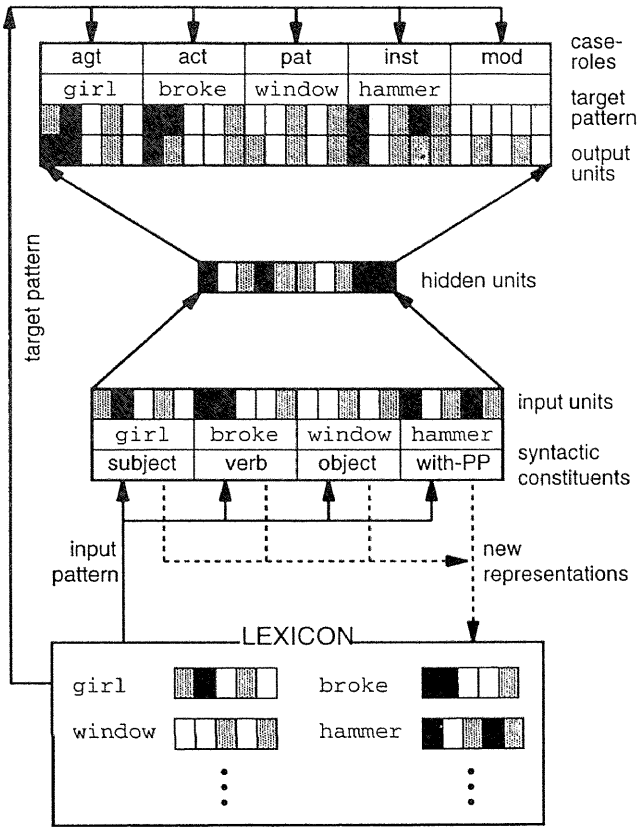

Figure 1: Miikkulainen's neural network using FGREP. The network is trained to reproduce input word representations at appropriate case-role positions.

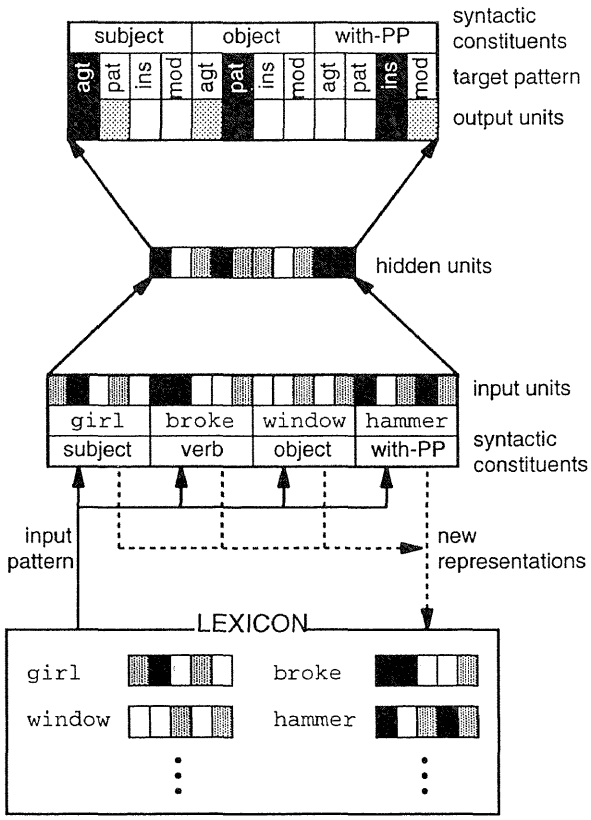

Figure 2: Structure of CAN. The network is trained to output case-roles at appropriate positions. The case-role of the verb is omitted because it is always 'act'.

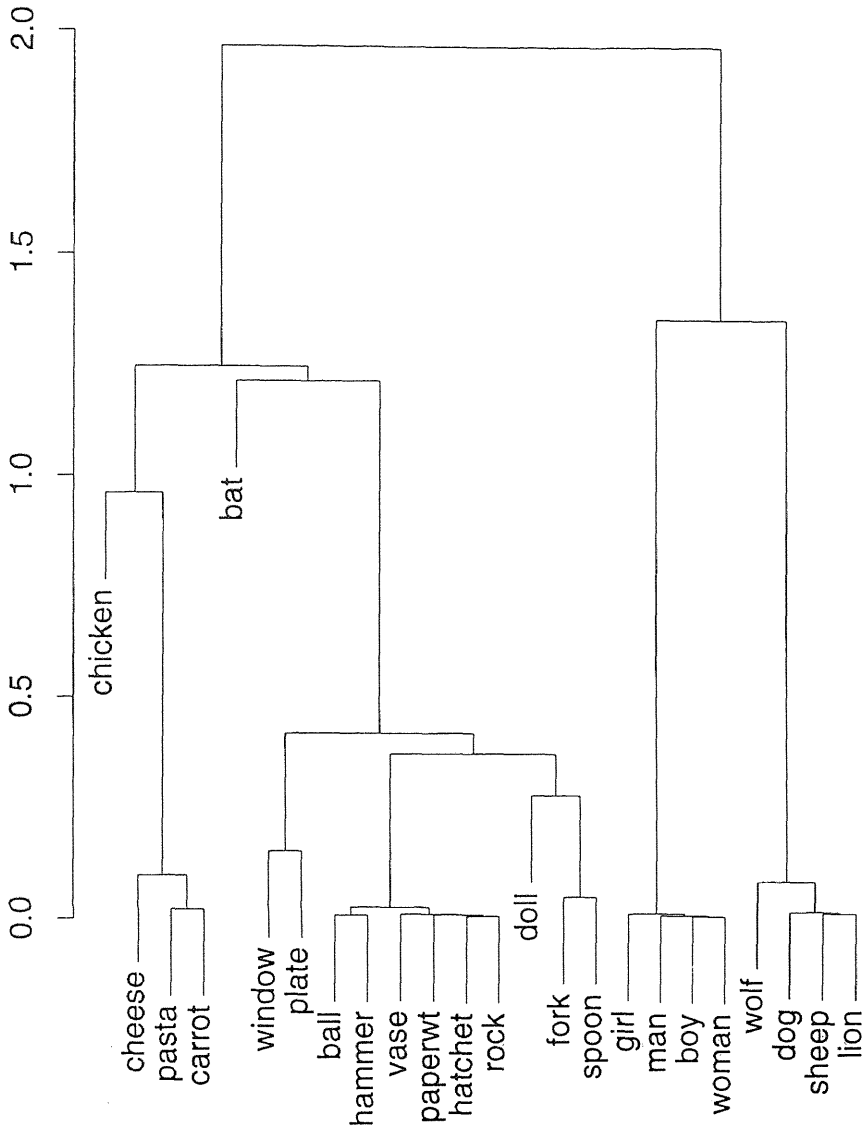

Figure 3: The clustering result of the word representations generated by CAN. The clustering is based on Euclidean distance. Similar representations are assigned to similarly used words. The nouns 'chicken' and 'bat' have peculiar representations because they are used as polysemous words in the training corpus. (Food and bird for 'chicken' and stick and animal for 'bat'.)

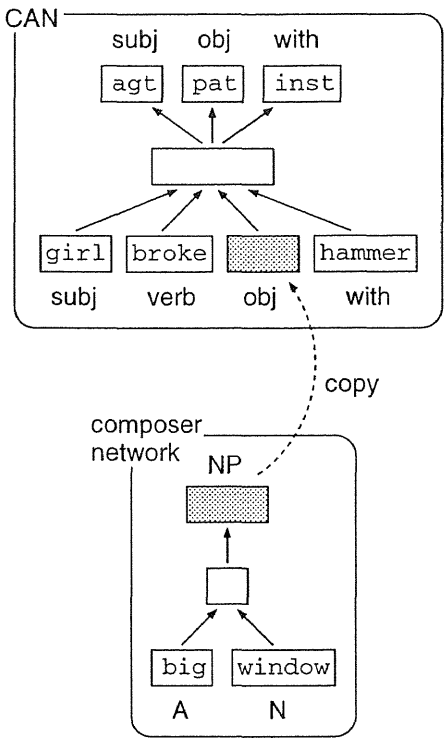

Figure 4: The composer network generating the vectorial representation of the noun phrase 'big window' from 'big' and 'window'. Its output is copied to an input assembly of CAN.

Figure 2 and Figure 3 show their neural network, called CAN (Case-role Assignment Network), and the clustering result of the generated word representations.

2 Composing noun phrase representation

CAN, as well as the FGREP network, accepts only those sentences whose syntactic constituents are simple nouns; it cannot accept sentences that contain noun phrases consisting of two or more words.

Takahashi and Motoki proposed a mechanism to make CAN accept general noun phrases [5], but it was merely an idea without experiment. In this section, we first point out that their proposal is not sufficient to accept noun phrases, then we give a supplemental mechanism to make it complete. We use noun phrases of the form 'adjective + noun' as an example, but the technique described here can be applied to any other forms of noun phrase.

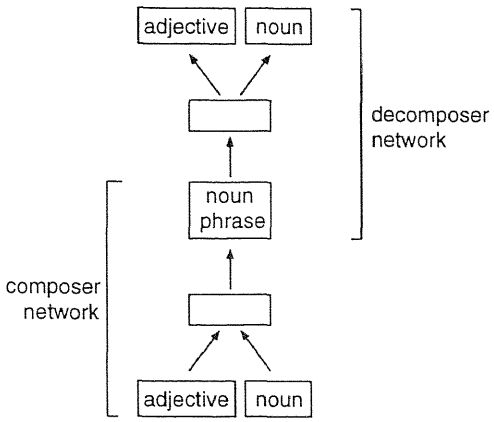

Figure 5: The composer and the decomposer networks. They are connected tandem to make a five layer network for the auto-association task.

2.1 Composer network

Takahashi and Motoki assumed Fregean principle and proposed *the composer network* that composes the vectorial representation of a phrase from the vectorial representations of its components.

The composer network is to be trained in conjunction with CAN. Figure 4 shows how the two neural networks are to be trained with the sentence "The girl broke the big window with a hammer".

First, the composer network composes the vectorial representation of the noun phrase 'big window' from the representations of 'big' and 'window'. This noun phrase representation is passed to CAN as if it were a single word. CAN processes all input representations and calculates the output. The output is compared with the target pattern and the error signal is propagated backwardly from CAN to the composer network. Finally, the word representations given to the composer network, as well as the word representations directly given to CAN, are updated based on the extended backpropagation as defined by Equation (3).

We adopt their composer network for the same purpose. We, however, use yet another network as described below.

2.2 Decomposer network

Adding the composer network to CAN makes it possible to process noun phrases of the form 'adjective + noun', but it is not sufficient because there is a potential problem as described below.

We assumed that the existence of an adjective does not affect the target case-role of the head noun in a noun phrase. For example, the noun 'window' in "The girl broke the *big window* with a hammer" has the case-role 'patient' as well as the 'window' in "The girl broke the *window* with a hammer" has.

Since the representation of 'big window' is given to CAN as if it were a noun, CAN would learn that 'big window' and 'window' were two nouns that have similar usage. Thus, it is highly probable that the representation of 'big window' and the representation of 'window' eventually become the same vector.

Once they become the same, all the composer network has to do is ignore the adjective part in the input layer and just copy the noun representation to the output layer as it is.

The argument above stands for any combination of a noun and an adjective. This is obviously unsatisfactory behaviour.

To prevent such inappropriate learning, it is necessary to make sure that the composed noun phrase representation contains both the noun information and the adjective information. Thus we pose the following restriction on the noun phrase representation: it must be possible to reproduce both the noun representation and the adjective representation from the composed noun phrase representation.

To satisfy this restriction, we add yet another neural network, called *the decomposer network*. After the vectorial representations (and the link weights) are updated by CAN and the composer network, the composer network and the decomposer network are connected tandem to make a five-layer neural network (Figure 5). The task to learn here is auto-association; namely reproducing the input words at the output layer. If the composer network ignores the adjective part, it is impossible to reproduce the perfect input pattern. Therefore the learning should go on so that the composed noun phrase representation contains both the noun information and the adjective information.

3 Simulation and result

We added six adjectives ('big', 'small', 'old', 'young', 'tall' and 'short') to the training corpus used in Takahashi and Motoki's work [5] to train our three neural networks. For each input, we generated the training sentence on the fly. The procedure of sentence generation was as follows.

1. Select a sentence pattern from Table 1 at random.
2. Replace each meta-word in the selected pattern with one of its instances listed in Table 2 to make an input sentence. Instances are selected also at random.
3. To each noun in the generated sentence, add an adjective according to the probabilities shown in Table 3.

The parameters of the neural networks were set as follows: CAN has ten hidden units; both the composer and the decomposer networks have five hidden units; in all networks, the activation function of hidden units is the logistic function $f(x) = 1/(1 + e^{-x})$, and the activation function of the output units is the linear function $g(x) = x$; a word representation is a five dimensional vector and the values of its elements are limited within ± 1; a noun phrase representation, i.e., output of the composer network, is also five dimensional, but the values of elements are not

	input pattern				target output		
No.	subject	verb	object	with-PP	subject	object	with-PP
1	HUMAN	ate			agent		
2	HUMAN	ate	FOOD		agent	patient	
3	HUMAN	ate	FOOD	FOOD	agent	patient	modifier
4	HUMAN	ate	FOOD	UTENSIL	agent	patient	instrument
5	ANIMAL	ate			agent		
6	PREDATOR	ate	PREY		agent	patient	
7	HUMAN	broke	FRAGILE		agent	patient	
8	HUMAN	broke	FRAGILE	BREAKER	agent	patient	instrument
9	BREAKER	broke	FRAGILE		instrument	patient	
10	ANIMAL	broke	FRAGILE		agent	patient	
11	FRAGILE	broke			patient		
12	HUMAN	hit	THING		agent	patient	
13	HUMAN	hit	HUMAN	POSSESSION	agent	patient	modifier
14	HUMAN	hit	THING2	HITTER	agent	patient	instrument
15	HITTER	hit	THING		instrument	patient	
16	HUMAN	moved			agt+pat		
17	HUMAN	moved	OBJECT		agent	patient	
18	ANIMAL	moved			agt+pat		
19	OBJECT	moved			patient		

Table 1: Sentence patterns used for the training. Symbols in SMALL CAPS are meta-words and replaced by one of the nouns listed in Table 2. For example, template No. 8 generates, among others, "(The) girl broke (the) window (with a) hammer", and its target output pattern is [subject = agent, object = patient, with-PP = instrument].

limited; noun representations are updated only at CAN's input layer and are left unchanged in the composer and the decomposer networks.

Table 3 shows vectorial representations of words and noun phrases that were generated after the networks had been trained with 10,000,000 sentences. A quick investigation reveals the following facts.

- The adjective 'tall' doubles the second, the fourth and the fifth elements of the following noun.
- The adjective 'old' reverses the sign of the first element.
- The adjectives 'big' and 'small', which accompany the same nouns with the same probabilities, have almost the same representations.
- The difference between the representations of 'man' and 'boy' is more prominent than in the preceding study [5]. We consider that the reason for this should be attributed to the difference of the distribution of adjectives that modify these two nouns.

Figure 6 shows the result of principal component analysis of the generated representations. Noun phrases that contain 'old' and that contain 'tall' clearly form their own clusters. Noun phrases containing 'big' seem to be spread over the plain, but their second principal components (the horizontal axis in the figure) have similar values. This result implies that the information of the adjective in a noun phrase is encoded as the second principal component.

4 Summary and future work

By adding two other neural networks to Takahashi and Motoki's CAN, we succeeded in applying Miikkulainen's extended backpropagation for noun phrases that contain multiple words. As a result, vectorial representations not only for simple words but also for noun phrases were generated. The result of principal component analysis implied that the information of the adjective in a noun phrase is encoded as a principal component.

We are now planning to conduct another experiment in which adjectives affect the case-role of the head nouns. For example, the noun 'bat' is ambiguous, but not in 'young bat' nor in 'wooden bat'; the former should be a flying animal and the latter should be an instrument used in ball games. New interesting features are expected to be found with adjectives that disambiguate polysemous words.

References

[1] Hindle, D.: Noun Classification from Predicate-Argument Structures, *Proceedings of the 28th Annual Meeting of the ACL*, pp. 268–275, 1990.

[2] Miikkulainen, R. and Dyer, M. G.: Natural Language Processing With Modular PDP Networks and Distributed Lexicon, *Cognitive Science*, Vol. 15, pp. 343–399, 1991.

meta-word	nouns
HUMAN	man woman boy girl
ANIMAL	bat chicken dog sheep wolf lion
PREDATOR	wolf lion
PREY	chicken sheep
FOOD	chicken cheese pasta carrot
UTENSIL	fork spoon
FRAGILE	plate window vase
HITTER	bat ball hatchet hammer vase paperweight rock
BREAKER	bat ball hatchet hammer paperweight rock
POSSESSION	bat ball hatchet hammer vase dog doll
OBJECT	bat ball hatchet hammer paperweight rock vase plate window fork spoon doll pasta cheese chicken carrot
THING	HUMAN ANIMAL OBJECT
THING2	ANIMAL OBJECT

Table 2: Meta-words and the nouns that replace the meta-words.

noun	adjective	probability
cheese, pasta carrot	ϕ	1.0
boy, girl	tall, short	0.2
	ϕ	0.6
man, woman	tall, short, old young, ϕ	0.2
dog, sheep wolf, lion	big, small, old young, ϕ	0.2
others	big, small	0.2
	ϕ	0.6

Table 3: Possible adjectives for each noun and their probability. ϕ means 'no adjective'.

