自然言語処理 110－14
（1995．11．18）

実数ベクトルによる語句の表現の試み

高橋 直人*
電子技術総合研究所 自然言語研究室†

要旨

本稿では、日本語語句の意味を実数ベクトルを用いて表現する方法について述べる。この意味表現ベクトルは、ニューラルネットを用いて、コーパスからボトムアップに学習する。この際、似た意味を持つ語句には似たベクトルを割り当てることを目標とする。また複数の語句に同一の意味表現ベクトルが割り当てられることを避けるため、意味表現ベクトルから語句の表層表現への逆変換のタスクを新たな制約として与える。さらに同じ文法範疇の語句同士だけでなく、異なる文法範疇に属する語句の間でも意味的類似度が計算できるように、意味表現ベクトルの次元数は常に一定とする。計算機上で簡単なシミュレーションを行ったところ、本稿の手法を用いることで、ほぼ期待通りの結果を得ることができることがわかった。

An attempt to represent words and phrases by real vectors

TAKAHASHI Naoto
Natural Language Section, Electrotechnical Laboratory

This paper presents a method to represent the meaning of Japanese words and phrases by real vectors. These vectors are called *semantic representation vectors* (SRV's), and are acquired from a corpus by using neural networks, so that similar SRV's are allocated to similar words or phrases. In addition to that, a task of reverse conversion, from SRV to surface representation, is imposed to the neural networks so that no two words or phrases will share the same SRV. Regardless of the syntactic category, all SRV's are of the same rank. This feature guarantees an easy calculation when two SRV's of different categories are compared. According to the result of a preliminary experiment, the method presented in this paper gives satisfactory SRV's.

*ntakahas@etl.go.jp
†〒305 茨城県つくば市梅園 1-1-4

1 序論

日本語文における係り受け、あるいは英語文における pp-attachment 等を正しく解析するためには、意味情報の利用が必須である。このためには当然、語句の意味を何らかの形で表現しなくてはならない。

語句の意味を表現する際には、意味間の類似度が計算できるような表現形式を採用することが重要である。さらに類似度が数値で表現できるならば、意味的な近さの度合が容易に比較できて便利である。また語句と語句との類似度を測る尺度は1通りではないので、多次元的な比較が可能でなければならない。実数ベクトルは、類似度を数値で表わすことができ、しかも多次元である。したがって、語句の意味表現形式として有用であると考えられる。

また、各文法範疇に与える意味表現ベクトルの次元はすべて同一であることが望ましい。なぜなら語句の表層構造が互いに異なっている場合でも、それらの間の意味的類似度を容易に計算できるからである。表層構造の異なる語句の意味を比較する必要があることは、同一の意味内容を表す表層構造が複数存在することより明らかである。たとえば「数学を勉強する」と「数学の勉強をする」はほぼ同じ意味であるが、表層の木構造は異なる。また「毒殺」と「毒を用いて殺すこと」も意味的には同じと言ってよいが、前者は語で後者は句という違いがある。

本稿では以上の点を踏まえ、さまざまな文法範疇に属する日本語語句の意味を、同一次元の実数ベクトルを用いて表現する方法について述べる。この意味表現ベクトルは、コーパス中の用例に基づき、ニューラルネットを用いてボトムアップに学習する。また、異なる内部構造を持つ語句をすべて同一次元のベクトルで表現するために、リカレント型のニューラルネットを用いる。

2 意味表現ベクトル

語の意味を実数ベクトルで表現する研究としては Miikkulainen の方法が興味深い。Miikkulainen は FGREP と呼ぶニューラルネットを用い、人工的に作った小規模コーパスから語の意味表現を自動的に学習することが可能であることを示した [1]。FGREP において、それぞれの語には初期値としてランダムなベクトルが与えられる。次にこれらのベクトルを用いて、コーパス中の例文の学習を試みる。実際の出力と目標出力との誤差は、バックプロパゲーションによってニューラルネット中を逆向きに進み、最後に入力に用いられた語彙ベクトルそのものを書き替える。このプロセスを繰り返すことにより、類似した意味を持つ語[1]のベクトル表現同士は、次第に類似したものへと近付いていく。

