情報処理学会論文誌　数理モデル化と応用　Vol.12　No.1　23–31 (Mar. 2019)

Word2Vecにおける加算型単語ベクトルの効果と応用

内田 脩斗[1,a)]　吉川 大弘[1,b)]　古橋 武[1,c)]

受付日 2018年8月23日，再受付日 2018年10月17日，
採録日 2018年11月2日

概要：Word2Vec は，単語の分散表現を獲得する最も一般的な手法の 1 つであり，自然言語処理分野における構文解析や文書分類などに適用した研究も数多く報告され始め，その有用性が示唆されている．この手法では，2 種類の分散表現が生成されるが，従来は一般的に，その一方のみを利用して単語の分散表現としている．一方で，それらを加算した単語ベクトル W_{ADD} を利用することによる，意味関係性能の向上が報告されている．しかし，その際示された実験では，同時にパラメータのチューニングも行っており，性能向上の要因が不明瞭なものとなっている．また，W_{ADD} を実タスクに応用した際に，精度への貢献が期待できるかどうかについては明示されていない．そこで本論文では，アナロジータスクにおいて分散表現の意味関係性能を評価し，観測的事実に基づいて W_{ADD} の精度向上原因の解析を行う．加えて，文書分類タスクにおいて，各分散表現による分類精度の比較・検討を行い，その有用性について報告する．

キーワード：Word2Vec，分散表現，加算ベクトル，アナロジータスク，文書分類

Effect and Application of Additional Vector in Word2Vec

Shuto Uchida[1,a)]　Tomohiro Yoshikawa[1,b)]　Takeshi Furuhashi[1,c)]

Received: August 23, 2018, Revised: October 17, 2018,
Accepted: November 2, 2018

Abstract: Word2Vec is one of the most common methods for acquiring a distributed representation of words. In the field of natural language processing, many studies applying Word2Vec to syntactic analysis and document classification have been reported and its usefulness is suggested. In this method, two kinds of distributed representations are generated, and generally, only one of them is actually used. On the other hand, the improvement of semantic relation performance has been reported by using the word vector W_{ADD} generated by adding two kinds of distributed representations. However, in the experiment, other parameters are tuned at the same time, and factors of the improvement are not clear. Moreover, it is unknown whether the improvement of accuracy can be expected when W_{ADD} is applied to real tasks. Therefore, in this study, we evaluate the semantic relation performance of distributed representations with analogy tasks and analyze the cause of improvement of W_{ADD} based on observational facts. In addition, we conduct a document classification task using each distributed representation and report its usefulness.

Keywords: Word2Vec, distributed representation, additional vector, analogy task, document classification

1. はじめに

インターネットの普及にともない，膨大な情報が生成・拡散されている現代において，テキストデータの自動解析や情報抽出技術は，様々な場面での応用が期待されており，研究が進められている．また，言語処理分野においては，単語を原子単位（要素）として取り扱うことが一般的であり，その表現方法は重要な問題として扱われている．従来から広く普及している単語の表現手法として，個々の単語に固有のインデックスを与えることで単語を表現する One-hot 表現がある [1], [2]．この手法は非常にシンプルで分かりやすい反面，各単語が独立であることを前提としているため，同義語や類似語がまったく関係のない単語とし

[1] 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University, Nagoya, Aichi 464–8603, Japan
[a)] uchida@cmplx.cse.nagoya-u.ac.jp
[b)] yoshikawa@cse.nagoya-u.ac.jp
[c)] furuhashi@cse.nagoya-u.ac.jp

て扱われることがある．また，1 単語に 1 次元を割り当てるため，ボキャブラリ数が増えれば増えるほど高次元となり，計算コストがかかるという問題がある．これを解決するために，単語ベクトルを生成する手法が数多く研究されているが [3], [4], [5], [6]，Mikolov らが発表した Word2Vec [7] は，大規模コーパスから教師なし学習を行うことで，自動的に語義の似た単語が類似したベクトルを持つ分散表現と呼ばれる単語ベクトルを生成することができる．これにより，従来では困難であった単語ベクトル間での意味の演算が可能となり，たとえば，学習された分散表現に対し，

$$vector(Paris) - vector(France) + vector(Italy)$$

により算出されるベクトルが $vector(Rome)$ に，

$$vector(king) - vector(man) + vector(woman)$$

により算出されるベクトルが $vector(queen)$ に，それぞれ近くなるという性質を持っている．この分散表現を適用した研究が多く報告され始めており [8], [9], [10], [11]，分散表現の意味関係性能を高めることはきわめて有益であると考えられる．