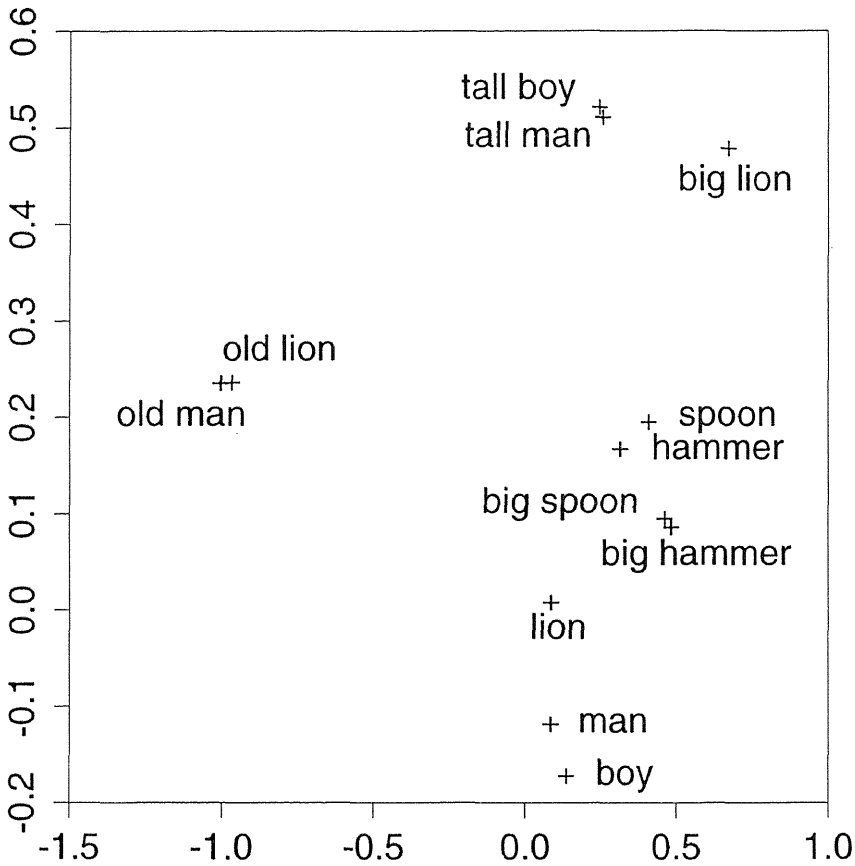

Figure 6: Result of principal component analysis of the vectorial representations. The horizontal axis and the vertical axis represent the second and the third principal components, respectively.

boy	-0.65	-0.33	-0.81	0.79	-0.98
tall boy	-0.61	-0.78	-0.63	1.14	-1.85
man	-0.57	-0.30	-0.78	0.69	-0.99
tall man	-0.62	-0.77	-0.62	1.14	-1.83
old man	0.58	-0.38	-1.05	0.23	-0.73
lion	-0.47	-0.28	-0.62	0.41	-0.91
big lion	-1.08	-0.81	-0.55	1.58	-2.23
old lion	0.59	-0.37	-0.97	0.11	-0.60
hammer	-0.26	-0.32	0.09	-0.92	0.33
big hammer	-0.41	-0.30	0.17	-0.91	0.44
spoon	-0.32	-0.37	0.15	-0.91	0.34
big spoon	-0.38	-0.30	0.18	-0.96	0.48
big	-0.51	-0.23	-0.52	-0.10	0.40
small	-0.52	-0.22	-0.52	-0.10	0.40
old	0.01	-1.00	-0.05	-1.00	-1.00
tall	-0.59	-0.37	-0.59	-0.31	-0.04

Table 4: Generated word representations and composed noun phrase representations (excerpt).

[3] Miikkulainen, R.: *Subsymbolic Natural Language Processing*, MIT Press, 1993.

[4] Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: Learning Internal Representations by Error Propagation, in Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* Volume 1, pp. 318–362, The MIT Press, Massachusetts, 1986.

[5] Takahashi, N. and Motoki, M.: A Subsymbolic Approach for Acquiring Semantic Representations, *Third International Workshop on Computational Semantics*, Tilburg, 1999.

国語辞書を用いた意味ネットワークの自己組織化に関する基礎検討

平井有三

1 はじめに

言語活動など我々の高度な認知活動は、構造化された記憶により支えられている。脳内で、実際にどのように構造化され、どのように利用されているかについては知る由もないが、自然言語に関する情報処理様式を科学的あるいは工学的に意味ある形で表現するためには、記憶の構造をモデル化する必要がある。意味ある形にするためには、オモチャの世界をモデル化したのでは不十分である。実世界にある知識体系を記憶構造に反映させる必要がある。

最近急速に普及してきた電子化辞書は、人間の英知を傾けて作られた知識体系であり、記憶構造、特に意味記憶の構造に関するモデル化を行う場合の基本的なデータ、すなわち重要な知識源として利用できる可能性がある。この可能性を検証することを目的として、国語辞書 (広辞苑第 4 版 [1]) を用いて基礎的な検討を行うためのシステム環境整備を行ってきた。その中には、

- 電子化辞書から語義文を抽出するためのソフトウエアの開発
- 形態素解析のパブリックドメインソフトウエアである茶筅の利用技術の修得
- 辞書情報の統計的解析のためのソフトウエアの開発

等が含まれている。

今年度は、これらのシステムを用いて国語辞書の利用可能性に関する予備的で小規模な検証を行った。検証には、類語辞典 ([2]) を用いて選んだ 41 個の形容詞を用いた。分類が明確な言葉を、語義情報を用いて正しく分類できるか否かを検証することが目的である。形容詞の語義文を形態素解析し、名詞と形容詞を抽出し特徴要素とした。41 個の形容詞と抽出された 283 個の特徴要素を用い、数量化 III 類により 41 個の形容詞の数量化を行った。その結果、語義文から抽出された特徴要素により、代表的な反義語関係にある形容詞の数量化が行えることが分かった。

2　語義文の抽出

電子化辞書の情報を利用するためには、語義文から必要な情報を抽出する必要がある。語義文の一例として、形容詞の見出し語「たかい」の語義文を表1に示した。語義文を抽出するためには、

- 見出し語の処理
- 語義文の処理

に分けて行う必要がある。

見出し語は仮名表記で、語幹と語尾が「・」で区切られている。また、語構成を示すために「あん - ぜん」のように「 - 」で区切られた仮名表記もある。これらの区切り記号は、語義文中の用例で省略されている語幹を補完するための手がかりとして用いることができる。語義文の形態素解析を行うためには、例えば「一・い山」を「高い山」に再構成する必要がある。

語義文は、^E 記号で語義の区分を行っている。E の後の数字が中分類を表しており、表の例では 5 つの語義がある。語義はさらに (1) などで小分類が、和数字記号で大分類されている。本研究では中分類を単位とし、その 1 番目の語義を見出し語の語義として利用している。形態素解析不要な小分類記号の除去、古典からの引用（例えば、“源若紫「日一・う寝起き給ひて」。”）の除去、先に示した現代語用例の省略の復元など、語義文を抽出し形態素解析を行うために必要な前処理は極めて多岐にわたる。

3　形容詞を対象とした統計解析

国語辞書を用いた意味ネットワークの自己組織化へ向けてその利用可能性を検証するために、類語辞典 ([2]) を用いて選んだ形容詞の解析を行った。反義語などの関係が、語義文から正しく検出できるか否かの検証がその目的である。「性状」の大分類から関連性のある 41 個の形容詞を選んだ。

これらの形容詞の語義文を切り出し、奈良先端科学技術大学院大学で開発された茶筅を用いて形態素解析を行い、名詞と形容詞を抽出した。表 2 から 4 に、類語辞典による区分と各形容詞、およびそれらから抽出された名詞と形容詞を示した。まだ一部に不完全なところがあるが、名詞と形容詞に関してはすべて抽出できている。基本的な形容詞の語義文から抽出された特徴要素の数と、それから派生した形容詞

たか・い【高い】	^E『形』■たか・し(ク)
	^E1 空間的な位置が上方にあって下との距離が大きい。(1) 上方へ長く突き出ている。そびえている。記中「—・き地に登りて西の方を見れば国土は見えず唯大海のみ有り」。「—・い山」(2) 上にある。遥か上方にある。源若紫「日—・う寝起き給ひて」。風雅秋「秋風の—・きみ空は雲はれてつきのあたりに雁のひとつら」。「空—・くのぼる」(3) 丈が長い。源桐壺「闇にくれて臥し沈み給へる程に、草も—・くなり、野分にいとどあれたる心地して」。「背が—・い」「—・い鼻」(4) 幅・深さなどが大きい。伊勢「ひえの山のふもとなれば雪いと—・し」(5) 高価である。狂、末広がり「是又—・いことでござる。ちつとねぎりませう」。日葡「アタイタカイモノ」
	^E2 物事の程度が他より甚だしくすぐれている。ある基準をこえている。(1) 高貴である。身分・地位がすぐれている。源帚木「人の品—・くうまれぬれば」。「—・い位につく」(2) すぐれている。立派である。「眼が—・い」「格調が—・い」(3) 数値が大きい。「温度が—・い」
	^E3 音・声や評判にいう。(1) 音域が上の方である。振動数が多い。「—・い声」(2) 大きい。強い。よくひびく。万四「風—・く辺(へ)には吹けども妹がため袖さへぬれて刈れる玉藻そ」。源少女「鳴り—・し。鳴りやまむ」(3) ひろく世間に知られている。源桐壺「先帝の四の宮の御かたちすぐれ給へる聞え—・くおはします」。「悪名が—・い」
	^E4 時間が多く経過している。(1) 年齢が多い。年長(た)けている。顕宗紀「諸(もろもろ)の老賢(としたかきさかしきひと)に聞きき」(2) 時間が遠い。続後拾遺雑「よしの川よしとは誰か岩波の—・きむかしの道したへども」
	^E5 射芸で、張弓の弦と弓との間の距離が広い。ふとい。—○高きに登る

表 1: 「たかい(高い)」の語義文。

の特徴要素の数の間に大きな差が見られる。また、反義語は「ない」で定義されているものが多く、解析した語義文約250文中50文が「ない」を含むものであった。「ない」の取り扱いは、今後の課題である。

また、関連する形容詞間の差異は副詞で表現されることが多いので、名詞と形容詞に加えて副詞も抽出する必要がある者と考えられる。

4 数量化III類による解析

表2から4に示したように、各形容詞が、その語義文から抽出された名詞と形容詞の特徴要素によって特徴付けられるものと考える。すなわち、形容詞間の関係がこれらの特徴要素の有無に埋め込まれているものとする。このように量的でないデータから物事の間の量的な関係を抽出する統計的手法に、数量化III類と呼ばれる手法がある(例えば、[3])。特徴の有無を示す41行283列の配列から得られる固有値問題を解くことにより、形容詞と特徴要素の数量を得ることができる。得られた40個の固有値(41個の固有値の内一つは無意味な解を持つ)の分布を図1に示した。最初の5つの固有値は1であり、後述するように抽出された特徴要素がその形容詞だけで使用され、他の形容詞と無関係の場合に生じる。残りの固有値は0.95から0.35程度の間に均一に分布している。

固有値の一つに対する固有ベクトルの一例を図2に示した。図には38番目の固有値に対する固有ベクトルを示してある。横軸の0から40までが各形容詞に、縦軸がそれらの数量を表している。この場合、数量の最大値を示した形容詞は「ちいさい」で最小値が「おおきい」である。したがって、おおざっぱに言ってこの軸は大小関係を表す軸と言えるかも知れない。各固有値から決まる40個の軸にすべてにおける数量の大小関係を表すために、各数量のメディアンを中心とする標準偏差を求め、その3倍を越える値に+または−を、最大値と最小値にそれぞれ++と==を与えて表5と6に示した。

表の5番目までの軸は、抽出された特徴要素がその形容詞のみに固有のものの場合である。「ひらべったい」「まぢかい」「かぼそい」「まるっこい」「ひょろながい」などは他の言葉と無関係に定義されているということである。

特徴要素の数が少ない形容詞は多くの軸で大きな値を持ち、変動が大きく、大きな固有値を持つ固有ベクトルで主要な(値の大きな)特徴要素となっている。すなわち、これらは語義文からの特徴付けが困難な形容詞群と考えられる。「こだかい」

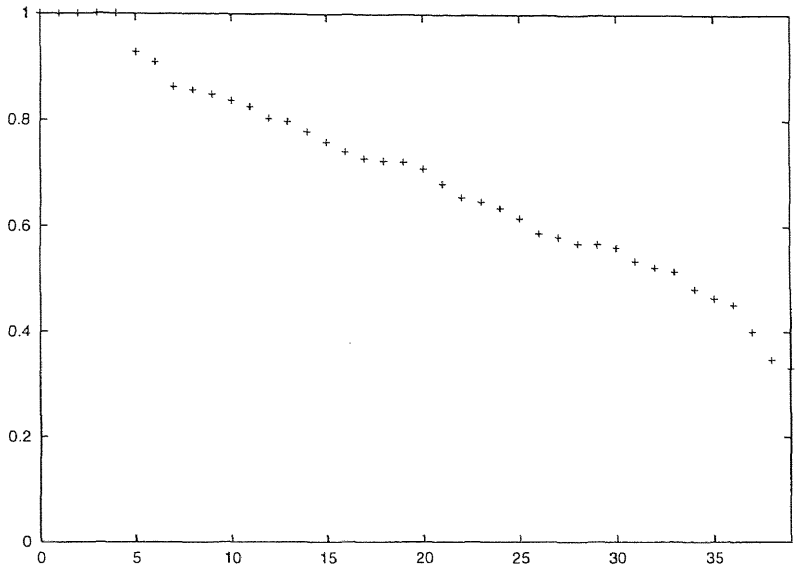

図 1: 40 個の固有値の分布。

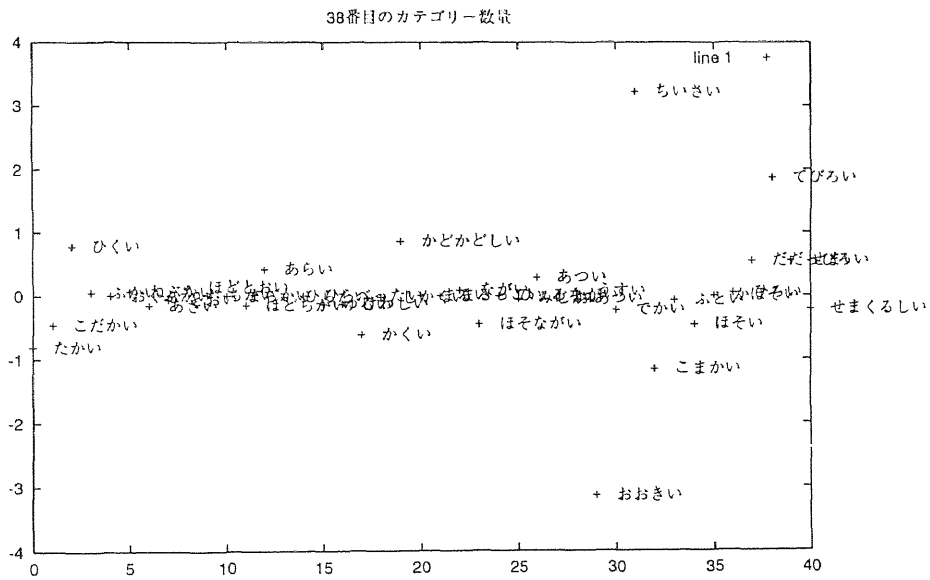

図 2: 38 番目の固有値に対する固有ベクトル。横軸が各形容詞に対応し、縦軸がそれらの数量を表す。「小さい」と「大きい」の関係を表す軸と考えられる。

「あらい」「かくい」「だだっぴろい」などがこのグループにはいる。

これに対して、特徴要素の数が多く十分に定義されている形容詞は、固有値の小さな固有ベクトルで大きな値をとる傾向がある。また、特徴に重なりのある反義語など他の形容詞と軸を特徴づける最大値、最小値の組を作る場合が見られる。「しかくい」「まるい」、「おおきい」「ちいさい」、「ほどとおい」「ほどちかい」などがその例である。これらはいずれも最小の3つの固有値に対する固有ベクトルの主要な構成要素である。したがって、数量化III類を用いた解析では、小さな固有値を持つ固有ベクトルに、言葉の分類にとって重要な情報が含まれていると考えられる。

5 まとめ

国語辞書を用いた意味ネットワークの自己組織化に向けて、語義文を切り出し、形態素解析を行い、名詞と形容詞を抽出するプログラムを開発した。類語辞典を用いて選択した41個の形容詞を用いて数量化III類による統計的な解析を行った。その結果、語義文から言葉の意味関係に関する情報を取り出せることを確認した。

今後、他の形容詞に解析を広めて有効な特徴要素に関する調査を行うほか、名詞や動詞も解析の対象として行く予定である。

なお、広辞苑第4版の利用を快諾して頂いた岩波書店殿に感謝する。

参考文献

[1] 新村出（編）. 広辞苑第4版. 岩波書店, 1991.

[2] 大野晋, 浜西正人. 類語新辞典. 角川書店, 1981.

[3] 田中豊, 脇本和昌. 多変量統計解析法. 現代数学社, 1983.