FGREP の問題点の1つとして、「最悪の場合、すべての語の意味表現が同一値に収束する可能性がある」という点を挙げることができる。また、たとえそうはならなかったとしても、コーパス中でまったく同じ使われ方をしている語の表現は、必然的に同一値に収束する。Miikkulainen は、同一値に収束した語を互いに区別するため、ID 部と呼ぶ特別のフラグを学習終了語に付加した。ID 部は語の識別のための2進数を表わす。たとえば a, b, c の3語の意味表現が [0.3 0.8 0.2] という同一値に収束したとする。このとき ID 部を追加した後の a, b, c の表現は、たとえばそれぞれ [0.3 0.8 0.2 0 0], [0.3 0.8 0.2 0 1], [0.3 0.8 0.2 1 0] のようになる。しかしこの方法はあまりに ad hoc である。

本稿で述べる意味表現ベクトル獲得の方法は、基本的にリカレント型の FGREP を日本語向きに手直ししたものと言える。これによって、類似した語には類似した意味表現ベクトルが与えられるようになる。ただし ID 部は採用しない。その代わり 意味表現から表層表現への復元可能性 を新たな制約として追加する。すなわちある意味表現ベクトルが与えられたとき、それから元の語が復元できるようにしながら学習を進める。この逆変換のタスクが制約として働くので、2つ以上の意味表現ベクトルがまったく同一の値を取ることはなくなる。

[1]学習しているのはコーパス中の用例であるから、厳密には「類似した意味を持つ語」ではなく、「類似した使われ方をする語」である。文献 [1] と同様、本稿でもこの両者の間には強い相関があるものとする。

名詞	主語	パスカル, ニュートン, ライプニッツ, ガリレオ
	目的語	寿司, 天ぷら, すき焼き
動詞		食べた, 注文した
助詞		が, を

表 1: 実験に用いた語

3 コーパス

今回学習の対象とするのは、日本語の動詞文である。日本語では必要に応じて (あるいは「不必要」に応じて) 表層の格要素を削除することが可能である。よって、V、NPV、NPNPV、etc. はすべて動詞文であり、単に内容の詳しさだけが違うものと言える。そこで以下の簡単な文法を考える。

$$S \rightarrow [N\ P]^*\ V \tag{1}$$

すなわち日本語の動詞文を、「名詞 + 格助詞」という連用修飾句が動詞の前に 0 個以上付いたもの、と定義する。また、文型としては次の 2 通りを考える。

< 主語名詞 > が < 目的語名詞 > を < 動詞 > (2)

< 目的語名詞 > を < 主語名詞 > が < 動詞 > (3)

語彙項目としては、表 1に示す 11 語を設定する。主語と書かれた名詞は「が」の前のみに許され、目的語と書かれた名詞は「を」の前にのみ許される。

コーパス内には、以上の規則に従うすべての文 (主語4 × 目的語3 × 動詞2 × 語順2 = 48文) が、ちょうど 1 回ずつ含まれている。後の処理のことを考え、各文はあらかじめ語の単位に区切られている。またそれぞれの語には 0 ～ 10 のシリアル番号が付与されており、実際のファイル中には文字表記の代わりにそのシリアル番号が記載されている。

4 ニューラルネットの構成

図 1に、コーパスから意味表現ベクトルを獲得するためのシステムを示す。全体は 3 つの部分に分けられ、各部分はニューラルネットで構成されている。エンコーダ部は単純な 3 層ニューラルネットで、コーパスから読んだ語の表層表現をベクトルに変換する。コンバータ部はリカレント型ニューラルネットで、意味表現ベクトル学習の中心部である。デコーダ部はエンコーダ部と同様、単純な 3 層ニューラルネットで、語をベクトルから表層表現へ逆変換する。以下、各々のニューラルネットに関して個別に説明する。