さらに，Word2Vec には多くの派生モデルが提案されており，Pennington らが提案した Glove [12] は，アナロジータスクにおいて Word2Vec よりも精度が向上することを報告している．一方，Levy ら [13] は，上記の手法を様々なアナロジータスクにおいて性能比較を行い，Word2Vec と Glove について，報告されたほどの性能差はないことを示している．その原因の 1 つとして，各々の手法で生成される 2 種類の単語ベクトルの利用方法の差について言及している．Word2Vec では単語ベクトル W_{IN} を，Glove では単語ベクトル W_{IN} と文脈ベクトル W_{OUT} を統合したベクトル W_{ADD} を評価に利用しており，Word2Vec においても同様の処理を行って比較を行った場合，Word2Vec が Glove の性能を上回ったことを報告している．しかし，文献 [13] で示されている実験では，同時にパラメータのチューニングも行っており，性能向上の要因が，加算ベクトルにあるのか，用いたパラメータにあるのかが不明瞭となっている．また，パラメータを一定にした W_{IN} と W_{ADD} の比較実験も行われているが，精度が向上している場合と低下している場合が確認でき，実際に W_{ADD} を実タスクに応用した際に，精度への貢献が期待できるかどうか不明である．また，英語のデータセットにおける評価は数多く報告されているが，日本語のデータセットにおける有効性の検証は報告されていない．

そこで，本論文では，従来の単語ベクトル W_{IN} と W_{ADD} をアナロジータスクで評価し，結果を検討する．さらに，観測的事実に基づいて W_{ADD} の精度向上原因の解析を行う．加えて，日本語と英語のニュース記事による文書分類タスクにおいて，W_{IN} と W_{ADD} の分類精度の比較・検討を行い，その有用性について報告する．

2. 関連研究

Word2Vec は分散表現の獲得手法として，最も広く普及している手法の一種である．**図 1** は，Word2Vec の学習モデルを表した図である．このとき，入力層と出力層の次元数はボキャブラリ数，隠れ層は埋め込み次元数に対応している．Word2Vec は，ニューラルネットワークを用いて分散表現を自動的に獲得する．そのモデル構造は，言語処理分野でよく用いられる分布仮説（同じ文脈で出現する単語は同じ意味を持つこと）[14] に基づいており，文脈上のある単語に対して，共起しやすい単語を予測するタスク設定になっている．つまり，入力層に入力された単語に対して，文脈上でその単語の周辺に出現している単語の出現確率が大きくなるように各層の重み（入力側の重み W_{IN}，出力側の重み W_{OUT}）を更新する．たとえば，ある文脈上において，「dog」という単語の周辺に「animal」や「cute」という単語が出現したとすると，$p(\overrightarrow{animal_{OUT}}, \overrightarrow{cute_{OUT}} | \overrightarrow{dog_{IN}})$ が 1 に近づくように W_{IN} と W_{OUT} が更新される．また，Word2Vec には CBOW モデルと Skip-gram モデルが存在し，CBOW は学習の高速化，Skip-gram は分散表現の意味関係性能の面でそれぞれメリットがある．本論文では，Skip-gram を対象とする．

さらに，両手法には Negative Sampling と呼ばれる高速化手法が用いられている．Negative Sampling では，共起する単語の予測を行う処理に加えて，共起しない単語（負例）の予測を行うことでモデルの近似が可能となり，学習の高速化と精度の向上を達成している．Skip-gram Negative Sampling（SGNS）の更新式は，式 (1)，(2) のようになる．

$$W_{Ii}^{(new)} = W_{Ii}^{(old)} - \eta \sum_{v \in W'_O \cup V_{Neg}} (\sigma(W_I \cdot W'_v) - t_v) W'_{vi} \quad (1)$$

$$W_{ij}^{\prime(new)} = W_{ij}^{\prime(old)} - \eta(\sigma(W_I \cdot W'_j) - t_j) W_{Ii} \quad (2)$$

図 1 のように，3 層のニューラルネットワークを用いているため，入力側の重み行列 W_{IN} と出力側の重み行列 W_{OUT} の更新が行われる．なお，数式では W_{IN} を W，W_{OUT} を W' で表記している．W_I は入力単語ベクトル，W'_j は j 列目の出力単語ベクトル，i は各ベクトルの i 番目，η は学