本研究に関連する1999年度公表論文

1. 中村眞、平井有三：「空間周波数を用いた透視投影画像の3次元形状復元モデル」電子情報通信学会技術研究報告、NC98-185、pp.247-252、1999年3月
2. Masaki Kawamura, Masato Okada and Yuzo Hirai: "Dynamics of selective recall in an associative memory model with one-to-many association." IEEE Transactions on Neural Networks, Vol.10, No.3, pp.704-713, 1999.

3. 西澤邦宜、平井有三：「連続時間 PCA 学習ハードウエアの実時間動作」日本神経回路学会第 9 回全国大会講演論文集、pp.14-15、1999 年 9 月
4. 勝本秀之、平井有三：「PDM ディジタルニューラルネットワークシステムの並列鋳型照合課題による性能評価」日本神経回路学会第 9 回全国大会講演論文集、page 54、1999 年 9 月
5. Seiichi Tsujimura, Satoshi Shioiri, Yuzo Hirai and Hirohisa Yaguchi: "Selective cone suppression by the L-M- and M-L-cone-oponent mechanisms in luminance pathway." Journal of the Optical Society of America, A, Vol.16, No.6, pp.1217-1228, 1999.

区分	形容詞	抽出された名詞と形容詞
上下	たかい	空間、位置、上方、下、距離、大きい、上方、長い、上、遥かだ、上方、丈、長い、幅、深さ、大きい、高価だ、日
	こだかい	こだかい、高い
	ひくい	高い、少ない、丈、短い、地位、身分、程度、数値、小さい、音声、高い
入り口	ふかい	表面、底、距離、長い、浅い、外、内、距離、遠い、密生、繁い、色、味、濃い、たけなわだ、軽率だ、重々しい、あさはかだ、かりそめ、交わり、浅い、親交、物事、奥底、程度、甚だしい、男女、ない、関係
	おくぶかい	奥、深い、意味、深い、深遠だ
	ねぶかい	根、深い、深い、物事、原因、古い、から、ところ、深い、容易だ、しつこい、執念深い、一通り
	あさい	表面、外面、その日、経過、地位、低い、程度、軽い、色、香、かすかだ、深遠だ、所、思慮、知恵、未熟だ、深み、ない、情、あつい、趣、乏しい
遠近	とおい	空間、大きい、距離、はるかだ、時間、大きい、久しい、長い、仲、親しい、うとい、交渉、少ない、血縁、関係、うすい、関係、あまり、ない、切実だ、よい、深い、遠い、近い、憂い、○、遠い、○、遠い、花、香、近、糞、香、○、遠い、近、男女、仲、○、遠く、火事、背中、灸、○、遠く、親類、近く、他人
	ほどとおい	みちのり、時間、相当だ、遠い、状況、
	ちかい	距離、少ない、遠い、日、時間、少ない、日、よい、心、心、少ない、親しい、血縁、遠い、身寄り、身内、物事、形状、内容、性質、数量、近視
	まぢかい	まぢかだ
	ほどちかい	ちか、みちのり、時間、近い、あまり、遠い

表 2: 形容詞の 1 番目の語義から抽出された名詞と形容詞 (その 1)

区分	形容詞	抽出された名詞と形容詞
面	あらい	ごつごつ 、けわしい 、はげしい 、すさまじい 、乱暴だ 、あらあらしい 、あらっぽい 、程度 、はなはだしい
	ひらたい	厚い 、少ない 、面 、広い 、ところ 、ない 、平坦だ 、凹凸、ない 、平易だ
	ひらべったい	平らだ 、扁平 、ひらたい
	ゆるい	ゆるやかだ 、すきま 、勢い 、弱い 、急速だ 、のろい 、寛大だ、きびしい 、濃度 、小 、薄い 、勾配 、傾斜 、曲線
	けわしい	傾斜 、急だ 、困難だ 、険阻だ 、困難だ 、荒い 、はげしい、非常だ 、はげしい 、とげとげしい 、あわただしい 、忙しい
角	かくい	四角い 、かどだ
	しかくい	四角だ
	かどかどしい	かどだ 、多い 、性格 、態度 、かどだ 、とげとげしい
	まるい	円形 、球形 、かどかどしい 、穏やかだ 、所 、ない 、円満だ、角 、ない 、ふっくら 、丸い 、卵 、よう 、四角だ
	まるっこい	丸み 、まるい
長短	ながい	空間 、時間 、一点 、他 、点 、大きい 、抽象的だ 、事 、永久、変だ 、長い 、目 、○ 、長い 、物 、○ 、長い 、草鞋
	ほそながい	ほそながい 、ほそい 、ながい 、狭い 、長い
	ひょろながい	弱々しい 、細長い
	みじかい	長さ 、少ない 、高い 、少ない 、低い 、こと 、少ない 、久しい 、日 、位 、低い 、考え 、浅い 、かしこい 、性急だ 、短慮だ 、日
	あつい	物体 、一面 、対面 、大きい 、はなはだしい 、深い 、恩恵 、情愛 、良い 、場合 、多い 、悪い 、場合
	ぶあつい	相当 、厚み
	うすい	厚み 、少ない 、密度 、濃度 、少ない 、液体 、霧 、密度 、少ない 、色合い 、色 、けが 、少ない 、あわい 、まばらだ 、味 、濃い 、淡泊だ 、物事 、程度 、強い 、量 、少ない 、深い 、浅い 、軽微だ 、光、強い 、あわい 、弱い 、つたない 、乏しい 、貧しい 、少ない

表 3: 形容詞の 1 番目の語義から抽出された名詞と形容詞 (その 2)

区分	形容詞	抽出された名詞と形容詞
大小	おおきい	容積、身長、多く、場所、かさ、量、多い、程度、はなはだしい、ひどい、範囲、広い、規模、包容、度量、年上、大げさだ
	でかい	大きい、意、俗語、甚だしい、でっかい
	ちいさい	容積、身長、場所、少ない、大きい、年、おさない、声音、弱い、かすかだ、些細だ、単位、下、度量、狭い、規模、大きい、卑下、状態
	こまかい	形、単位、小さい、ほど、小さい、些細だ、瑣末だ、小さい、ところ、綿密だ、勘定高い
	ふとい	まわり、長さ、横、幅、大きい、肉、豊かだ、安定、横着だ、ずうずうしい、声、低め、音量、ゆたかだ、太い、短い
	ほそい	周囲、小さい、幅、広い、狭い、低い、小さい、量、少ない、僅かだ、弱い、かすかだ、芸、張、弓、弦、弓、間、距離、狭い、ひくい、細い、長い、○、細い、長い
	かぼそい	かぼそい
	ひろい	面積、大きい、場所、ゆとり、多い、頻繁だ、物事、範囲、大きい、すみずみ、ふかい、ゆるやかだ、おおようだ、日
	だだっぴろい	やたらだ、広い
	てびろい	かまえ、広い、てぜまだ、規模、関係、範囲、広い
	せまい	面積、幅、小さい、ゆとり、ない、広い、ゆとり、ない、ゆるやかだ、窮屈だ、狭量だ、狭い、門
	せまくるしい	せまい、窮屈だ、せせこましい、くるしい

表 4: 形容詞の１番目の語義から抽出された名詞と形容詞（その３）

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
たかい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
こだかい	0	0	0	0	0	0	0	0	0	−	0	0	0	0	++++	+	++++	==		
ひくい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ふかい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
おくぶかい	0	0	0	0	0	0	0	0	0	0	0	0	0	+	0	+	−	+	==	0
ねぶかい	0	0	0	0	0	0	0	0	0	+	==	0	==	++	0	0	0	0	0	0
あさい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	0	0	−	0
とおい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ほどとおい	0	0	0	0	0	0	0	+	0	0	+	0	0	0	0	0	+	−	0	0
ちかい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
まぢかい	0	++	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ほどちかい	0	0	0	0	0	0	0	0	0	0	+	0	0	0	0	0	+	−	0	0
あらい	0	0	0	0	0	+	0	0	0	0	0	++	0	0	0	0	+	0	0	++
ひらたい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ひらべったい	++	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ゆるい	0	0	0	0	0	+	0	−	0	0	0	−	0	0	0	0	0	0	0	0
けわしい	0	0	0	0	0	+	+	0	0	0	0	+	0	0	+	0	−	0	0	0
かくい	0	0	0	0	0	++	+	++	−	+	0	==	0	0	0	0	+	0	0	+
しかくい	0	0	0	0	0	==++		0	0	0	0	0	0	0	0	0	0	0	0	0
かどかどしい	0	0	0	0	0	+	+	+	−	0	0	−	0	0	0	0	0	0	0	0
まるい	0	0	0	0	0	−	+	0	0	0	0	0	0	0	0	0	0	0	0	0
まるっこい	0	0	0	==	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ながい	0	0	0	0	0	0	0	0	0	0	+	0	0	0	0	−	0	+	0	+
ほそながい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	0	+
ひょろながい	0	0	0	0	++	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
みじかい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	0	0	+	0
あつい	0	0	0	0	0	0	0	0	0	0	0	+	0	0	==	0	0	0	0	−
ぶあつい	0	0	0	0	0	0	0	−	++	0	0	−	++	+	0	==++		0	+	−
うすい	0	0	0	0	0	0	0	0	0	0	0	−	+	0	0	0	0	0	0	0
おおきい	0	0	0	0	0	0	0	0	0	0	0	0	0	−	0	0	0	0	0	0
でかい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	0	====		+	0
ちいさい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
こまかい	0	0	0	0	0	0	0	0	0	0	−	0	−	0	+	0	0	0	0	0
ふとい	0	0	0	0	0	0	==	0	−	==	0	0	0	0	0	0	0	0	0	0
ほそい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
かぼそい	0	0	++	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ひろい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
だだっぴろい	0	0	0	0	0	0	0	0	0	0	0	0	0	==	0	0	0	0	0	0
てびろい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
せまい	0	0	0	0	0	0	0	0	−	0	0	0	0	0	0	0	0	0	0	0
せまくるしい	0	0	0	0	0	0	0	====		++++		+	0	+	0	0	0	0	0	0

表 5: 数量化III類による解析結果（その１）

	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
たかい	0	0	0	0	==	0	+	==	0	0	++	0	−	0	+	+	0	−	−	0
こだかい	0	0	+	0	+	−	++	0	==	0	0	0	0	0	++	+	+	0	0	0
ひくい	0	0	0	0	0	0	+	−	0	0	0	0	0	0	==	−	−	+	0	0
ふかい	0	0	0	0	0	0	+	0	0	0	0	0	0	0	0	0	0	0	0	0
おくぶかい	−	0	++	==	−	==	0	0	0	0	0	0	0	0	0	0	−	0	0	0
ねぶかい	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
あさい	0	0	0	0	0	++	0	0	0	0	0	0	0	0	+	0	+	0	0	0
とおい	0	0	0	0	0	0	0	+	0	0	0	0	0	0	0	0	0	0	+	0
ほどとおい	0	0	0	0	++	0	−	−	0	0	0	0	0	0	0	0	0	0	++	0
ちかい	0	0	0	0	−	0	0	0	0	0	−	+	0	0	0	0	0	0	−	0
まぢかい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ほどちかい	0	0	0	0	+	0	0	0	0	0	0	0	0	0	0	0	0	0	==	0
あらい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ひらたい	0	−	+	++	0	0	0	0	0	++	+	0	0	0	0	0	+	0	0	0
ひらべったい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ゆるい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
けわしい	0	0	0	0	0	0	0	0	0	0	0	0	0	+	0	0	0	0	0	0
かくい	0	0	0	0	0	0	0	0	0	−	0	++	+	++	0	0	0	−	0	0
しかくい	0	0	0	0	0	−	0	0	0	0	0	0	0	0	−	0	++	0	0	0
かどかどしい	0	0	0	0	0	0	0	0	0	+	0	==	0	==	0	0	0	+	0	0
まるい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	==	0	0	0
まるっこい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ながい	0	−	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ほそながい	+	++	0	0	+	0	0	0	0	+	0	0	−	0	0	0	0	0	0	0
ひょろながい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
みじかい	0	0	0	0	0	0	==	+	+	0	0	0	0	0	0	0	0	0	0	0
あつい	−	+	0	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ぶあつい	0	0	+	0	−	+	+	+	0	0	0	0	+	0	0	0	0	0	0	0
うすい	0	0	0	0	0	−	0	0	0	0	0	0	0	0	0	0	0	0	0	0
おおきい	0	−	0	−	0	0	0	0	0	0	0	0	0	0	0	0	0	==	+	+
でかい	0	0	+	0	0	0	−	0	0	0	0	0	0	0	0	0	0	0	0	0
ちいさい	0	0	0	−	0	0	0	0	0	0	0	0	0	0	+	−	0	++	0	0
こまかい	==	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	0	0
ふとい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ほそい	0	0	0	0	0	0	0	0	0	−	0	0	++	0	0	0	0	0	−	−
かぼそい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ひろい	0	0	0	0	0	0	0	0	0	+	−	−	0	+	+	0	+	0	0	==
だだっぴろい	++	==	+	+	+	−	+	0	++	==	==	+	==	−	+	==	0	0	0	−
てびろい	+	−	0	0	0	0	+	++	0	0	0	0	0	+	−	++	−	+	−	0
せまい	0	0	0	0	0	0	0	0	0	0	0	−	0	+	0	0	0	0	+	++
せまくるしい	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−

表 6: 数量化 III 類による解析結果（その 2）

常体から敬体への言い換えシステム

横山　晶一（山形大学）

1．はじめに

日本語の尊敬語、謙譲語、丁寧語などの、いわゆる待遇表現は、日本語を特徴づける言語現象の一つである。特に、対話を行う際にこれらの待遇表現を正確に使いこなすことができれば、自然な対話システムが構築できると考えられる。

待遇表現については、これまで、敬語論といった形で、言語学的に考察したものはいくつかある[1,2,3]が、機械処理の観点から取り扱った研究は非常に少ない[4]。予備的な調査によれば、尊敬語や謙譲語をうまく使用できるシステムを構築するためには、対話者間の社会的な相対的地位関係をうまく取り入れる必要がある。また、そのための準備として、尊敬語や謙譲語の性質自体を調査検討する必要もある。

そこで、本研究では、待遇表現を扱うことのできるシステムを構築するための準備段階として、常体を敬体に変換するシステム用のアルゴリズムを構築する[5]。これらを組み込んだワープロソフトがすでにいくつか存在するが、その中でどのようなアルゴリズムが働いているかは明らかではない。ここで構築したアルゴリズムを手作業で実際の新聞記事に適用した。その結果、このアルゴリズムでおおむね良好な書き換えが行われることが分かったが、なおいくつかの問題点が明らかになった。現在実際のシステム構築に向けて、プログラムを作成中である。また、最終目標である待遇表現に関してもある程度の知見が得られた。以下ではこれらについて報告する。

2．研究背景

上に述べたように、待遇表現については、言語学的観点から種々の研究がなされている[1,2,3]が、その多くは、敬語を分類したり、敬語の文法的な役割について考察しているだけではなく、敬語が使われるときの文化的背景や相対的地位についてまで言及している。現在の機械処理の中にこれらの視点を導入することは困難である。

一方、機械処理の観点で待遇表現を扱ったものとしては、杉村の研究がほとんど唯一のものである[4]が、この研究では、上下や親疎の関係を数値化し、単語検索、パターンマッチング、単一化等を行うことによって、文章を解析し、

待遇表現を取り扱おうと試みている。

本研究[5]では、話し言葉もある程度視野には入れるが、実際には、常体で書かれた（書き言葉の）文を自動的に敬体に変換するためのアルゴリズムを作成する。すなわち、「だ、である」の形の文を、「です、ます」調の文に変換するアルゴリズムを作成し、そのアルゴリズムに基づいて、手作業で実際の文を変換し、考察する。

３．変換規則

３．１．文末以外の変換

変換は、ほとんどの場合、文末のみについて行う。話し言葉以外では、文末以外で変換した方がよい場合は少ない。ここでは、従属句等への「です、ます」のつながり方[6]を考慮して、文末以外では、次の場合のみ書き換えを行う。

・「が、と、から」の前は敬体に変える。

（例）運転手は必死に沿道の建物の人に 119 番通報を<u>頼みます</u>が、電話が不通だと言われます。

３．２．接続詞

接続詞や副詞的な接続語句は、一般的には書き換えを行わないが、次のものに限って、変換する。

・「だが」→「ですが」、「だけど」→「ですけど」

それ以外のものを変換すると、過剰になる。

３．３．その他の品詞

その他の品詞については、次のような規則を設ける。

・動詞：　終止形→連用形＋ます
　　「である」→「です」
・形容詞：　「ない」→「ありません」
　　その他の終止形→終止形＋です*
・形容動詞：終止形→語幹＋です
・助動詞（た、う、ようを除く）：
　「せる、させる、れる、られる」
　　→「せます、させます、れます、られます」
　「たい、らしい」→「たいです、らしいです」*
　「ようだ、そうだ、だ」→「ようです、そうです、です」

未然形＋ない→連用形＋ません

・た（活用の型で区別する）：

動詞型連用形＋た→動詞型連用形＋ました

形容詞型連用形＋た→形容詞型連用形＋たです（助動詞「ない」を除く）

なかった→ませんでした

だった→でした

・う（動詞五段活用未然形につく）：

動詞型未然形＋う→動詞型連用形＋ましょう

形容詞型未然形＋う→形容詞型終止形＋でしょう

だろう→でしょう

・よう（五段活用以外の動詞未然形につく）：

動詞型未然形＋よう→動詞型連用形＋ましょう

４．変換例

上の規則を順次適用するアルゴリズムを作って、実際の文章（天声人語）を手作業で書き直した。以下にその例を示す。

変換前

昔は、見知らぬ人同士も「失礼」「どうぞ」くらいのあいさつはしたものだ。今でも海外では、車中で一緒になったり、すれ違ったりする時に、あいさつ、目礼、微笑などに出会う。そこに人間の仲間がいるのだから交流があって当然、という感じである。