4.1 エンコーダ部

エンコーダ部の役目は、個々の語をベクトルに変換することである (図 2)。

エンコーダ部の入力層には、語の 表層表現 を与える。表層表現は、それから元の語が特定できるようなパターンであればどんなものでもよい。今回の実験では、それぞれの語に付与されているシリアル番号の局所表現を用いた。ここでシリアル番号 n の局所表現とは、n 番目の要素のみが 1 で、残りはすべて 0 であるようなベクトルのことを意味する。局所表現は、3 層のニューラルネットを通る間に分散表現に変換され、コンバータ部およびデコーダ部へと渡される。

エンコーダ部の学習に際しては、出力すべき意味表現ベクトルが教師信号として直接与えられるのではなく、コンバータ部およびデコーダ部から逆向きに伝搬されてくる誤差信号が用いられる。

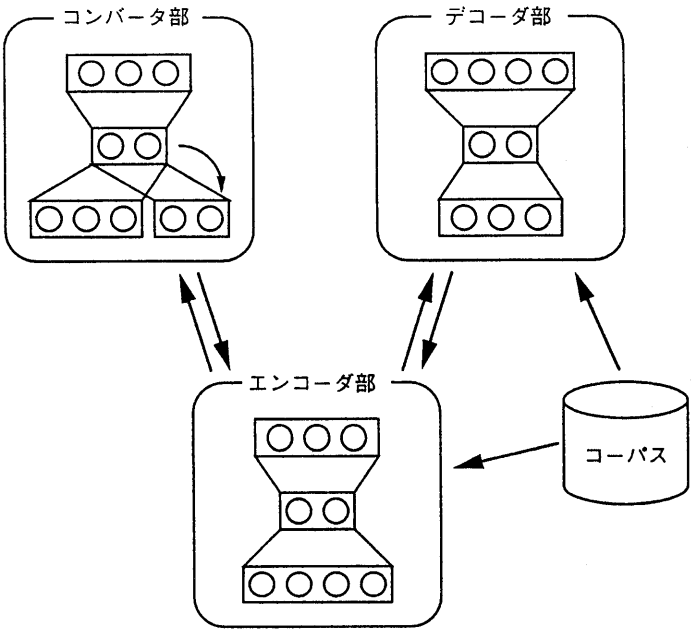

図１：ニューラルネット・システムの全体像

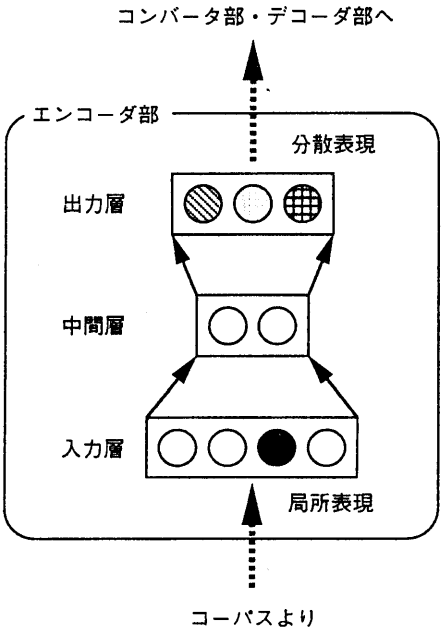

図２：エンコーダ部の構成

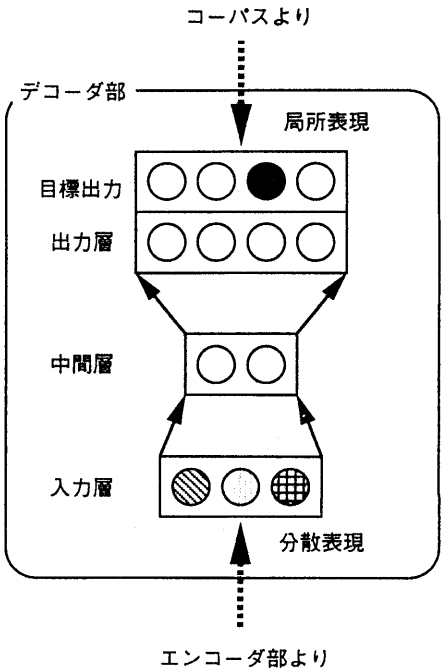

図３：デコーダ部の構成

4.2 デコーダ部

コンバータ部での学習 (これについてはすぐ下で述べる) は、ある自明な解に収束する可能性がある; すべての語句の意味表現ベクトルが同一値になるというのがそれである。デコーダ部は、この自明な解への収束を避けるように働く。さらに、たとえある 2 語の用例が全く同じであっても、それぞれには異なった意味表現ベクトルが与えられるように働く。

すべての意味表現ベクトルがそれぞれ異った値に収束するようにするには、意味表現ベクトルを表層表現に逆変換するというタスクを与えればよい。なぜなら、もしある 2 語の意味表現ベクトルが同一であるならば、異なる表層表現を復元し分けることはできないからである。デコーダ部は意味表現ベクトルを入力とし、表層表現を目標出力とする学習を行う (図 3)。この逆変換タスクによって、異なる語には異なる意味表現ベクトルが与えられるようになる。

目標出力と実際の出力との誤差は、バックプロパゲーションに従ってデコーダ部の中を逆に進み、入力層まで戻った時点でエンコーダ部に渡される。エンコーダ部はこの誤差信号を利用してリンクの重みを更新する。

4.3 コンバータ部