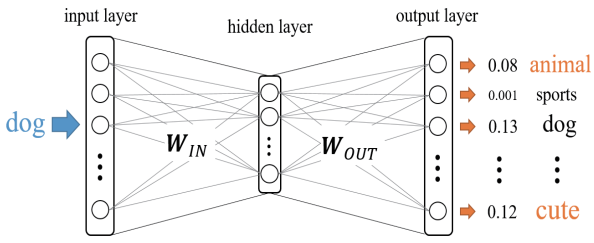

図 1　Word2Vec の学習モデル図
Fig. 1　Learning model of Word2Vec.

習率，σ はシグモイド関数，t_v は v 番目の出力単語が入力単語 I に対して正解単語か否かのフラグを表している．また，V_{Neg} は負例を生成するノイズ分布である．アルゴリズムの詳細については，文献 [15] を参照されたい．

Word2Vec の派生モデルは数多く報告されており，Pennington らが提案した Glove [12] は，Word2Vec の改良モデルであり，CBOW と Skip-gram の両方の特徴を含んだ分散表現が獲得できるといわれている．そのほかにも，文書の分散表現を獲得する doc2vec [16] や，分散表現に LDA の機能を持たせる lda2vec [17]，文脈を考慮した分散表現の生成が可能な ELMo [18] など，様々な手法の基礎となっている．

一方，前述のように，Word2Vec では，W_{IN} と W_{OUT} の 2 種の単語ベクトル行列が生成されている．また，Word2Vec では，W_{IN} を単語の分散表現として実際に利用しており，W_{OUT} は W_{IN} を獲得するために生成される副産物的な重みとして無視されることが一般的である．しかし W_{IN} と W_{OUT} は更新式が異なるため，それぞれのとらえる特徴も異なっていると考えられる．既存研究では，W_{IN} は単語の意味関係をとらえたベクトル，W_{OUT} は共起関係をとらえたベクトルとされる．また，上述の Glove では，経験的に，W_{IN} をそのまま用いるより，$W_{IN}+W_{OUT}$ とした分散表現 W_{ADD} を用いることで，アナロジータスクの精度が向上するため，標準的に W_{ADD} が利用できるプログラム仕様となっている．さらに，Levy ら [13] は，その有用性についても述べている．W_{ADD} の生成には，学習後の単語ベクトル W_{IN} と W_{OUT} があれば可能なため，生成コストがきわめて安価であるという利点がある．よって，既存の分散表現で容易に利用することが可能であり，その波及効果は高いと考えられる．

3. 単語ベクトルの統合手法

本章では，W_{IN} と W_{OUT} を統合する手法について紹介する．

3.1 加算型単語ベクトル

加算型単語ベクトルは，式 (3) を用いて W_{ADD} を生成する．

$$\overrightarrow{W_{ADD\_word}} = \overrightarrow{word_{IN}} + \overrightarrow{word_{OUT}} \tag{3}$$

W_{ADD_word} は，新しく生成される単語ベクトルを表している．W_{IN} と W_{OUT} を加算することで，両者のとらえているそれぞれの特徴を含有したベクトルが生成され，アンサンブル学習に似た効果が期待できる．

3.2 連結型単語ベクトル

複数の分散表現を結合する手法の 1 つに，Yin らが提唱した連結型単語ベクトル [19] がある．具体的には，複数のコーパスを用いて独立に分散表現を獲得し，ベクトルの次元を拡張してそれぞれの分散表現を連結する．たとえば，次元数が 100，50，300 の 3 種の分散表現を生成したとき，次元数 $k = 100 + 50 + 300 = 450$ となる分散表現を新たに生成する．これにより，単語ベクトルの表現力の拡張と単語のカバレッジの向上が期待できると報告している．本論文では，この手法を単一のコーパスから生成される W_{IN} と W_{OUT} に適用し，W_{CONC} を生成する．この手法においても，W_{IN} と W_{OUT} のとらえているそれぞれの特徴を含有したベクトルが生成されると考えられたため，同時に検討を行う．

4. 実験

本章では，文献 [6] で紹介されている手法を用いて，単語ベクトルの性能評価を行う．個々の単語ベクトルの性能比較を行うことで，その関係性を明らかにし，実タスクに応用した際の結果の考察に利用する．