夏休みで、子供たちも見知らぬ人に接する場面が多い。見知らずとも人間同士、自然体の接し方を会得する好機だ。ひと声かけるのは緊急事態だけ、と決まったものではない。

変換後

昔は、見知らぬ人同士も「失礼」「どうぞ」くらいのあいさつはしたものです。今でも海外では、車中で一緒になったり、すれ違ったりする時に、あいさつ、目礼、微笑などに出会います。そこに人間の仲間がいるのだから交流があって当然、という感じです。

夏休みで、子供たちも見知らぬ人に接する場面が多いです。見知らずとも人間同士、自然体の接し方を会得する好機です。ひと声かけるのは緊急事態だけ、と決まったものではありません。

上の例では、文末のみの変換であるが、これでも丁寧語の文にすることはできている。ただし、「多い」→「多いです」のように、形容詞の終止形に「です」を付けるとやや不自然である。変換規則で＊を付した部分がそれである。

５．終わりに

以上のように、比較的簡単なアルゴリズムを用いて常体から敬体への変換システムを構築する準備が整った。このアルゴリズムでは、構文解析や意味解析を行わず、形態素解析ができていればほぼパターンマッチングで処理を行うことができるので、フリーソフトウェアの形態素解析システムを通した後に書き換えを行う実際のシステムを現在構築中である。

また、どの程度までを敬体にすれば過剰でないか、さらに、尊敬語や謙譲語をどのような形で取り入れていくかについては、今後の課題である。

謝辞

本研究は、山形大学工学部電子情報工学科４年生　羽根木規君の卒業研究[5]によるところが多い。ここに記して感謝する。

参考文献

[1] 大石初太郎：敬語、ちくま文庫 (1986)
[2] 南不二男：敬語、岩波新書 (1987)
[3] 菊地康人：敬語、講談社学術文庫 (1997)
[4] 杉村領一：日本語の待遇表現の解析と状況意味論、コンピュータソフトウェア Vol.3, No.4 (1986) pp.350-358
[5] 羽根木規：常体と敬体の言い換えシステム、山形大学卒業論文 (2000)
[6] 野田尚史：「は」と「が」、新日本文法選書１、くろしお出版 (1996)

多言語音声コーパスの設計と構築

西尾晋太郎、山本幹雄、板橋秀一

1.　はじめに

音声合成や音声認識に代表されるような音声研究を進めるにあたり、性別・年齢・方言・人数などにおいて多種多様な音声データは必要不可欠である。これまでは音声研究者が、各自で自分に必要な音声データを収集していたが、この作業には多大な労力と時間を要してしまう。また、近年各種の統計的手法の発達により、大量の音声データがシステム学習のために必要とされている。

一方、近年インターネットの爆発的な普及により海外の情報や資料が非常に身近なものになってきている。また、マルチメディア化社会に向けてのインターフェースとして音声による入出力が注目されるようになり、多言語による音声コーパスの構築は、近い将来グローバル化されてくるシステムの開発には欠かすことができないものになってきている。

近年日本でも数多くの音声コーパスの収録が行われてきたが、依然として海外に比べて大きな遅れをとっているのが現状である。特に多言語による音声コーパスに関する報告は数少ない。そこで本研究では、多言語の音声データの収録と、言語の違いを音声学的にしらべることを目的とし、言語の分類、また多言語による音声認識や言語識別システムなどの開発につながる多言語音声コーパスの構築を行う。

2.　既存の多言語音声コーパス

現在報告されている多言語音声コーパスの例を以下に示す。

（1）"Multilingual Speech Database for Teleponometry 1994"

NTT によって音声通信機器などの性能評価などを目的に作成された。ネイティブの一般人による短文読み上げのデータベースであり、ＣＤ－ＲＯＭ４枚に格納されている。収録言語は、英語（American , British)，アラビア語、中国語、オランダ語、フィンランド語、フランス語、ドイツ語、ギリシア語、ヒンドゥー語、ハンガリー語、インドネシア語、イタリア語、日本語、韓国語、ポーランド語、ポルトガル語、ロシア語、スペイン語、スウェーデン語、タイ

語の２１言語を収集している。話者数は各言語ともに男女４名ずつであり、全ての言語で異なった内容で１発話あたり２～３秒の短文章を発声する。

（２）"Multi-language Telephone Speech Corpus"

OGI(Oregon Graduate Institute)において、自動言語識別と多言語音声認識を目的に作成された。商用電話回線を用いて英語、ペルシア語、フランス語、ドイツ語、ヒンドゥー語、日本語、韓国語、中国語、スペイン語、タミール語、ベトナム語の１１言語が収集されている。一人あたりほぼ５分くらいの発声であり、内容は、話者の母語、曜日、０～１０までの数字を発声する語彙固定による発声、街の好きなところや天気など１０秒程度の発声で返答できる４種類の話題指定による発声、話者による１分程度の自発的な発声に分かれている。

３．　多言語音声コーパスの設計・収録

３．１　多言語音声コーパスの仕様

本研究で構築する多言語音声コーパスの特色は、各言語において共通の内容の発声を行う点と、防音室において高音質な音声データを収録する点にある。以下に多言語音声コーパスの仕様を示す。

（１）　収録言語

世界の人口や言語の分布、また筑波大学における外国人留学生の内訳などを考慮して、以下の１０言語の音声データを収録した。この選択により、世界の各地域において使用されている言語を獲得することができ、また東西における言語を比較するためにも利用可能である。

英語・ドイツ語・フランス語・スペイン語・ロシア語・アラビア語・中国語・韓国語・日本語・インドネシア語

（２）　話者数

個人差ではなく言語の違いを見るという意味で、各言語とも最低男女各２名以上の話者が必要である。話者は筑波大学に在籍する外国人留学生及び外国人教師を中心に選定した。

（３）　収録系

収録は、筑波大学知能情報・生体工学研究室内の防音室で行い、話者用のマイクロホンはヘッドセットマイクを、収録媒体にはＤＡＴ（ディジタル・オーディオ・テープ）を用いてテキスト読み上げによる収録を行った。使用した機器は以下の通りである。

・マイクアンプ；　SONY Stereo Microphone Amplifier MX-45
・ＤＡＴデッキ；　SONY DAT DECK DTC-300ES
・ヘッドセット；　SENNHEISER Headset 25-1
・ミキサー；　SONY AUDIO MIXER SRP-X6004
・ＤＡＴデッキ；　SONY DAT DECK DTC-2000ES
・ヘッドフォン；　SONY MDR-Z600
・マイク；　SONY ELECTRET CONDENSER MICROPHONE MP600Ω

（4）　発声内容

各言語において共通の内容の発声を行う。発声内容は、全世界において広く一般的に使用されているような単語として、表１に示すような数字１４語、月１２語、曜日７語、天気４語、挨拶６語、返答３語、１５分刻みの時間４語の合計５０語を選択した。連続音声には、誰でも知っていて、資料が容易に手に入る１分くらいの物語として、イソップ童話の「北風と太陽」を用いた。

表１：発声単語の日本語と英語の対応表

	日本語	English		日本語	English
１	いち	one	２６	じゅうにがつ	December
２	に	two	２７	げつようび	Monday
３	さん	three	２８	かようび	Tuesday
４	し	four	２９	すいようび	Wednesday
５	ご	five	３０	もくようび	Thursday
６	ろく	six	３１	きんようび	Friday
７	しち	seven	３２	どようび	Saturday
８	はち	eight	３３	にちようび	Sunday
９	きゅう	nine	３４	はれ	fine weather
１０	じゅう	ten	３５	くもり	a cloudy sky
１１	れい	zero	３６	あめ	rain
１２	ひゃく	one hundred	３７	ゆき	snow
１３	せん	one thousand	３８	おはよう	Good morning
１４	まん	ten thousand	３９	こんにちは	Good afternoon
１５	いちがつ	January	４０	こんばんは	Good evening
１６	にがつ	February	４１	おやすみなさい	Good night
１７	さんがつ	March	４２	ありがとう	Thank you
１８	しがつ	April	４３	すみません	I'm sorry
１９	ごがつ	May	４４	はい	Yes
２０	ろくがつ	June	４５	いいえ	No
２１	しちがつ	July	４６	もしもし	Hello
２２	はちがつ	August	４７	２じ	two o'clock
２３	くがつ	September	４８	１０じ１５ふん	quarter past ten
２４	じゅうがつ	October	４９	４じ３０ぷん	four thirty
２５	じゅういちがつ	November	５０	７じ４５ふん	quarter to eight

各言語の発声リストは日本語を基にして話者により作成してもらい、２通りの言い方がある単語に関しては標準的に使用されている方を選択してもらった。言語によっては朝、昼、夜の挨拶を１つの単語で表す言語もあり、発声単語数は言語により多少異なる。表２に言語別の発声単語数を示す。

表２：言語別の発声単語数の比較

単語数	言語
５０語	英語、ドイツ語、ロシア語、アラビア語、中国語、日本語、インドネシア語
４９語	フランス語、スペイン語
４８語	韓国語

３．２　音声データの収録

話者には、収録した音声データをＣＤ－ＲＯＭなどの電子的媒体に格納し、各研究機関や音声研究者に配布することへの承諾書と、年齢、性別、出身地、母語、居住履歴、両親の出身地などの発声者データを記入してもらい、各言語の発声リストをもとに防音室において収録をおこなった。今回の収録においては方言のある話者にも各言語の標準語とされている発声をおこなってもらい収録を行った。話者には、テスト収録も兼ねてまず単語５０語の発声を行った後、もう一度単語の発声を行い、最後に連続音声を発声する。連続音声に関しては、発声中に言い淀みや、誤った発声などをしてしまった場合には、誤りなく発声出来るまで何度も収録を繰り返した。発声時間は平均して１人５分くらいであった。

３．３　収録した音声データ

現在までに収録した音声データの言語別の内訳を表３に、また話者の年齢分布を図１に示す。筑波大学に在籍する外国人留学生を中心に収録を依頼したため、２０代と３０代に話者が集中している。

表３：音声データの言語別内訳

	男性	女性	合計
英語	1	1	2
ドイツ語	3	0	3
フランス語	3	2	5
スペイン語	1	3	4
ロシア語	1	1	2
アラビア語	2	2	4
中国語	4	5	9
韓国語	4	2	6
日本語	3	3	6
インドネシア語	1	0	1
合計	２３	１９	４２

（単位：人）

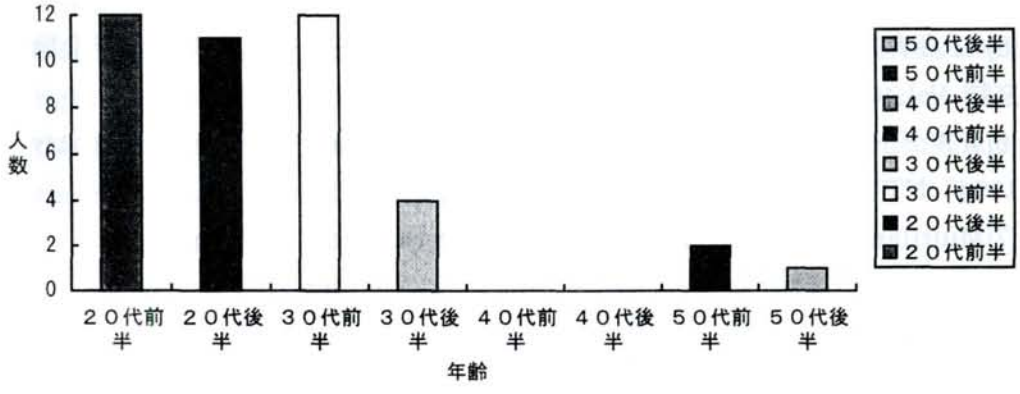

図１　話者の年齢分布

３．４　音声データの品質評価

発声リストとの相違、アクセントなどの発声上の特徴、雑音の混入、発声速度、発話の明瞭性などの観点からデータの品質を評価する必要がある。今後最低２名の各言語のネイティブな話者により、編集した音声データを上記のような観点から品質の評価を行っていく。

4．おわりに

本稿では、言語の分類や、多言語による音声認識・言語識別システムの開発を目的とした多言語音声コーパスの収録に関して記した。今後各言語において、特にロシア語、インドネシア語、ドイツ語などの話者を増やしていき、音声データを分析することで、言語の特色を音声学的に調べていく方針である

参考文献

[1] 板橋秀一“音声コーパス”「情報処理」第38巻　第11号 1977

[2] 上田直子、高木一幸、板橋秀一“テレフォンショッピング対話の収録と分析”日本音響学会講演論文集 ,2-Q-21 ,pp337-338 ,Mar.1995

[3] 木内俊一、山本幹雄、板橋秀一“基本周波数とケプストラムによる多言語音声の分類”情報処理学会研究報告 ,Vol.99,No.14,99-SLP-25,pp79-84 ,Feb.1999

[4] 板橋秀一“DAT 音声データベース「単音節・単語・文・文章（物語他）」”平成3年度　文部省科学研究費補助金試験研究報告書

[5] 工藤育男、中間崇夫“Voice Across Japan データベース”情報処理学会論文誌 Vol.40,No.9,pp3432-3445 1999

[6]音声データベース“Multi-Lingual Speech Database for Telephonometry 1994”NTT アドバンステクノロジ

[7]Multilanguage Telephone Speech : http://cslu.cse.ogi.edu/corpora/available/

付録

イソップ童話「北風と太陽」

（1） 日本語

あるとき、北風と太陽が力比べをすることになりました。「あそこを歩いていく、旅人の外套を脱がせた方を勝ちとしよう。」と決めてまず北風からはじめました。北風は自信満々に「ひとまくりしてみせよう。」といい、激しく吹きたてました。ところが、風が激しく吹けば吹くほど旅人は、外套をしっかりと押さえてしまいました。今度は太陽の番になりました。太陽が雲の間から暖かな光をおくると、旅人は汗をかき始めました。そこで、太陽が光を強くすると、旅人は「暑くなってきたなあ。」といって外套を脱ぎました。こうしてこの力比べは太陽の勝ちになりました。

（2） 英語

A long time ago the North Wind and the sun had a dispute as to who was stronger. A traveler passing by became the object of their conflict. They decided that whoever could remove his cloak would be the stronger. The North Wind blew and the cloak nearly came off. The man held his cloak tightly and it remained on him. Next the Sun shone fiercely. The heat from the sun warmed the man quickly and he removed his coat. In this manner the Sun proved to be the more powerful of the two.