異なる内部構造を持つ語句をすべて同一次元のベクトルで表現するため、コンバータ部にはカレント型のニューラルネットを用いる [2, 3, 4, 5, 6]。入力層は名詞部と助詞部の 2 つに分かれ、出力層は agent 部、patient 部、action 部の 3 つに分かれている。コンバータ部の役割は、類似した語に類似した意味表現ベクトルを割り当てることである。以下では特にことわらない限り、「X の意味表現ベクトル」を V(X) と略記する。

1 文の学習は 3 サイクルからなる。例として「パスカルが寿司を食べた」という文を取り上げる。この場合の目標出力は agent 部が V(パスカル)、patient 部が V(寿司)、action 部が V(食べる) である[2]。これら目標出力の意味表現ベクトルには、エンコーダ部によって生成されたものを用いる。サイクルが進むにつれて、コンバータ部の中間層には順次、V(食べた)、V(寿司を食べた)、V(パスカルが寿司を食べた) が形成される。

まず最初に、V(食べた) を、エンコーダ部からコンバータ部の中間層に直接入力する[3](図 4の矢印 (1))。次に中間層から出力層に信号を送り (同図 (2))、目標出力との誤差を求める。この誤差はバックプロパゲーションに従って出力層から中間層へと逆向きに伝搬され (同図 (3))、更に中間層からエンコーダ部に戻されて (同図 (4))、V(食べた) の学習に利用される。

第 2 のサイクルは、第 1 のサイクルで入力した V(食べた) と、新たに入力する V (寿司) および V(を) とから、V(寿司を食べた) を生成する過程である。まず、第 1 サイクルの中間層のパターン (すなわち V(食べた)) を context 層にコピーする (図 5(1))。同時に V(寿司) および V(を) を、エンコーダ部からコンバータ部の入力層に入力する (同図 (2))。これらの意味表現ベクトルは同時に中間層に送られ (同図 (3))、V(寿司を食べた) となる。次にこの意味表現ベクトルが出力層に送られて (同図 (4))、目標出力と比較される。出力層での誤差信号は、コンバータ部の中を逆向きに伝搬し (同図 (5)(6))、最後にエンコーダ部に返されて (同図 (7))、V(寿司) および V(を) の学習に利用される。

第 3 サイクルの学習も第 2 サイクルと同様に進む。まず、中間層から context 層にコピーされた V(寿司を食べた) と、エンコーダ部から送られてくる V(パスカル) および V(が) とから、V(パスカルが寿司を食べた) が生成され、新たな中間層パターンとなる。これが出力層に送られて誤差が計算され、コンバータ部・エンコーダ部の順に逆伝搬されて V(パスカル) および V(が) の学習に用いられる。