4.1 アナロジータスク

アナロジータスクとは，単語ベクトルがどの程度単語間の意味関係をとらえられているかを評価する方法である．本実験では，マイクロソフト社が公開している MSR Word Relatedness Test Set [*1] と Google Analogy Test Set [*2] を用いた．**表 1** は MSR Word Relatedness Test Set の詳細である．Category は単語の品詞，Relation は単語間の関係性，Patterns Tested は単語に付与されたタグ，#Questions はテストセット数，Example はそれに含まれる一例を示している．これより，このテストセットには，文法的な意味関係を反映した単語セットが計 8,000 セット含まれている．

一方の Google Analogy Test Set は，文法的な関係（10,675 セット）と意味的な関係（8,869 セット）を含有したデータセットであり，計 19,544 セット含まれている．

アナロジータスクは，単純な 2 単語間の単語類似性を測る方法より，頑強であると考えられる．たとえば，「dog」に対して類似する単語をあげると，「cat」や「dogs」など意味的な関係と文法的な関係が混合する恐れがある．それに対してアナロジータスクでは，「dog:dogs cat:?」に対する回答は「cats」であり，「dog:bark cat:?」に対する回答は「meow」とほぼ一意に決定される．

4.2 評価方法

(1) $a:b$，$c:d$ という関係性の単語セットであるとき，d を未知データとする．
(2) 各々の分散表現 W 内の $\vec{a}$, $\vec{b}$, $\vec{c}$ を利用し，$\vec{y} = \vec{b} - \vec{a} + \vec{c}$ を算出する．

[*1] https://www.microsoft.com/en-us/research/project/recurrent-neural-networks-for-language-processing/
[*2] https://aclweb.org/aclwiki/Analogy_(State_of_the_art)

表 1　MSR Word Relatedness Test Set.
Table 1　MSR Word Relatedness Test Set.

Category	Relation	Patterns Tested	# Questions	Example
Adjectives	Base/Comparative	JJ/JJR, JJR/JJ	1,000	good:better rough:___
Adjectives	Base/Superlative	JJ/JJS, JJS/JJ	1,000	good:best rough:___
Adjectives	Comparative/Superlative	JJR/JJS, JJS/JJR	1,000	better:best rougher:___
Nouns	Singular/Plural	NN/NNS, NNS/NN	1,000	year:years law:___
Nouns	Non-possessive/Possessive	NN/NN_POS, NN_POS/NN	1,000	city:city's bank:___
Verbs	Base/Past	VB/VBD, VBD/VB	1,000	see:saw return:___
Verbs	Base/3rd Person Singular Present	VB/VBZ, VBZ/VB	1,000	see:sees return:___
Verbs	Past/3rd Person Singular Present	VBD/VBZ, VBZ/VBD	1,000	saw:sees returned:___

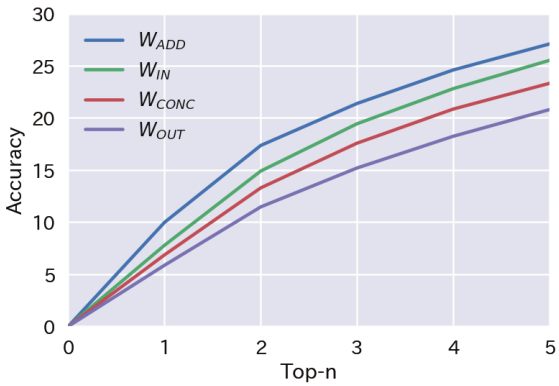

図 2　単語ベクトルの性能比較（MSR）
Fig. 2　Performance comparison of distributed representations (MSR).

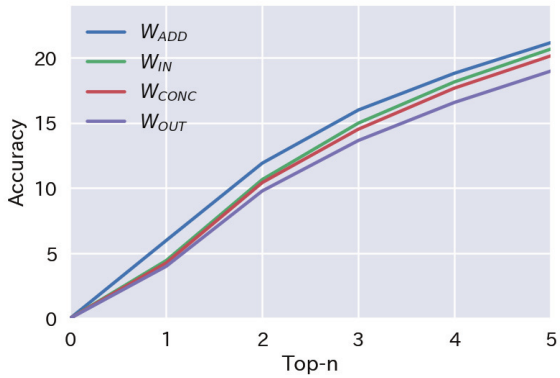

図 3　単語ベクトルの性能比較（Google）
Fig. 3　Performance comparison of distributed representations (Google).