実音声サンプル低依存型の音声認識方式

田中 和世(電子技術総合研究所)

1．はじめに

いわゆるグローバル化が進む今日では、われわれが話す言葉にも外国語あるいは外来語が混じることは日常的である。また、インターネットに代表される情報システムにおいては英語が事実上の世界標準になりつつあるが、このことは逆に、音声的にはきわめて分散の大きい英語（あるいは話者の母語が混入した英語）が世界中で使用されるということにもなる。このような状況は、音声の世界では多言語を扱うマルチリンガル(multi-lingual)処理ではなく、いわばmixed-lingualともいうべき処理の必要性が増大することを示唆している。音声認識や合成といった音声情報処理においても、これらに対処するには従来のように標準的な国語（ないし主要な方言）を対象とした研究開発では限界がある。処理方式として、音声言語サンプルデータに依存することが少ない手法の開発が必要となる。

ところが、現在の最有力な音声認識・合成処理のパラダイムは、所望の用途と同じ環境の（音声・言語）サンプルをモデルの構築などに直接大量に利用する方式である。認識に限っていえば、精密な確率統計的音声（言語）モデルをできるだけ同じ環境の大量の音声(言語)サンプルを用いて作成し、このモデルを用いて認識する方式である[1)]。これは、こうした方式を可能にする安定な手法が開発されたことによるもので、その進歩は大きいが、一面では、常にバランスのとれた均質で大量の「生」データを収集整備する必要がある。また、日本語対応のシステムであれば、その中に外国語が混じるという想定はなく、それらは基本的に日本語（に含まれる外来語）として扱う必要があった。

筆者等は、音声を音素やIPAなどの音声記号よりも音響領域に近い中間符号系列を用いて表わし、この符号系より上位の処理は記号領域における距離計算に基いて行う認識方式の開発を進めている[2)-4)]。こうした方式を採用することにより、従来法（音声サンプルから直接音素モデルなどの尤度を計算し、その積分をとる方式）に比べ認識性能の若干の劣化が予想されるが、音声サンプルそのものを大量に収集・蓄積する作業や、実音声データに依存する処理を軽減できると見込まれる。

ここでは、ボトムアップに導出されるセグメントとして、筆者等が提案してきた

区分線形セグメントラティス(Piece-wise Linear Segment Lattice)表現[5]の個々のセグメント（以下、PLSと略記）を採用した場合について報告する。この条件では、個別言語に依存することが少ないボトムアップ中間符号系を採用したことになり、共通中間符号系への試験となる。最初に、基本的な枠組みについて述べ、次に上記方式に基づいて行った単語音声認識実験の具体的手法と結果を示す。

2．処理方式の基本的枠組

図1に、本方式をワードスポッティングとして実現したときの処理の枠組みを示す。ここで、共通中間符号系(Common phonetic Code)は音声サンプル（音響特徴量）から適当な規準に基づきボトムアップに求めた符号系である。一方、トップダウンの中間符号表現は音素記述などからルールにより導出される。ボトムアップ中間符号系とトップダウン中間符号系の間の距離行列は、予め（あまり大量ではない）音声サンプルを利用して作成しておく。認識処理は、仮説単語から導かれる中間符号表現を入力音声から得られるボトムアップ中間符号系列から検出・認識する。この処理方式には時空間パターンの不均一な量子化と標本化が入り、ラベルの尤度をアナログ的に積分する方式などに比べ認識率という点では通常不利になる。しかし、一旦、中間符号列に変換すれば上位からの多様な仮説（例えば多言語など）に対して容易に適応できる利点がある。

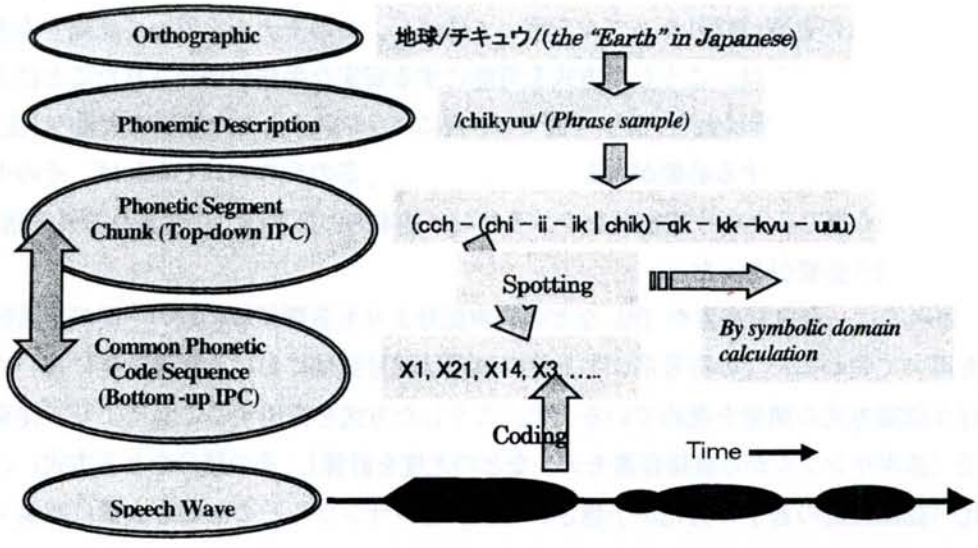

Fig.1 Schematic diagram illustrating the proposed phrase spotting procedure.

ボトムアップ中間符号系として望ましい性質は、できるだけ言語系などに独立であること、音声サンプルデータからその表現モデルを自動学習することができること等が挙げられる。現時点では、前述したようにこの符号系として言語系を仮定する必要の無い PLS と呼ぶセグメント [6]を採用し、トップダウン中間符号系列として音響的に定常なセグメントと過渡的セグメントを分割した音声単位である音素片 [5]を採用している。

3．音声サンプルから中間符号系列の導出

3．1 区分線形セグメント (PLS)

PLS は、音声サンプル（特徴ベクトル時系列）の各セグメントを区間平均値と線形回帰係数とで区分的に表わしたものである。音素バランス単語セットの多数話者音声サンプルを学習データとして、単語の類別が可能になるようなセグメント符号セットを導出する。この時与える情報としては、それぞれの音声サンプル同士が同じ語であるか否かのみで、（音素表記のような）単語の記述は与えない。その導出手順は、ボトムアップ方向のセグメント化とセグメントの分割と統合を一定の数理統計的基準の下で行う処理である。詳細は文献4）を参照されたい。

3．2 セグメントの表現と音声サンプルの符号化

中間符号系同士の距離計算を行うために、PLS と音素片に関しても隠れマルコフモデル(Hidden Markov Model, HMM)によってその音響量ベクトルパターンを表現した。学習用音声サンプルは、3．1節の処理によって PLS セグメント符号にラベル付けされている。したがって、このラベル系列を基に、ボトムアップセグメント (PLS) ラベルの HMM が作成される。HMM を使用される音響量は、概略、ピーク強調パワースペクトルのメルケプストラムとΔメルケプストラム（各12次元）である。

PLS は、音響的にコンパクトであることが想定されるので、LR(Left-to-Right)型、3状態、3ループの連続分布 HMM で、各状態の確率分布は単一ガウス分布である。一方、トップダウン中間符号である音素片も、同じく LR 型、3状態3ループの HMM でモデル化されているが、各状態を2～4個の混合ガウス分布で表わしている。

音声サンプルから PLS 符号列への変換は、次に続く符号の種類を制限する拘束（この拘束条件は学習サンプルセットから求めておく）のみを利用して、HMM に基づく認識として実行される。

4．認識手法

音素片 HMM と PLS-HMM 間の距離は、各状態の確率分布のセントロイドの値 $c_{ij}(k)$ (k:

ラベル、i:状態No.、j:サブ分布 No.)のみを用いて、例えば各セグメントラベル k と l の距離は次式のように定める。

$$D(k,l)=\sum_{i=1}^{3}\ \min_{j,j'}\left[\{c_{ij}(k)-c_{ij'}(l)\}^2\right] \quad (1)$$

認識手法は、この距離行列を利用したワードスポッティングを基本としている。具体的には、入力音声サンプルから得られた PLS 系列と単語仮説から得られる音素片ネットワークとの最適マッチング距離を標準パターン側を基本軸とする連続 DP によって求める。この距離の時間パターンに基いて認識する。今回の実験では、PLS の区間長は音素片に比べ短いことを考慮した DP の許容パスを用いた(**図 2** および式(2)参照)。また、以前の孤立発声単語認識との比較のため音声区間に 1 単語存在するのみという仮定の基に単語の検出を行っている。(なお、単語自体を PLS で表現すれば高認識率は得られるが[4]、語彙に対する汎化が難しく、また個別言語系音韻との関係づけがない。)

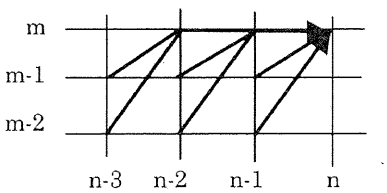

Fig.2 Allowable path type used in the distance calculation

$$G(m,n)=\min\left\{\begin{array}{l} G(m-2,n-1)+2D(m,n) \\ G(m-1,n-1)+D(m,n) \\ G(m-2,n-2)+D(m,n-1)+D(m,n) \\ G(m-1,n-2)+1/2\cdot\{D(m,n-1)+D(m,n)\} \\ G(m-2,n-3)+2/3\cdot\{D(m,n-2)+D(m,n-1)+D(m,n)\} \\ G(m-1,n-3)+1/3\cdot\{D(m,n-2)+D(m,n-1)+D(m,n)\} \end{array}\right\} \quad (2)$$

5．単語音声認識実験

5．1 実験概要

ここでは、提案する処理方式について、a）トップダウン中間符号とボトムアップ中間符号が共に音素片（個別言語系依存）とした場合、b）ボトムアップ中間符号

を PLS とした場合、の比較を行う。また、PLS の符号帳サイズ（種類数）を変えた場合についても実験する。

音素片HMMは、単語音声（ETL-WD-II、1542 語の前半 1050 語、男性話者 6 人）から学習した。また、PLS-HMMも同じデータから学習し、両者の間の距離行列を求めた。認識実験におけるテスト音声サンプルセットは、別の男性話者 4 人、単語は WD-II の後半 492 語を用いて話者・語彙共に学習データに対して独立の認識実験を行った。

5．2 実験結果と考察

PLS の符号帳サイズを 256、512、1024 としたときの単語音声認識率、およびボトムアップセグメントも音素片ラベルとしたときの認識率を**表 1** に示す。ボトムアップ側も音素片の場合が概ねボトムアップラベルの選択に関して認識率の上限と考えられる。この PLS を用いた実験結果でも符号帳サイズを上げるとこの上限にある程度近くはなるが、3%程度の差がある。

Table 1 *Word recognition results for several kinds of bottom-up IPC(Intermediate Phonetic Code) sets.*

Category of the Bottom-up IPC Set		Recognition Rate
Demiphonemes		90.5 %
PLS (Codebook size)	(256)	85.0 %
	(512)	86.3 %
	(1024)	87.8 %

6．むすび

認識率自体を上げるためには、ボトムアップ側の表現を工夫する（たとえば、複数のラベル候補で表現）必要がある。また、音響量選択自体に関しても検討の余地がある。音響特徴としてホルマント周波数とその相対強度を採用した実験も行ったが、現在のところ、パワースペクトル包絡特徴の方が認識率の点で優位にある。今後は日本語以外の音声も含めて検討する計画である。

終わりに、共同研究者である児島宏明氏、実験を手伝って戴く藤村奈保子氏、日頃ご討論を戴く電総研音声研究グループの皆様に感謝します。

参考文献

1）Junqua, J.C., Haton, J.P., *Robustness in Automatic Speech Recognition*, Kluwer

Academic Publishers, 1995.

2) Tanaka, K., Kojima, H., "A between-word distance calculation in a symbol domain and its applications to speech recognition", *Proc. of International Conference on Neural Information Processing (ICONIP-97)*, pp.1107-1111 (1997-11)

3) Tanaka, K., Kojima, H.,"Estimation of the degree of speech recognition difficulty for word sets: An application of between-word distance calculation in a symbolic domain," *J. Acoustical Society Japan(E)*, Vol.19, No.5, pp.341-349 (1998-9).

4) Tanaka, K., Kojima, H., "Speech recognition based on distance calculation between intermediate phonetic code sequences in symbolic domain," *Proceedings of International Conference of Spoken Language Processing (ICSLP'98)*, pp.361-364 (1998-12).

5) Kojima, H., Tanaka, K., "Generalized phone modeling based on piecewise linear segment lattice," *Proceedings of ICSLP'98*, pp.2943-2946 (1998-12).

6) 田中和世、速水悟、太田耕三、"実音声の音素片ラベリングに基づいた音響音声的変動パターンの類型化と自動認識への応用"、電子情報通信学会論文誌 D-II,Vol.J73, No.10, pp.1619-1629 (1990-10).

Suffix Array ツールキット

—— 電子化コーパスを用いた各種統計量の計算 ——

山本幹雄

1.はじめに

自然言語の計算機処理手法を研究する分野（自然言語処理）では、大規模なコーパスから得られる統計量を用いる方法が最近盛んに研究されている。統計量の基本は数を数えることである。しかし、日本語テキストの場合、単語の間にスペースを入れる慣習がないため、単語を数えることすら単純ではない。単語間にスペースを入れる慣習のある英語は単語を数えることはさほど困難ではなく、単語を数えることから始めればよいが、日本語の場合文字を数えるところから始めなければならない（幸い、最近は性能のよい日本語形態素解析システムがフリーソフトウェアとして多数存在するためこの壁は取り払われつつあるが）。また、単語や文字を単純に数えることよりももう少し複雑なものとして、複数の単語あるいは文字の連続(ngram)の出現回数をカウントすることが重要であるが、これも大規模なコーパスに対してはそう簡単なことではない。

本稿では、最近の計算機科学の分野で発達してきたいくつかの手法を用いて、任意長のngramの各種統計量を大量のコーパスから比較的高速に得るためのソフトウェアの開発について述べる。

2. Suffix Arrays

最近では、言語研究等に使用できる電子化された新聞記事が多数存在する。新聞記事はおおよそ１年分で数千万文字であり、メモリーにして約100Mbyteほどである。これを10年分集めたコーパスはテキストデータだけて1Gbyteとなるが、各種統計量をこの1Gbyteの元のデータに対して毎回計算していては最近の計算機の性能をもってしても効率的ではない。しかし、各種統計量に最適な前処理を行うことにより劇的に効率化することができる。その中でも、Suffix Arrayと呼ばれる構造は、任意長のngramの出現回数を計算するのに適切である[Manber&Myers 93]。以下、Suffix Arrayについて簡単に解説する。

コーパス全体を長い文字列と見なしたとき、ある部分からコーパスの最後ま

での文字列をここで言うsuffixと定義する。長さNのコーパスにはN個のsuffixが存在することになる。すべてのsuffixを辞書順にソートしたものがsuffix arrrayである。計算機上では各suffixを最初のunit（コーパスの単位：文字あるいは単語）へのポインタとして表現すれば、suffixの集合全体を長さNのポインタ配列で表現できる。この配列の各要素（suffixへのポインタ）を辞書順にソートすればよいので、必要なメモリはNの定数倍ですむ。この様子を図１に示す。この例ではコーパスの単位を単語としているが、文字でもかまわない。

suffix arrayが一旦できてしまえば、任意の長さの文字列（長さをMとする）の出現頻度、出現位置を、２分探索手法を使ってO(MlogN)で計算することができる。この様子を図２に示す。コーパスを頭から検索する手法ではO(N)の時間が必要であるが、suffix arrayの場合はコーパスの長さに関してO(logN)であり極めて効率がよい（最近のコーパスではN=100Mは当たり前である。100Mとlog100M= 26.6とを比べるといかに効率がよいか分かるであろう）。

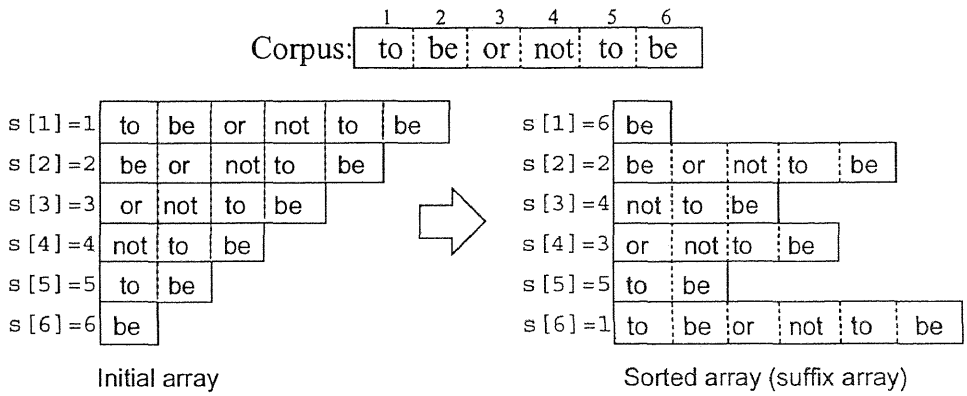

図１　suffix arrayの作成

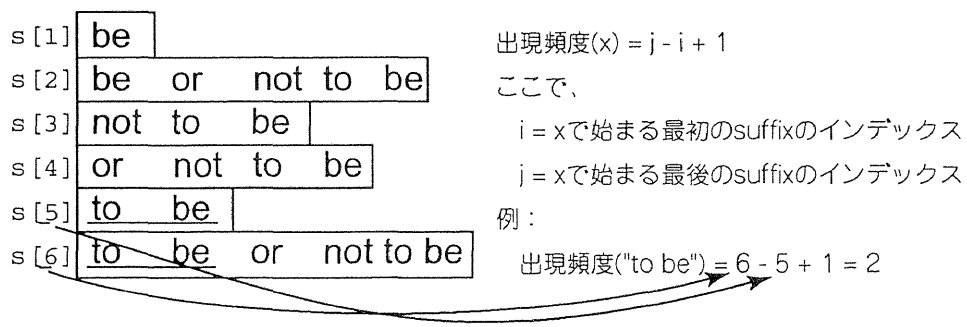

図２　suffix arrayを使った出現頻度の計算

3. Suffix Array ツールキット

前節で述べたsuffix arrayを基本データ構造とした研究用ツールキットをUNIX上に開発した。作成したツールキットはコーパスからsuffix arrayを作成し、基本統計量として任意長ngramの「出現頻度」、「出現箇所」、および「文書頻

makevoc: 1行1単位（単位は単語でも文字でもよい）の入力から語彙を作成する。
makecor: makevocで作成された語彙ファイルを元にコーパスの単位を数値で表現したファイルへ変換する。これによって、英語や日本語といった言語の違いや、単語や文字といった単位の違いをこの段階で吸収する。
qsufsort: suffix arrayを作成する。
allclasses: class arrayを作成する。
jconc: suffix arrayを使ってKWICを出力する。

図3 Suffix Array ツールキットのコマンドの一部

・・・
ら近い五色沼を歩いた。一転して　暖かな春の日差しの中で、観光客
こえてくると、人々が待ちわびた　暖かな春の訪れを告げるかのよう
登場してきた人物や動物の様子を　暖かな色や、幸せな気持ちでえが
登場してきた人物や動物の様子を　暖かな色や、幸せな気持ちでえが
ていたので、主人と動物病院へ。　暖かな診察室に入ったら途端に元
）<EOP>　◇２割が日本人<EOP>　暖かな太陽が降り注ぐ米国フロリ
境川の堤防に出た。大寒にしては　暖かな朝で、散歩する人影がちら
く片付けられていたかの巨人が、　暖かな追憶のまなざしを浴び、窮
（１万円）<EOP>◎「一日も早く　暖かな日が来る事を信じて頑張っ
婦・３３歳＞<EOT>　冬の真昼の　暖かな日だまりの中、いつものよ
に生きて／１　【大阪】<EOT>　暖かな日差しの下、鏡のように穏
、同じゴザに。ビールで乾杯し、　暖かな日差しの下でゲームや踊り
の踊子」で有名な天城峠周辺は、　暖かな日差しを楽しむハイキング
本真弓さん（１６）は「昨日から　暖かな陽気で売り上げがダウン」
残ると景気は冷える？　ちょっと　暖かな立冬<EOT>　立冬の八日朝
ふつくりとした、少しは濕つぽい　暖かみで、身を包むことが（略）
「雪が深いため春にあこがれ、　暖かみのある赤を求めるのでは」
娘でなければ言えない、残酷さと　暖かみの篭もった言葉が、なぜか
ぼり吸うのも、高齢の男女が手の　暖かみを感じ合いながら散歩する
ら送られてきたのですが、室内が　暖か過ぎ、すぐに枯れ始めました
・・・