コーパス中には、動詞が同じで主語および目的語の異なる文が多数含まれている。したがって学習の第 1 サイクル、すなわち動詞のみが与えられた状態で agent 部と patient 部を正しく出力することは理論的に不可能である。しかし、あえてそのような学習を繰り返すことによって、主語あるいは目的語が与えられていない状態では、agent 部には主語名詞の表均パターン、patient 部には目的語名詞の平均パターンが出力されるようになると期待される。こ

[2]すなわち、V(パスカルが寿司を食べた) と V(寿司をパスカルが食べた) とは同じ意味表現ベクトルを持つことになる。

[3]すぐ下で述べるように、第 2 サイクルでは V(寿司)・V(を)・V(食べた) の 3 つの意味表現ベクトルから V(寿司を食べた) の生成を試みる。このとき、V(食べた) は context 層に置かれていなければならず、そのためには第 1 サイクルで中間層に V(食べた) が置かれていなければならない。中間層に直接ベクトルを入力するのは例外的であるが、この場合やむを得ない。

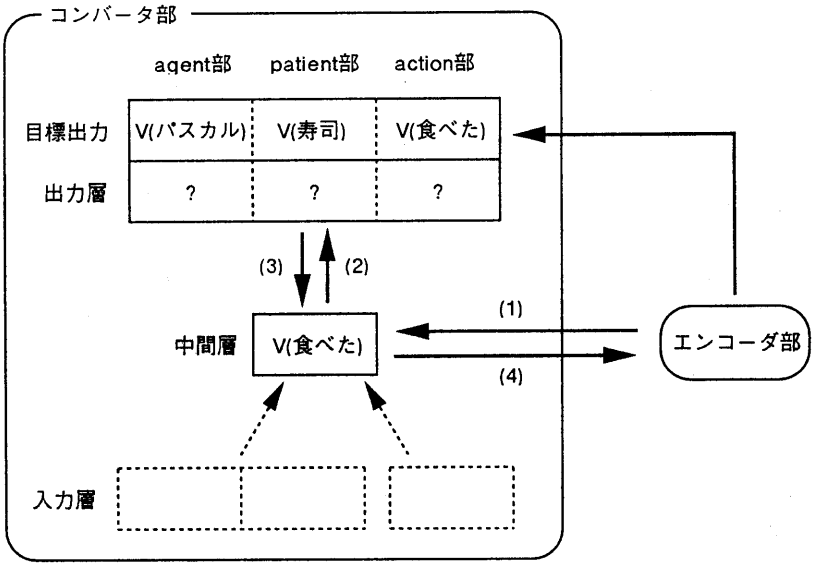

図4：コンバータ部における学習の第1サイクル

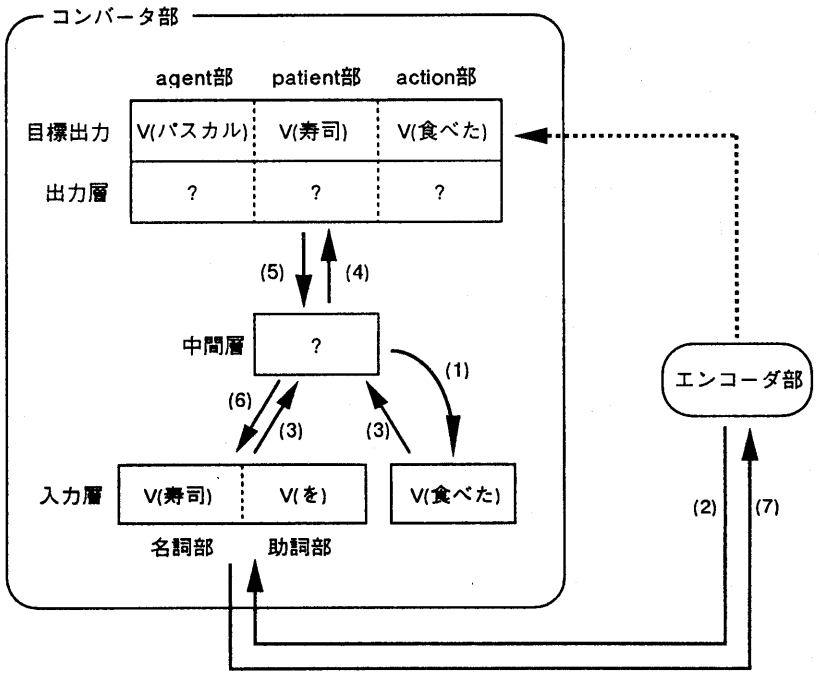

図5：コンバータ部における学習の第2サイクル

ネットワーク	層・部	ユニット数
エンコーダ部	入力層	11
	中間層	7
	出力層	4
デコーダ部	入力層	4
	中間層	7
	出力層	11

ネットワーク	層・部		ユニット数
コンバータ部	入力層	名詞部	4
		助詞部	4
	中間層		4
	context 層		4
	出力層	agent 部	4
		patient 部	4
		action 部	4

表 2: 各ニューラルネットの各部におけるユニット数

のように、与えられていない格に対して、その格を占めうる意味表現ベクトルの平均パターンを出力するという特性は、日本語のように表層の格要素が省略できる言語において有用であると考えられる。

5 シミュレーションとその結果