(3) $\vec{y}$ と $\vec{d}$ の Cos 類似度と他単語ベクトルとの Cos 類似度を比較することで，単語ベクトルの性能を測定する．

また，本実験では，単語ベクトルが存在しない単語が含まれている単語セットの場合は，除外して評価を行う．

4.3　Word2Vec 学習条件

Word2Vec の学習には，英語 Wikipedia を用いた．また，各種パラメータは，$window = 5$, $size = 300$, $negative = 5$ とした．「*window*」は，前後何単語を教師データとするかを指定するオプション，「*size*」は，学習する単語ベクトルの次元数を指定するオプション，「*negative*」は，Negative Sampling のサンプリング数を指定するオプションである．Word2Vec の実装は Python の gensim ライブラリを使用し，また，W_{OUT} はライブラリ内の syn1neg に保存されているものを用いた．

4.4　実験結果

図 2，**図 3** に，それぞれのデータセットにおけるアナロジータスクの結果を示す．今回，評価できない単語セットを除いたところ，評価可能な単語セット数は，MSR では 6,820 セット，Google では 18,682 セットとなった．また，$\vec{y}$ の算出に利用した単語（$\vec{a}$, $\vec{b}$, $\vec{c}$ に相当するもの）は比較対象から除外している．Top-n は，上位何単語まで許容するかを表すパラメータである．よって，Accuracy は，Top-n 単語内に正解単語が出現する割合を表している．単語ベクトルは，L^2 ノルムが 1 となるように正規化している．また，Word2Vec は内部にランダム性を保持しているため，実験では 5 試行平均での結果を表示している．

図 2，図 3 より，精度が $W_{OUT} < W_{CONC} < W_{IN} < W_{ADD}$ という関係になっていることが確認できる．$W_{OUT} < W_{IN}$ の関係は，Press ら [20] において示唆されており，Word2Vec で標準的に W_{IN} を利用している要因の 1 つといえる．また，W_{ADD} は従来の W_{IN} よりも優位であることが確認できた．この結果から，分散表現の意味関係性能の向上により，他のタスクに応用した場合においても，W_{ADD} が精度の向上に寄与する可能性が高いと考えられる．本論文では，5 章において，実際に文書分類タスクに適用した場合の評価を行う．

また，W_{CONC} は，W_{IN} と W_{OUT} の中間を推移しており，従来の W_{IN} よりも精度が低いことが確認できる．これは，W_{CONC} が W_{IN}，W_{OUT} ごとの意味関係のとらえ方の違いを平均化したベクトルになっていると考えられ，W_{ADD} のような性能向上は期待できないことが分かった．

参考までに，各パラメータを変更した際の精度を**表 2** に示す．なお，ここでは，MSR において Top-n = 1 のときの，1 試行分の結果を表示している．また，w は *window*,

表 2　各パラメータにおけるアナロジータスク（MSR, Top-n = 1）
Table 2　Analogy task in each parameter (MSR, Top-n = 1).

each parameter(w, n, s)	$(5, 5, 300)$	$(10, 5, 300)$	$(15, 5, 300)$	$(5, 10, 300)$	$(5, 15, 300)$	$(5, 5, 100)$	$(5, 5, 200)$
W_{IN}[%]	7.79	7.87	8.03	7.52	7.60	4.62	7.01
W_{OUT}[%]	5.85	5.67	5.72	6.11	6.48	3.24	5.28
W_{CONC}[%]	6.84	6.92	6.89	6.73	7.10	3.96	6.04
W_{ADD}[%]	9.85	9.99	9.91	9.43	9.25	7.16	9.55

表 3　各単語ベクトルにおける Cos 類似度平均
Table 3　Cosine similarity in each word vector.

単語ベクトル	W_{IN}	W_{OUT}	W_{CONC}	W_{ADD}
Cos 類似度	0.16	0.21	0.19	**0.052**

表 4　内積上位単語ペアの単語一致率
Table 4　Word coincidence rate of inner product of word pair.