図4 KWICの例:　「% jconc mai 暖か」の実行結果の一部

度」を効率的に計算する。文書頻度とは、与えられたngramを含む文書の数である。たとえば新聞記事の場合はそのngramを含む記事の数となる。文書頻度に関しては、suffix arrayだけでは高速に計算できないが昨年我々が開発したclass arrayと呼ぶ構造を使って高速に検索できる[Yamamoto&Church 98]。出現頻度からは出現確率、相互情報量等が計算できる。出現箇所からはコンコーダンス(KWIC)が作成できる。文書頻度からは情報検索で単語の重要度としてよく用いられるIDF (Inverse document frequency)[Salton83]が計算できる。これらの統計量を任意のngramについて高速に求めることができる点が本ツールキットの特徴である。

基本コマンドとその簡単な解説を図3に示す。これらのコマンドを組み合わせてsuffix arrayを作成したり利用したりする。Suffix arrayが作成されたとして、jconc（KWIC作成）コマンドの実行例を図4に示す。

4. Suffix Array ツールキットの使用例

本節では、ツールキットの具体的な使用例として、いわゆる形容動詞の名詞的あるいは形容詞的な文法的傾向を経験的に分析する。寺村秀夫[寺村82, pp.62-75]によれば、形容動詞（寺村流に言えば名詞的形容詞）と名詞、形容動詞と形容詞の境界は連続であり、中間的な単語が多数存在する。たとえば、「名誉」は「名誉の勲章」でも「名誉な勲章」でもどちらの形でも使用できる。また、「暖か」は「暖かい風」でも「暖かな風」どちらでもよい。しかし、「特殊」は「特殊の方法」や「特殊い方法」とは言えない。

「茶筅」[松本99]というフリーソフトウェアの形態素解析システムの辞書から抽出した2970個の形容動詞について、その名詞度と形容詞度を測定した。測定方法としては、まず形容動詞に「を」、「な」、「い」の3種類の文字を付属させた3つの文字列に対する出現頻度を新聞記事2年分からツールキットを用いて計算した。それぞれの頻度を「＋を」頻度、「＋な」頻度、「＋い」頻度と呼ぶことにする。たとえば、「名誉」であれば、「名誉を」「名誉な」「名誉い」の出現頻度がそれぞれ、「＋を」頻度、「＋な」頻度、「＋い」頻度となる。それぞれの頻度は、名詞、形容動詞、形容詞として使われている傾向をおおよそ表していると考える。これらの出現頻度を使って少々荒っぽいが名詞度と形容詞度を以下のように定義する。

名詞度　＝　「＋を」頻度／「＋な」頻度
形容詞度　＝　「＋い」頻度／「＋な」頻度

毎日新聞CD-ROMの1994, 1995年版、計約1億文字のデータを用いて、上記で述べた2970個の形容動詞について統計量を計算した。処理時間は3年ほど前のワークステーションで1分程度である（現在のパソコンの方がはるかに処理能力は高い）。結果から20個の例を拾い出したものが以下である。名詞度の高いものを上、形容詞度の高いものを下に配置した。この表は、直感とおおよそ合うと思われるものを拾い出した。「健康」のように「＋い」頻度が91となり直感と合わないものももちろん含まれているが、KWICを使って調べるとそのすべての出現が「健康いま」という新聞のコラムのタイトルであることが分かった。このように、疑わしいものはKWICを使って詳細にチェックすることができる。

形容動詞	+を	+な	+い	名詞度	形容詞度	
傑作	35	2	0	17.500	0.000	
暇	292	49	0	5.959	0.000	
めんどう	17	5	0	3.400	0.000	
名誉	203	72	0	2.819	0.000	
無関心	20	39	0	0.513	0.000	名詞的傾向が強い
意地悪	9	27	2	0.333	0.074	
親切	15	75	0	0.200	0.000	
ちぐはぐ	1	25	0	0.040	0.000	
必要	141	4917	0	0.029	0.000	
軽やか	0	68	0	0.000	0.000	
特殊	0	467	0	0.000	0.000	
大げさ	0	33	0	0.000	0.000	真(?)の形容動詞
猛烈	0	190	0	0.000	0.000	
割高	1	59	10	0.017	0.169	
骨太	0	16	3	0.000	0.188	
手荒	0	8	5	0.000	0.625	形容詞的傾向が強い
柔らか	0	93	190	0.000	2.043	
間近	2	5	36	0.400	7.200	
暖か	0	22	221	0.000	10.045	
幅広	0	0	995	0.000	995.000	

この表から、「傑作」や「暇」はかなり名詞的な傾向が強い形容動詞であり、「暖か」や「柔らか」は形容詞的な傾向が強いものであることが確認される。また、「軽やか」や「特殊」は名詞的・形容詞的な傾向をほとんど持た

ず、真(?)の形容動詞であると言うことができる。

5．おわりに

本稿では、ある程度大規模なコーパスに対しても任意のngramの出現頻度、出現箇所、文書頻度を高速に計算できる手法、ツールキットの開発とその応用例について述べた。現在、ツールキットはUNIXワークステーション上でしか動作していないが、一般的なパソコン上で動作するように移植する予定である。

参考文献

[寺村82] 寺村秀夫: 日本語のシンタクスと意味I, くろしお出版, 1982.

[松本99] 松本祐治, 北内啓, 山下達雄, 平野善隆: 日本語形態素解析システム「茶筅」 version 2.0 使用説明書, Information Science Technical Report NAIST-IS-TR98008, Nara Institute of Science and Technology, 1999.

[Salton83] G. Salton and M.J. McGill, The SMART and SIRE Experimental Retrieval Systems, pp.118-155, New York: McGraw-Hill, 1983.

[Manber&Myers 93] Udi Manber and E. Myers: Suffix array: a new method for on-line string searches, SIAM Journal on Computing, 22:5, pp.935-948, 1993. http://glimpse.cs.arizona.edu/udi.html.

[Yamamoto&Church 98] M. Yamamoto and K.Church: Using suffix arrays to compute term frequency and docuement frequency for all substrings in a corpus, In proceedings of 6th Workshop on Very Large Corpora, Ed. Eugene Charniak, Montreal, pp.28-37, 1998.

文書構造変換規則と記述内容処理関数を用いた構造化文書操作系 *

北川 博之 †　品川 徳秀 ‡　石川 佳治 †

† 筑波大学 電子・情報工学系　‡ 筑波大学 工学研究科

1 はじめに

近年、各種文書データベースの構築やその利用の高度化が急速に進みつつある。特に、インターネットの爆発的な普及に伴い、XML をはじめとする構造化文書データベースの構築とその高度利用の要求が増大している [2]。構造化文書データベースに対する処理記述においては、タグの階層構造で与えられる文書構造に対する処理と、文字列テキストで与えられる記述内容に対する処理の両者を対象とする必要がある [3]。構造化文書データベースに対する基本的な処理記述言語として、これまでに各種の問合せ言語が提案されている [4]-[14]。

XML を対象とした問合せ言語の代表例としては、XML-QL [4] や XQL [5] 等がある。また、半構造データを対象とした問合せ言語である Lorel [8]、UnQL [9]、StruQL [10]、YATL [11] 等もその適用対象の一つとして XML 構造化文書を想定している。これらの問合せ言語のいずれにおいても、何らかの意味で文書構造処理と記述内容処理の両者に関する機能が提供されている。特に、XML-QL や半構造データ問合せ言語は、文書構造に対する比較的強力な操作体系を有する。しかし、これに対比して記述内容処理に関しては、指定した語句やパターンに合致する部分文字列を有する文書要素を選択するといった、極めて限定された機能しか提供していない。

今後、電子商取引をはじめとして、構造化文書の利用がますます活発になると予想される。様々な利用者の要求に応えるためには、構造化文書のより高度な利用が必要である。その一つの方向として、記述内容処理機能の一層の拡充は重要な課題である。特に、大量の構造化文書から目的とする情報を効率的に獲得するためには、与えられたキーワード群との意味的な類似度に基づく文書要素の絞り込みやランキング、文書要素の要約や話題抽出等の機能を上記のような文書構造操作の枠組みに統合することが極めて重要であると考えられる。例えば、XML 構造

* 本稿の主要部分は著者らの学会発表論文 [1] に基づく。

化文書群として与えられた新聞記事データベースに対する問合せを考えてみた場合、該当する記事の掲載日や見出しを抽出して適当な一覧に再構成するといった文書構造操作に加えて、記事内容と問合せの類似度を測定し、必要に応じて要約を添付する等の記述内容処理を伴うような要求が処理できる事が求められる。ここで、類似度の計算方法や要約の作成方法は、対象文書や文書要素の種類に依存するということにも注意する必要がある。例えば、見出しのような特定の文書要素に出現する語には、他の文書要素に含まれる語句に比べ、より大きな重み付けを行なう等がしばしば行なわれる [15]。また、自動要約生成、話題抽出等に対しては各種の方式が提案されている [15]-[17]。それゆえ、類似度の導出や要約の作成等の具体的な記述内容処理の詳細は、利用者がその目的に合致するものを必要に応じて外部から与えられる拡張性を有することが望ましい。

本研究では、このような要求に対応するため、構造化文書に対する問合せ言語が持つ文書構造操作機能と、類似検索や要約生成等のより高次の記述内容処理機能を統合した、構造化文書操作系を提案する。本操作系は、XML 構造化文書を操作対象とし、XML-QL に準じた様式で与えられる文書構造変換規則群と、記述内容処理関数を組み合わせることで、上記の統合を実現する。この記述内容処理関数は利用者がその目的に応じて追加定義できるものとする。また、本操作系で与えられる XML 構造化文書操作を、ある種の制約の下で XSLT [6] に基づく文書操作へと変換するための手順を示す。従って、ここでの制約を満たす入力の範囲内では、本操作系を XSLT 処理系上に実装することが可能である。

以下では、まず 2 節で文書構造操作と高次の記述内容処理の統合を必要とする構造化文書操作例を示す。次に、3 節で XML-QL について概説し、4 節で本操作系について詳細を述べる。また、5 節で本操作系から XSLT による記述への変換方式を述べる。6 節で関連研究について言及し、7 節でまとめと今後の課題を示す。

2 構造化文書操作例

本節では、記述内容処理と文書構造操作の両者の能力を必要とする構造化文書操作の例を示す。4 節では、本節の例 1 を利用して本操作系を説明する。以下では、必要に応じて ⟨識別子⟩ で "識別子" という文書要素名もしくはその文書要素を表記する。また、属性名も同様に表記する。

例 1　ある新聞の記事の記事データベースとして構成された XML 文書を考える。各記事には、掲載日、見出し、本文が含まれているものとし、文書の DTD は次で与えられるものとする。

```
<!-- 入力文書の DTD -->
<!ELEMENT 文書    記事+>
<!ELEMENT 記事   (掲載日, 見出し, 本文)>
<!ELEMENT 本文    段落+>
<!ELEMENT 掲載日 #PCDATA>
<!ELEMENT 見出し #PCDATA>
<!ELEMENT 段落    #PCDATA>
```

読者はある事件に興味を持っており、この文書から興味に合致する記事を掲載日別に n 件ずつ抽出し、得られた各記事の見出しと内容の要約を得たいと仮定する。これは、例えば次のような操作で実現される。

1. ⟨文書⟩から各⟨記事⟩を取り出し、それらを⟨掲載日⟩によってグルーピングを行なう。
2. グループ毎に、読者が与えた興味内容を表わすキーワード群に対する各⟨記事⟩の類似度を調べ、ランキング結果の上位 n 件を選び出す。
3. 選び出された各⟨記事⟩について、⟨見出し⟩を抽出し、更に⟨本文⟩からその要約となる文書要素を生成する。

この例において、1 は文書構造操作の典型例の一つである。また、2 は⟨掲載日⟩別に分類された⟨記事⟩集合の各々に対して類似度に基づく文書要素の選択を行なっている。3 では、結果の各⟨記事⟩について文書構造の再構成と⟨本文⟩の自動要約生成を行なっている。要約生成には様々な方法があるが、ここでは⟨見出し⟩との類似度が高い段落や文を一定量抽出するという方法を考える [16]。これは抽出単位となる段落や文等の文書要素が導入されていれば、抽出単位となる段落や文等の文書要素が導入されていれば、2 と同様な処理であるとみなせる。具体的な抽出単位として、⟨段落⟩を用いる。本操作の結果として得られる文書は次に示す DTD に従ったものとする。

```
<!-- 出力文書の DTD -->
<!ELEMENT 文書    日別+>
<!ELEMENT 日別    記事+>
<!ELEMENT 記事   (見出し, 本文)>
<!ELEMENT 本文    段落+>
<!ELEMENT 見出し #PCDATA>
<!ELEMENT 段落    #PCDATA>
```

例 2　ある分野の製品シェアに関する調査結果表と、それらの製品の比較・評価に関する記事を含む構造化文書がある。記事は段落分けされており、この文書は次の DTD に従うものとする。

```
<!-- 入力文書の DTD -->
<!ELEMENT 文書       (シェア表, 評価記事)+>
<!ELEMENT シェア表   製品情報+>
<!ELEMENT 製品情報 (製品名, シェア)>
<!ELEMENT 評価記事   段落+>
<!ELEMENT 製品名     #PCDATA>
<!ELEMENT シェア     #PCDATA>
<!ELEMENT 段落       #PCDATA>
```

この文書に対して、評価記事の中でシェア 30% 以上の製品について触れている「部分」を抜き出したいという状況を考える。ここで「部分」とは、文書要素として明示的に与えられてはいない、複数の段落から構成される「話題単位」を示すものとする。テキストタイリング [17] で行なわれているように、話題単位の境界は、連続する段落間の類似度の推移グラフがある深さ以上の谷となる段落の境界によって与えるものとする。この要求は、例えば次のような処理で実現され、その結果として与えられる文書は下に示す DTD に従ったものとする。

1. ⟨評価記事⟩について、⟨段落⟩間の類似度の推移条件に従って話題単位を構成する⟨段落⟩列を抽出し、新たな文書要素である⟨話題単位⟩としてまとめる。

2. ⟨シェア表⟩から⟨シェア⟩が 30% 以上の製品の⟨製品名⟩を抽出し、1 で得られた⟨話題単位⟩の列から、⟨製品名⟩を文字列として含むものを選択する。

```
<!-- 出力文書の DTD -->
<!ELEMENT 文書       製品+>
<!ELEMENT 製品       (製品名, 記事)>
<!ELEMENT 記事       話題単位+>
<!ELEMENT 話題単位   段落+>
<!ELEMENT 製品名     #PCDATA>
<!ELEMENT 段落       #PCDATA>
```

ここで、例 1 では、1 のグルーピングによって得られる⟨日別⟩に対して 2 の処理を行ない、また、例 2 では 1 による再構成で生成される⟨話題単位⟩に対して 2 の処理を行なっている。このように、同じ部分について複数の処理を重ねて行なう必要がある。このような、処理結果を中間結果とみなし、更に処理を重ねることを多段階の処理という。

3 XML-QL

本稿で提案する操作系は、基本構文として XML-QL [4] をベースとしている。これは、従来の問合せ言語との親和性を高め、可読性の高い記述とするためであ

る。本節では、4 節での操作記述方式を理解する上で必要な XML-QL の機能について説明する。

3.1 問合せ

XMl-QL は XML 文書に対する問合せ言語であり、基本構文は次の通りである。

```
where      パターン式 (及び変数束縛)
           [in 対象文書もしくは文書要素]
[order-by 順序評価基準式 [descending]]
construct 出力生成式
```

例えば、前節の例 1 の 1 は次のように記述される。出力は ⟨ 日別 ⟩ の列を内容とする ⟨ 文書 ⟩ である。

```
where     <文書></> content_as $x
construct <文書>
  where     <記事>
              <掲載日> $d </>
            </> element_as $a in $x
  order-by  $d
  construct <日別 ID=DateID($d)>
              $a
            </>
          </>
```

where 節　問合せ条件の指定を行ない、対象文書ないし文書要素中のパターン式に適合する文書要素に対して、必要ならば変数を束縛する。パターン式は、ルートを起点とする文書要素の並びであり、変数の束縛は適合する全ての組合せに対して行なわれる。尚、文書要素の属性も同様に対象とする事ができるが、本稿では説明を省略する。

パターン式 `<a><b></> element_as $x </>` は、⟨a⟩ の子要素である ⟨b⟩ を変数 `$x` に割り当てる。また、パターン式 `<a><b></> content_as $x </>` は、`<a><b> $x </></>` と等価であり、⟨a⟩ の子要素である ⟨b⟩ の内容全体を `$x` に割り当てる。パターン式 `<a><$x></>[$i]</>` は ⟨a⟩ の任意の子要素の文書要素名をタグ変数 `$x` に割り当て、`$i` にその出現位置を示す添字番号 (0,1,...) を割り当てる。