前節で述べた3つのニューラルネットをシミュレートするプログラムを作成し、実際に動作させた。各ニューラルネットの各部におけるユニットの数は、表2に示した通りである[4]。3つのネットワークすべておいて、学習の係数 η および慣性 α はどちらも 0.1 とした。また、すべてのユニットの入出力関数には、標準的なシグモイド関数を用いた。したがって、意味表現ベクトルの各要素がとりうる値は 0 と 1 の間 (両端を含まない) である。

学習は、コーパス中の各文を先頭から1つずつ順に用いて行った。最後の文が終ったら、また先頭に戻って同じことを繰り返した。リンクの重みの更新は、各データを呈示するごとに行った。

表3に、学習が進むにつれて個々の語の意味ベクトル表現がどのように変化したかを示す。

まず、全文の学習を 750 回繰り返したときにエンコーダ部が生成した意味表現ベクトルでを見ると、名詞は全部まとまって1つのグループを作っており、動詞は動詞でまた別のグループを使っていることがわかる。特に名詞同士の間にはほとんど差がないので、この時点ではまだデコーダ部の働きが顕在化していないと言える。

一方2つの助詞は、ベクトル空間の中で互いにほぼもっとも遠い場所に位置している。この理由は次のように考えられる: コーパス中には、ガ格が先に来る文とヲ格が先に来る文が同数存在している。コンバータ部は、入力層の助詞部に「が」が来るか「を」が来るかで、そのとき名詞部に与えられた名詞を出力層の agent 部と patient 部のどちらに出力するか決定しなくてはならない。したがって助詞の識別は非常に重要である。しかもこれら2つの助詞は各文に必ず1回ずつ現われるので、他の語に比較して学習される頻度が高い。以上の理由により、「が」と「を」は極めて速やかに、識別が最も容易な2つの値 (すなわち [0 0 0 0] と [1 1 1 1]) に分離するものと考えられる。

学習回数が 1800 回になると、主語になる名詞と目的語になる名詞が、それぞれ別のグループを形成するようになる。また2つの動詞も、互いにはっきり区別されるようになる。

4200 回を過ぎると、個々の名詞の区別がつくようになる。しかしそれでも、主語名詞同士、あるいは目的語名詞同士でそれぞれ1かたまりのグループを作っているように見える。これが、「似た語には似た意味表現ベクトルが与えられるが、完全に同一なものはない」という本来の目的にかなった状態である。