構造指定にはパス正規表現のサブセットを利用できる。`<a|b>` は ⟨a⟩ もしくは ⟨b⟩ に、`<a.b>` は ⟨a⟩ の子要素である ⟨b⟩ に適合する。*, + はそれぞれ、直前のパス名の 0 回以上, 1 回以上の繰返しを示し、$ は任意の文書要素に適合する。

尚、パターン式中での文書要素の出現順序は、対象文書要素中での出現順序を規定しない。

冒頭の記述例では、一つ目の where 節で〈文書〉の内部を $x に束縛し、二つ目の where 節では $x 中の各〈記事〉を $a に、その〈掲載日〉を $d に束縛する。

construct 節　where 節のパターン式に適合する変数の各組合せに対する出力の構成を指定する節であり、変数を含む文書要素もしくは文字列の形式で記述される。問合せ結果は、この記述の変数を展開する事で生成される文書要素の列となる。

また、construct 節中に where～construct 節を入れ子にする事ができる。この場合、外側の where 節で適合する変数の値を固定した上で、内側の where～construct 節が処理される。

生成する文書要素には、その ⟨ID⟩ 属性をスコーレム関数で与えることができ、同じ ⟨ID⟩ を持つ文書要素は一つの文書要素実体に集約される。スコーレム関数はその引数と返値が一対一対応する。これを利用することによりグルーピングを実現できる。

冒頭の例では、副問合せの処理結果を内容とする、ただ一つの〈文書〉が生成される。また、副問合せでは〈記事〉($a) を内容とする〈日別〉の列が生成される。それぞれの〈日別〉は、construct 節中でスコーレム関数 DateID() によって〈掲載日〉の値 $d から生成された ⟨ID⟩ が与えられる。このため、同じ〈掲載日〉から生成された〈日別〉は一つにまとめられる。結果として、〈掲載日〉によって〈記事〉が〈日別〉にグルーピングされる。

order-by 節　文書要素の出力順を指定する節で、出力は指定されたキーの値について昇順に、descending が指定された場合には降順にソートされる。

冒頭の例では、出力の〈日別〉文書要素を変数 $d の値によって日付順にソートして出力を行なう。

3.2　関数定義

次の構文を用いて、ユーザは必要に応じて関数を定義することができる。

```
function 関数名 ( 引数リスト )
where～construct 問合せ
end
```

関数定義中の問合せ結果が関数呼出しの返値として扱われる。このため、XML-QL で定義可能な関数は問合せとして表現できるものに限られる。

4 提案記述方式

4.1 文書構造変換規則

本操作系では、構造化文書に対する記述内容処理関数を含んだ操作を文書構造変換規則に基づいて記述する。文書構造変換規則 (以下、変換規則) はパターンに適合する部分構造に対する変換方法を表す。ルール指向の記述を用いることは、本来、文書構造操作機能とより高次の記述内容処理機能を統合することとは独立な事項ではあるが、次のような利点がある。

1. 様々な文書に共通に現れる部分構造に対する変換操作をより簡潔に記述することが容易になる。例えば、種々の文書構造の中に埋めこまれているある種の文書要素を別の文書要素で置き換えるような処理を実現するには、XML-QL では上位の文書構造に応じて異なる問合せ記述とし、更にその上位の構造を保存するよう記述する必要がある。これは記述に繁雑さをもたらす。また、あらかじめ上位の文書構造がある程度限定できない場合には、記述自体が困難な場合もある。
2. XSLT をはじめとするルール指向の処理方式が今後広く普及する可能性があるが、それらとの整合性が高い。例えば、YATL や XQL 等、ルール指向の枠組をベースとした構造化文書操作系も既に幾つか存在する。また、本論文では 5 節において XSLT への変換方法を示す。
3. 例えば、XML-QL 等での通常の問合せ記述は、一定の条件を満たす単一の変換規則からなる変換規則と捉えることが可能であり、この意味で従来の問合せ記述の自然な拡張になっている。

個々の変換規則の記述は、可読性と既存の問合せ言語との親和性を考慮し、XML-QL をベースとする。変換規則の基本構文は次の通りである。

```
rule
   where          パターン式 (及び変数束縛)
                  [in 対象文書もしくは文書要素]
    [ [order-by 順序評価基準式   [descending]]
      [first    出力数計算式     [from-bottom]]
    | [rank-by  順序評価基準式
      top       出力数計算式] ]
    construct    出力生成式
```

文書操作においては、上位 n 件のみといったように、出力として生成される文書要素列の長さをランキング結果に従って限定するという場面が現れる。これに

対応するため、本操作記述では first 節及び rank-by～top 節を導入する。この指定により、construct 節で生成される出力が出力数計算式の評価結果の個数に限定される。

first 節は、出力の生成順で先頭から、あるいは from-bottom の指定がある場合には最後尾から、指定数の文書要素を出力する。この時、order-by 節があればソート結果に対して作用する。

rank-by～top 節は、順序評価基準式の評価結果を基準にしてランキングを行ない、順位が出力数計算式の評価結果以内のもののみを生成順に出力する。これは、order-by 節と first 節を用いた指定と似た作用を持つが、出力の物理的な位置に影響を与えない。また、物理的なソートを行なわないため、descending 及び from-bottom 修飾句を持たない。

4.2 記述例と適用手順

2 節の例 1 で述べた文書操作に対応する変換規則を図 1 に示す。三つの変換規則は、例 1 における操作 1～3 にそれぞれ対応する。

XML-QL ではパターン照合の起点は常にルート文書要素であるが、本操作記述では異なる解釈を行なう。前節の項目 1 に対応するため、図 2 の適用手順に示すようにパターン照合の対象はルート文書要素に限らず、より限定的な部分文書とする。具体的な変換規則の適用手順を以下に示す。

0. 初期状態としてルートの文書要素を照合対象として、変換規則を次のように多段階に適用する。

1. 照合対象の文書要素と変換規則とのパターン照合が行なわれる。変換規則群中に where 節が適合する変換規則が見つかった時は、それを適用し、更に他の照合対象候補が存在する場合には次の照合対象を処理する。出力文書中で照合対象に合致した部分は、その construct 節の出力結果によって置き換えられる。複数の変換規則が適合する場合には、先に定義したものを優先する。

一方、適合する変換規則がなく、現在の照合対象が子要素を持つ時は、それらを出力順に照合対象として順次、同様の処理を行なう。この繰り返しは、下方への再帰が優先され、その処理が終わった後に後続の兄弟文書要素が照合対象とされる。以上が文書の末尾に到達するまで行なわれる。

例 1 においては、⟨文書⟩が最初の照合対象となる。これは、変換規則 1 によって処理されるため、子要素への再帰は発生しない。また、副問合せで適合した全

```
// 変換規則 1 : 例 1.1 におけるグルーピングに相当
rule
  where     <文書></> content_as $x
  construct <文書>
    where     <記事>
                <掲載日> $d </>
              </> element_as $a in $x
    order-by  $d
    construct #<日別 ID=DateID($d)> $a </>
            </>

// 変換規則 2 : 例 1.2 における絞り込みに相当
rule
  where     <日別></>   content_as $x
  construct <日別>
    where     <記事></> element_as $a in $x
    order-by  $a.sim_cosine(keywords)
    first     5
    construct #$a
            </>

// 変換規則 3 : 例 1.3 における要約生成に相当
rule
  where     <記事>
              <見出し></> element_as $h
              <本文></>   element_as $b
  construct <記事>
              $h
              <本文>
    where       <段落></> element_as $p in $b
    rank-by     sim_cosine($h,$b)
    top         3
    construct   $p
              </>
            </>
```

図 1: 変換規則

```
def main():
  # 初期条件にルートを用いて
  target = [ root ]
  # 照合対象がなくなるまで変換規則を適用
  while( target <> [] ):
    target = transform( target )
  exit

# 1 パス分の変換規則の適用
def transform( target ):
  next = []              # 次パスの照合対象リスト
  for n in target:       # 全ての照合対象について
    rule = match( n )    # パターン照合して
    if( rule <> null ): # 適合したら適用して次へ
      next.append( rule.apply( n )
    else:                # あるいは子要素へ再帰
      next.append( transform( n.children() ) )
  return next
```

図 2: 変換規則の適用手順

ての⟨記事⟩から、3 節の例と同様に掲載日別にグルーピングされた⟨日別⟩の列が生成され、これらを子とする⟨文書⟩によって元の⟨文書⟩が置き換えられる。元の文書の構造により、⟨文書⟩は一つだけ生成される。この処理の終了時に得られる文書は、2 節で示した出力文書の DTD に従っている。以上で、例 1 の 1 の操作が完了する。

2. 続いて例 1 の 2 以降の操作を行なうが、上記の手順だけではその処理に移行することができない。このような多段階の処理を実現するため、照合対象指定子 "#" を導入する。文書全体のパターン照合が終わった時点でその出力文書が照合対象指定子を含む場合には、更に変更処理が続けられる。これについては、construct 節中で照合対象指定された文書要素が順次、照合対象として扱われ、同様に再帰的に処理が行なわれる。照合対象指定されたものの評価順序は文書中での出現順序に従う。

例の場合は、⟨日別⟩が照合対象指定されているため、これによって構成された全ての記事列が 2 パス目の照合対象の候補となる。これは例 1 の 2 に相当する変換規則 2 に適合する。尚、*keywords* にはユーザが指定したキーワード群が文字列として埋めこまれているものとする。sim_cosine() は記述内容処理関数である。これは 4.3 節で述べるように $a 即ち⟨記事⟩のメソッドであり、引数のキーワード群との類似度を返す。その際、⟨見出し⟩に含まれる語の重み付けを大きくする、

といった⟨記事⟩固有の類似度の測り方を行なう。これを利用して、各⟨記事⟩とキーワード群との類似度を求め、それに基づいたランキング結果の上位 n を出力とする。また、出力中の各⟨記事⟩が照合対象指定されているため、文書の末尾まで処理が行なわれた後に出力文書中での照合対象として扱われる。

3. 更に、3 パス目として、⟨記事⟩に対して 2 と同様にパターン照合が行なわれ、変換規則 3 の処理が適用される。これにより、例 1 の 3 の処理が適用される。ここでは、記事内の全ての⟨段落⟩に対して⟨見出し⟩との類似度を⟨段落⟩の類似度として与え、そのランキング結果に従って上位 3 件の⟨段落⟩のみが要約結果として残される。ここでも、記述内容処理関数である sim_cosine() を用いている。4.3 節で述べるように、これは類似度を測定する汎用の関数であり、ある文書要素固有のメソッドではない。そのため、変換規則 2 のそれとは異なり、特殊な類似度の測り方は行なわれない。尚、変換規則では照合対象指定が行なわれていないので、この処理が全て終了した時点で処理が完了する。

4.3 記述内容処理関数

XML-QL で定義可能な関数は問合せとして記述できるもののみであることを述べた。このため、柔軟な処理を行なう関数を記述できない。また、組み込み関数の拡充によって記述内容処理が必要とする機能を網羅的に提供することも不可能である。そこで、本操作系においては、Java 等の言語を用いて外部プログラムとして記述内容処理関数を定義可能とする。

一般に、対象文書の従う DTD は変換規則の想定する範囲内で揺れを持っている。例えば、同じ名前の文書要素であっても、DTD 毎に内容の記述フォーマットや特徴量の抽出方法が異なっていたり、定められるタグ名等がまちまちであったりという事は容易に想定できる。このような多様性を吸収するため、記述内容処理関数群の定義は DTD 毎に与える (図 3)。これにより、記述内容処理の実装方法が抽象化され、同一の方法で利用可能になる。

また、図 1 における sim_cosine() で例示したように、同じ DTD 内であっても、文書要素の種類に応じてその記述内容に対する処理が異なっていることが想定される。即ち、関数名が同じであっても文書要素毎の多態性を持つことがある。この他、最初の人名が主著者であるような⟨著者リスト⟩と本文中の⟨段落⟩とでは「重要なフレーズ」は抽出法が異なる等も考えられる。この点から、ある文書要素固有のメソッドとして “文書要素名. メソッド名 ()” という形式で関数を定義し、“文書要素変数. メソッド名 ()” という形式で利用できるものとする。

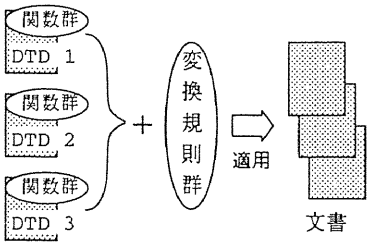

図 3: ユーザ定義関数と DTD の関係

記述内容処理関数の構文は次の通りである。

```
function  データ型 関数名 ( 引数リスト )
define-by "関数実装を含む URI"

function  データ型 文書要素名. メソッド名 ( 引数リスト )
define-by "関数実装を含む URI"

引数リスト ::= データ型 引数名 [, データ型 引数名 ]*
```

記述内容処理関数の定義ではデータ型を明示する。関数の引数及び返値として有効なデータ型は、number (数値)、string (文字列)、element (文書要素)、contents (要素内容) である。ここで、contents 型は、文書要素の内容として許される文字列に相当し、テキスト (#PCDATA) と文書要素からなる列が許される。他のデータ型は、要素数が 1 の列とみなせ、その意味で contents 型は常に他の型の値を受け取ることができる。

例えば、sim_cosine() は、それぞれ次のように定義が与えられる。前者の実装は cosine、後者の実装は sim えあることを示している。

```
function  number sim_cosine( contents x, contents y )
define-by "http://fqdn/path/common/vecspace#cosine"

function  number 記事.sim_cosine( ocntents x )
define-by "http://fqdn/path/dtd1/article#sim"
```

5 XSLT 記述への変換

本節では、本操作系での記述から XSLT 記述への変換について述べる。これによって、既存の XSLT 変換系を利用した本操作系のプロトタイプシステムの構築が可能になる。

5.1 XSLT

XSLT は W3C で策定中の XSLT (XML スタイルシート言語) で用いられる変換言語であり、本操作操作系と同様、ルール指向である。XSLT 記述のことを XSLT

スタイルシートと呼ぶ。XSLT スタイルシートは XML 文書として記述され、主に次に示すテンプレートと拡張関数定義で構成される。

以下では、次の変換規則を例として説明を行なう。これは、⟨文書⟩中の⟨項目⟩をその⟨番号⟩順にソートする。

```
rule
  where     <文書></> content_as $d
  construct <文書>
    where     <項目>
                <番号> $n </>
              </> element_as $i in $d
    order-by  $n
    construct $i
          </>
```

上記の本操作記述に対応するテンプレートは次である。

```
<xsl:template match="文書">
  <!-- ⟨文書⟩が起点となる -->
  <xsl:variable name="d" expr"."/>
  <文書>
    <xsl:for-each select="項目 [番号]">
      <!-- ⟨項目⟩が起点となる -->
      <xsl:variable name="i" expr="."/>
      <xsl:variable name="n" expr="番号"/>
      <xsl:sort     select="$n"/>
      <xsl:copy-of  select="."/>
    </xsl:for-each>
  </文書>
</xsl:template>
```

5.1.1 テンプレート

XSLT スタイルシートは複数のテンプレートによって構成される。これは本操作操作系での変換規則に相当する。

⟨xsl:template⟩ テンプレートを定義するもので、次の基本構造を持つ。

```
<xsl:template match="...">
  <-- 処理 -->
</xsl:template>
```

⟨xsl:template⟩の⟨match⟩属性が変換規則の最初の where 節の条件記述に相当し、適用対象の選択を行なう。<!-- 処理 --> には、他の節に相当する記述が XSLT で定められる文書要素、例えば以下で説明する ⟨xsl:variable⟩ や ⟨xsl:for-each⟩ 等

によって行なわれる。これらは XSLT 処理系によって解釈される。それら以外の文書要素は変数等の展開がされ、その結果がリテラルとして出力される。⟨match⟩ で選択された各ノードはカレントとして扱われる。テンプレート内部はこのカレントに対して適用され、その結果でカレントを置き換える。ここで、ノードとは文書中の文書要素、属性、テキスト等に対応し、その親子関係、所属関係等がツリーの親子関係で表されている。

⟨match⟩ にはロケーションパスと呼ばれるパス式を記述する [7]。ロケーションパスは、以下で説明する [] を除いた末端のノードを選択する。4 節の例を変換するために必要となる主な記法として次がある。(1) "." はそのノード自身を、(2) ".." は親のノードを選択する。また、(3) a/b は ⟨a⟩ の子である ⟨b⟩ を、(4) a/@b は ⟨a⟩ の属性である ⟨b⟩ を、(5) a[x] は条件 x を満たす ⟨a⟩ を、(6) a/node() は ⟨a⟩ の全ての子ノードを選択する。条件 x にはロケーションパスも記述でき、複数の条件を and, or で組み合わせられる。

⟨xs:variable⟩ 変数の束縛をする。⟨name⟩ 属性で変数名を指定し、⟨expr⟩ 属性もしくは記述内容によって値を指定する。これは、属性を記述する値として利用でき、変数名の前に "$" を付けることで参照される。

⟨xsl:for-each⟩ 入れ子問合せは、これを用いて実現される。これは ⟨select⟩ 属性で指定されたロケーションパスで選択された各ノードをカレントとして、その記述内容を適用する。入れ子になった where 節に対応する。

⟨xsl:sort⟩ これは ⟨xsl:for-each⟩ の子要素として与えることができ、その出力を ⟨select⟩ 属性で指定されたソートキーでソートする。また、これを複数指定することで、辞書順のソートを行なえる。order-by 節に直接対応し、rank-by 節もこれを用いて実現される。

⟨xsl:copy-of⟩ ⟨select⟩ 属性のロケーションパスで選択されたノードを複製する。

冒頭のテンプレートの例では、入力の ⟨ 文書 ⟩ について、出力として ⟨ 文書 ⟩ を作り、その内部を ⟨xsl:for-each⟩ によって生成する。この内部は、⟨ 番号 ⟩ を子に持つ ⟨ 項目 ⟩ 全てを ⟨xsl:copy-of⟩ によって複製している。また、結果は変数 $n 即ち ⟨ 番号 ⟩ によってソートされる。結果として、変換規則と同等の処理が記述されている。

また、変数 $d と $i は実際には使用されず、$n は直接展開することができるため、上記のテンプレートは次のように簡略化可能である。

```
<xsl:template match="文書">
  <文書>
    <xsl:for-each   select="項目[番号]">
      <xsl:sort     select="番号"/>
      <xsl:copy-of  select="."/>
    </xsl:for-each>
  </文書>
</xsl:template>
```

⟨xsl:if⟩ XSLT では条件分岐のために、⟨xsl:if⟩ と ⟨xsl:choose⟩ が利用できる。特に ⟨xsl:if⟩ は、⟨test⟩ 属性に記述された条件式が真である時に限り、その記述内容が適用される。

5.1.2 拡張関数

これはユーザ関数の定義に相当する。

⟨xsl:functions⟩ これは関数の宣言を行なうものであり、他の要素の子とはならない。⟨ns⟩ 属性でその関数が属する名前空間名を指定する。⟨code⟩ 属性がある時、⟨ns⟩ をそのプレフィックスとした URI で指定される実装を用い、与えられてなければ ⟨xsl:functions⟩ の記述内容で実装を与える。尚、XSLT では 4.3 節で示したような実装と異なる名前で関数名を利用することはできないため、実装名は一意でなければならない。

例: `func()` として利用するプログラムの実装が http://fqdn/classpath/classname に含まれている時、次のように記述される。ここで、名前空間 `global` は "http://fqdn/classpath" として定義されているものとする。

```
<xsl:funcions ns="global"
            code="/classpath/classname"/>
```

5.2 XSLT への変換方法

5.2.1 処理機構

変換規則からスタイルシートへ変換するに当たって、まず XSLT 処理系をどのように利用するかを明確にする。

提案操作系においては照合対象指定によって多段階の処理を実現しているが、XSLT では直接これをサポートしない。このため、多段階の処理を行なうためには、スタイルシートを複数回適用する必要がある。即ち、XSLT 処理系を上位のモジュールから複数回呼出さねばならない。この適用回数は文書依存であり、あ

らかじめ決定できないため、上位のモジュールはスタイルシートの適用が不要となったことを文書操作時に判断する必要がある。そこで、スタイルシートを更に適用する必要のあるテンプレートが適用された時、そのことを拡張関数 NOTIFY() を用いて上位モジュールに通知する。これによって、上位モジュールはスタイルシートの適用終了時までに通知がなければ、全ての処理が終わったものとして終了することが可能になる。

5.2.2 テンプレートへの変換

次に、個々の変換規則をテンプレートへ変換する際の主な点について説明する。ここでは、以下の条件を満たす変換規則のみを対象とする。

- XML-QL に準じた正規パス表現のサブセットを用いた記述はないものとする。これは、XSLT ではこれらを直接扱えないことによる。
- 副ブロック及び、スコーレム関数によって ⟨ID⟩ が指定された文書要素の入れ子はないものとする。

1. 基本変換　変換規則における where, order-by, construct 節の基本構造は、5.1 で述べたような文書要素に変換される。ここで、XSLT では直前の ⟨xsl:template⟩ の ⟨match⟩ 属性や ⟨xsl:for-each⟩ の ⟨select⟩ 属性等のロケーションパスの末端のノードが起点となるが、提案操作系においては where 節で指定する文書構造は照合対象ノードもしくは、in による指定を起点とする。構造に関する条件を変換する際に、この点を考慮する必要がある。

2. 照合対象指定子　これが指定された文書要素は必ずパターン照合が行なわれる。このため、前述の NOTIFY() を呼出す必要がある。

3. first 節　これに相当する処理は、NOTIFY() を用いて、2 段階の処理として実現する。最初の実行パスで ⟨xsl:sort⟩ による結果の列を生成し、次のパスでその先頭の指定件数だけを出力する。

4. rank-by〜top 節　これも first 節と同様だが、最終的な出力の順序を保存するため、次のように位置を保存した上で 2 段階の処理を行なう。最初のパスで ⟨xsl:sort⟩ により、rank-by 節の指定された順の結果を生成する。その際、各要素には元の文書中での位置を付加しておく。次のパスで、先頭の指定件数分を対象

として抽出する。更に、付加しておいた元の位置情報に従って再度 ⟨xsl:sort⟩ を用いて最終的な結果を生成する。

5. スコーレム関数　スコーレム関数の返値は、引数の値と一対一の対応を持つ。即ち、これによって与えられる ⟨ID⟩ の一意性は、引数の値の組の一意性と等しい。そこで、これによって実現されるグルーピングは次の手順へ変換を行なう。まず、スコーレム関数による ⟨ID⟩ が同じ文書要素のうち、最初の一つのみを選ぶ。これは、⟨xsl:if⟩ を用いた値の組の一致性判定で実現される。そして、その内部でグルーピング結果に相当する結果を抽出するような記述を行なう。これは、⟨xsl:for-each⟩ を用いて、同じ ⟨ID⟩ を割り当てられるもの全てについて必要な構造を生成すれば良い。以上によって、スコーレム関数の値毎に一つずつ、グルーピング結果が得られる。

5.2.3　スタイルシートの生成

上記のテンプレート生成方法を用いて、実行パスに応じたスタイルシートの生成を行なう。

1. 名前変換の対応表　照合対象指定されているものについて、それぞれ一意な別名を用意する。これは、以下でパターン照合を行なう対象を、照合対象指定された文書要素に限定するために用いられる。

2. 1 パス目のスタイルシート　全ての変換規則について、construct 節の照合対象指定されている文書要素の名前を対応表に従って別名に置換する。これをテンプレートへ変換したものをスタイルシートとして利用する。これによって、次パス以降で照合対象のみにテンプレートを適用可能になる。

3. 2 パス目以降のスタイルシート　いずれの照合対象指定されている文書要素も適合しない変換規則を除去する。残ったものについて、照合対象指定されている文書要素名と、条件部に出現するそれを対応表に従って置換し、それから変換されたテンプレートをスタイルシートとして利用する。これによって、以前のパスで照合対象指定された文書要素のみがテンプレートの適用対象となる。また、出力が照合対象指定されていない部分は置換されないため、そのテンプレートが適用されることで自動的に元の名前に戻る。このため、処理が終了した文書要素にテンプレートが過剰に適用されることはない。

4. 終了時のスタイルシート　上記では名前が置換されたまま出力されるため、適合するテンプレートがない場合には別名のまま終了してしまう。この問題を解決するため、3 のスタイルシートの適用が終了した時点で、対応表に従って逆の置換を行なうスタイルシートを適用する。

5.3　変換例

図 1 に示した変換規則を、上記の規則に従って XSLT の記述に変換した結果の一部を図 4, 5 に示す。テンプレート 1〜3 が、それぞれ変換規則 1〜3 に相当する。

ここで、テンプレート 1 は変換規則 1 から生成されるテンプレートである。このスコーレム関数によるグルーピングは、(xsl:if) 以下へ変換されている。また、変換規則 2, 3 はそれぞれ、テンプレート 2, 4 及び、3, 5 へ変換されている。特に、変換規則 2 の `first` 節はテンプレート 4 で、変換規則 3 の `rank-by`〜`top` 節はテンプレート 5 で実現されている。

これらのうち、テンプレート 1 は 1 パス目の、残りのものは 2 パス目以降のスタイルシートとして利用される。1 パス目のスタイルシートには、5.2 節の方法に従って生成された他のテンプレートが含まれるが、実際には利用されないため、記述を省いた。

尚、照合対象指定された文書要素名には "-照合対象" という語尾を付加している。この語尾は全ての処理が完了した後、除去される。

6　関連研究

XSLT [6] は W3C で策定中の XML 用変換言語であり、その拡張として XQL [5] が提案されている。これらは提案操作系と同じルール指向の記述方式である。5 節で述べたように、これらは提案操作系と同様の文書操作機能を持っているが、その記述は低水準で手続き的なものとなる。また、文書操作処理における多段階の処理を直接記述できず、スタイルシートを明示的に複数回適用せねばならない。特に、不定回の適用が必要な場合には、XSLT 処理系の外部で繰り返しの終了を検出するような機構を設ける必要がある。また、ランキングによる絞り込みが記述内容処理において利用されるが、これに直接相当する処理が行なえない。

同じく W3C で策定中の XML-QL [4] もまた、XML に対する問合せ言語である。これは前述の通り、本操作系での構文とその処理方法の基本となっている。しかし、単一の問合せとして記述され、ユーザ関数の問合せとして記述可能なものしか利用できないため、部分的な変換処理や記述内容処理には十分な機能を提供できない。今後の拡張としてユーザ定義述語の導入が言及されているが、その適

```
<!-- テンプレート 1 -->
<xsl:template match="文書">
 <文書>
  <xsl:for-each match="記事">
   <xsl:variable name="d" expr="掲載日"/>
   <xsl:sort select="掲載日"/>

   <!-- スコーレム関数によるグルーピング :
        各〈掲載日〉の最初の〈記事〉のみに適用 -->
   <xsl:if test="count(from-preceding-siblings()
                                [掲載日=$d])=0">
    <!-- # <日別> -->
    <xsl:variable name=DMY" expr="NOTIFY()"/>
    <日別-照合対象>
     <xsl:for-each select="../記事 [掲載日=$d]">
      <xsl:copy-of select="."/>
     </xsl:for-each>
    </日別-照合対象>
   </xsl:if>
  </xsl:for-each>
 </文書>
</xsl:template>

<!-- テンプレート 2 -->
<xsl:template match="日別-照合対象">
 <日別>
  <!-- where / order-by / first -->
  <xsl:variable name="DMY" expr="NOTIFY()"/>
  <記事-先頭 個数="5">
   <xsl:for-each select="記事">
    <xsl:sort select="sim(., keywords)">
    <!-- construct # 〈記事〉 -->
    <xsl:variable name="DMY" expr="NOTIFY()"/>
     <記事-照合対象>
      <xsl:copy-of select="node()"/>
     </記事-照合対象>
    </xsl:for-each>
  </記事-先頭>
 </日別>
</xsl:template>
```

図 4: XSLT 表現

```
<!-- テンプレート 3 -->
<xsl:template match="記事-照合対象">
 <記事>
  <xsl:variable name="h" expr="見出し"/>
  <xsl:copy-of select="見出し"/>
  <本文>
   <!-- where / rank-by -->
   <xsl:variable name="DMY" expr="NOTIFY()"/>
   <本文-上位 個数="3">
    <xsl:for-each select="段落">
     <xsl:sort select="cosine($h,.)">
     <項目 元位置="position()">
      <xsl:copy-of select="."/>
     </項目>
    </xsl:for-each>
   </本文-上位>
  </本文>
 </記事>
</xsl:template>


<!-- テンプレート 4 : first 節を実現 -->
<xsl:template match="記事-先頭">
 <記事>
  <xsl:copy-of
       select="*[position()<=@個数]/node()"/>
 </記事>
</xsl:template>

<!-- テンプレート 5 : top 節を実現 -->
<xsl:template match="本文-上位">
 <xsl:variable name="num" expr="@個数"/>
 <xsl:for-each select="項目">
  <xsl:sort select="@元位置"/>
  <xsl:copy-of select="*[positioin()<=num]"/>
 </xsl:for-each>
</xsl:template>
```

図 5: XSLT 表現 (続き)

用範囲は現在のところ明確ではない。記述内容処理では、検証される条件や出力の生成等、述語に限らず幅広い処理機能の拡張が必要である。本操作系の記述内容処理関数は、返値の型が適正であれば任意の位置に記述する事ができ、また、必要に応じて記述内要素そのものを生成することも可能である。

YAT [11] は木構造データモデル上のシステムであり、YATL はその言語として提案されている。これはパターン間の変換に基づくルール指向の言語であり、や

はり高い文書構造操作の能力を持つ。また、文字列処理等のための拡張関数が利用可能である。更に、YATL で記述された複数のプログラムを順次適用することで、多段階の変換をサポートし、これらのプログラムをあらかじめ合成する事で一つのプログラムに変換することも可能である。しかし、類似検索や要約等については考慮されておらず、ランキングによる件数の絞り込み等、文書操作に必要な記述力を持たない。YATL では頭部と本体からなるルールとしてプログラムを与えるのに対し、本研究での変換規則記述は既存の問合せ言語との親和性が高い。

Lorel [8] はラベル付き有向グラフに基づくデータモデルである OEM に対する問合せ言語であり、OQL の拡張として捉える事ができる。これは簡潔さを重視しており、再構成の能力をあまり持たない。一方、UnQL [9] もまた、OEM と同様の有向グラフに基づくデータモデルに対する問合せ言語であるが、高い再構成の能力を持つ。これらは、従来の問合せ言語と同様、構造の位置関係や、値の単純な関係しか扱う事ができず、記述内容処理機能は十分ではない。

7 まとめ

従来の構造化文書を対象として問合せ言語では、文書構造に対する比較的強力な操作体系を有するものの、記述内容処理に関しては、文字列比較等の極めて限定された機能しか提供していない。しかし、大量の構造化文書データベースを効率的に利用するには、これら文書構造操作と、より高度な記述内容処理の統合が重要である。本稿では、これらの処理をルール指向の変換規則と DTD 固有に定義された記述内容処理関数を利用した構造化文書操作系を提案した。また、本操作系における記述を XSLT による記述に変換する方法について言及した。

今後、XSLT への変換における最適化と制約の緩和について検討し、本操作系の記述力を検証していく必要がある。また、本稿で示したアプローチを用いて実際に処理系の実装を行なう予定である。

参考文献

[1] 品川 徳秀, 北川 博之, 石川佳治. “文書構造変換規則と記述内容処理関数に基づく構造化文書操作記述方式”, アドバンスト・データベース・シンポジウム '99, 情報処理学会シンポジウムシリーズ, Vol. 99, No. 19, pp. 123-132,

[2] World Wide Web Consortium, http://www.w3.org/.

[3] R. Sacks-Davis, T. Arnold-Moore, J. Zobel. Database systems for Structured Documents, *International Symposium on ADTI '94*, pp. 272-283, Nara, 1994.

[4] A. Deutsch, M. Fernandez, D. Florescu, A. Levy and D. Suciu. A Query Language for XML, *Proceedings of the Eighth International World Wide Web Conference (WWW8)*, Computer Networks, Vol. 31, No. 11-16, pp. 1155-1169, 1999.

[5] J. Robie, J. Lapp and D. Schach. XML Query Language (XQL). *The Query Languages Workshop (QL'98)*, http://www12.w3.org/TandS/ QL/QL98/pp/xql.html, 1998.

[6] J. Clark (ed.). *XSL Transformations (XSLT)*, http://www.w3.org/TR/WD-xslt, 1999.

[7] J. Clark and S. DeRose. *XML Path Language (XPath) Version 1.0 (working draft)*, http://www.w3.org/TR/WD-xpath, 1999.

[8] S. Abiteboul, D. Quass, J. McHugh. J. Widom and J. Wiener. The Lorel Query Language for Semistructured Data, *International Journal on Digital Libraries*, Vol. 1, No. 1, pp. 68-88, 1997.

[9] P. Buneman, S. B. Davidson, G. G. Hillebrand and D. Suciu. A Query Language and Optimization Techniques for Unstructured Data, *Proceedings of ACM-SIGMOD '96*, pp. 506-516, Motreal, 1996.

[10] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy and D. Sucie. Catching the Beat with Strudel: Experiences with a Web-site Management System, *Proceedings of ACM-SIGMOD '98*, pp. 414-425, Seattle, 1998.

[11] S. Cluet, C. Delobel, J. Simeon and K. Smaga. Your Mediators Need Data Convention!, *Proceedings of ACM-SIGMOD '98*, pp. 414-425, Seattle, 1998.

[12] D. Konopnicki and O. Shinueli. W3QL: Query System for the World Wide Web, *Proceedings of Twenty-First Conference on VLDB*, pp. 54-65, Zurich, 1995.

[13] A. Mendelzon, G. Mihaia and T. Milo. Querying the World Wide Web, *International Journal on Digital Libraries*, Vol. 1, No. 1, pp. 54-67, 1997.

[14] 田島 敬史. "半構造データのためのデータモデルと操作言語", 情報処理学会論文誌データベース, Vol. 40, No. SIG 3 (TOD 1), pp. 152-170, 1999.

[15] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading, MA, 1989.

[16] 奥村 学, 難波 秀嗣. "テキスト自動要約に関する研究動向", 自然言語処理「テキスト要約のための言語処理」特集号, Vol. 6, No. 6, 1999.

[17] H. A. Hearst. Subtopic Structuring for Full-Length Document Access, *Proceedings of ACM-SIGIR '93*, pp. 59-68, Pittsburg, 1993.