学習が更に進んで繰り返し回数が 10000 回になると、個々の名詞の分離が大きくなり、すべての語がベクトル空間内にばらばらに分布するようになる。これは、コンバータ部の働きよりも、デコーダ部の働き (個々の語の意味ベクトル表現を異なったものにしようという働き) が強く現れた状態である。コンバータ部は3層のニューラルネットなので、十分な数の中間層ユニットがあれば、個々の語の意味表現ベクトルがどんなものであっても、原理的には与え

[4] コンバータ部の各層におけるユニット数は、エンコーダ部の出力層のユニット数から自動的に決定される。

		繰り返し回数															
		750				1800				4200				10000			
名詞 (主語)	パスカル	.72	.77	.66	.58	.98	.99	.82	.55	.87	.99	.95	.51	.58	.99	.99	.39
	ニュートン	.72	.77	.66	.58	.98	.99	.82	.55	.46	.99	.97	.48	.25	.99	.99	.48
	ライプニッツ	.72	.77	.66	.58	.98	.99	.82	.55	.09	.99	.88	.23	.00	.62	.49	.11
	ガリレオ	.72	.77	.66	.58	.98	.99	.82	.55	.50	.99	.94	.29	.35	.99	.83	.09
名詞 (目的語)	寿司	.71	.77	.65	.58	.95	.99	.62	.29	.15	.99	.94	.48	.20	.99	.98	.32
	天ぷら	.71	.76	.65	.58	.95	.99	.62	.30	.94	.95	.66	.08	.91	.15	.08	.00
	すき焼き	.72	.77	.66	.58	.94	.99	.58	.27	.96	.94	.06	.19	.70	.99	.04	.00
動詞	食べた	.43	.50	.32	.27	.76	.96	.17	.04	.32	.92	.10	.11	.19	.79	.02	.00
	注文した	.37	.45	.27	.22	.20	.60	.02	.00	.00	.46	.00	.01	.18	.38	.01	.00
助詞	が	.98	.98	.98	.96	.99	.99	.99	.96	.11	.99	.85	.99	.01	.99	.92	.47
	を	.03	.04	.01	.01	.01	.05	.00	.00	.01	.02	.00	.00	.00	.06	.00	.00

表 3: 学習の繰り返し回数と、それに伴う意味表現ベクトルの変化

られたタスクを学習することが可能である。つまり、コンバータ部は学習が進むにつれて入力データに対して柔軟になり、その結果デコーダ部に与えられた制約の方が意味表現ベクトルの形成に強く反映されるようになったものと考えられる。

6 おわりに

実験結果より、ほぼ期待通りの意味表現ベクトルを得ることがわかった。ただし、過学習を避けるためのメカニズムを探る必要がある。また今後は、動詞文以外の語句に対しても、同様の手法を試みるつもりである。

謝辞

本研究は、リアルワールドコンピューティング (RWC) 研究計画の一環として行われた。研究を進めるにあたり、貴重な助言をいただいた電子技術総合研究所自然言語研究室ならびに同 新情報計画室の諸氏に感謝いたします.

参考文献

[1] Miikkulainen, R.: *Subsymbolic Natural Language Processing*, MIT Press, pp.47–83, 1993.

[2] Pollack, J. B.: Recursive Distributed Representations, Artificial Intelligence, 46, pp.77–105, 1990.

[3] Chrisman, L.: Learning Recursive Distributed Representations for Holistic Computation, Connection Science, Vol.3, No.4, pp.345–366, 1991.

[4] Berg, G.: A Connectionist Parser with Recursive Sentence Structure and Lexical Disambiguation, Preceedings of AAAI-92, pp.32–37, 1992.

[5] Sperduti, A. and Starita, A.: An Example of Neural Code: Neural Trees Implemented by LRAAMs, Artificial Neural Nets and Genetic Algorithms, Springer-Verlag, pp.33-39, 1993.

[6] Kwasny, S. C. and Kalman, B. L.: Tail-recursive Distributed Representations and Simple Recurrent Networks: Connection Science, Vol.7, No.1, 1995.