Top-n	Match rate [%]
1	91.2
2	94.2
3	95.1

n は *negative*，s は *size* のパラメータを表している．表 2 より，すべてのパラメータにおいて，精度は上述の関係（$W_{OUT} < W_{CONC} < W_{IN} < W_{ADD}$）を維持していることが確認できる．分散表現ごとにみると，W_{IN} は *window* に，W_{OUT} と W_{ADD} は *negative* に依存が確認できる．また，*size* に対しては，どの分散表現においても比例関係が確認できる．

4.5　考察

W_{ADD} の精度向上の結果を考察するために，個々のベクトル空間での単語ベクトル間の角度に注目した解釈を行った．まず，意味関係性能の高い単語ベクトルほど，他単語との区別が明瞭であると考えることができる．これは，ベクトル空間上での単語の広がり具合を測ること，すなわち，単語ベクトル間の角度の大きさを測ることで解釈可能であると考えられる．そこで，アナロジータスク（MSR）で用いた 930 単語を対象とし，それぞれの単語ベクトルで総当りによる単語ベクトル間の Cos 類似度を算出し，その平均値を求めた．**表 3** に，結果を示す．Cos 類似度の平均値は，より小さいほど，単語ベクトル間の角度が大きいことを意味しており，ベクトル空間を大きく利用している単語ベクトルであると考えられる．表 3 より，各値の大小関係とアナロジータスクの結果の大小関係が一致していることが確認できる．これにより，単語ベクトルの意味関係性能と単語ベクトル間角度の大きさには密接な関係があることが示唆される．また，W_{ADD} についても，単語ベクトルどうしの意味的な区別がしやすくなったことが，意味関係性能が向上した要因の 1 つとなったと考えられる．

次に，W_{ADD} の定義式 (3) より，意味演算を数式展開し，

表 5　Top-n = 1 における W_{IN} と W_{OUT} の平均内積値
Table 5　Average product between W_{IN} & W_{OUT} in Top-n = 1.

Rank	Cos_sim
1	0.32
2	0.17
3	0.13

表 6　分類対象データセット
Table 6　Dataset for classification.

データセット	文書総数	クラス数	平均単語数
livedoor	7,367	9	587.4
Reuters 21578	7,674	8	102.4

W_{ADD} の解釈を行った．今，単語ベクトルの大きさは 1 としているため，式 (4) は，W_{ADD} を用いた際の意味演算出力 $\vec{y}$ と正解単語 $\vec{d}$ の Cos 類似度を表している．

$$Cos\_sim = \overrightarrow{y_{ADD}} \cdot \overrightarrow{d_{ADD}} \tag{4}$$

$$= (\overrightarrow{y_{IN}} + \overrightarrow{y_{OUT}}) \cdot (\overrightarrow{d_{IN}} + \overrightarrow{d_{OUT}}) \tag{5}$$

$$= (\overrightarrow{y_{IN}} \cdot \overrightarrow{d_{IN}} + \overrightarrow{y_{OUT}} \cdot \overrightarrow{d_{OUT}}) + \overrightarrow{y_{IN}} \cdot \overrightarrow{d_{OUT}} + \overrightarrow{y_{OUT}} \cdot \overrightarrow{d_{IN}} \tag{6}$$

これより，式 (6) の第 1 項は，W_{IN} と W_{OUT} それぞれでアナロジータスクを行った結果を平均化する項であるととらえることができる．つまり，第 1 項のみを考慮した場合，W_{ADD} が W_{IN} の性能を上回ることは不可能であるため，残りの項が W_{ADD} の性能の高さに寄与しているといえる．加えて，第 2 項，第 3 項は，意味演算出力 $\vec{y}$ と正解単語ベクトルとの共起度合いを表していることが分かる．また，Mitra ら [21] により，W_{IN} と W_{OUT} の内積値を算出した際の最上位ペアは同一単語である（$\overrightarrow{dog_{IN}}$ と $\overrightarrow{cat_{OUT}}$ よりも，$\overrightarrow{dog_{IN}}$ と $\overrightarrow{dog_{OUT}}$ の内積の値が大きい）傾向が確認されている．そこで，本実験で用いた分散表現に対して，同様の確認を行った．その結果を**表 4** に示す．表 4 は，W_{IN} と W_{OUT} の内積上位単語ペアにおける同一単語となった単語ペアの割合を示している．Top-n は上位何単語まで許容するかを表すパラメータである．

表 4 より，実際に W_{IN} と W_{OUT} の内積値を算出した際の最上位ペアは，同一単語であることが多いことが確認できる．また，**表 5** は，表 4 において，Top-n = 1 の状態における W_{IN} と W_{OUT} の平均内積値を表している．Rank

表 7　各データセットにおける文書分類精度
Table 7　Document classification accuracy in each dataset.

データセット（学習コーパス）	livedoor（Wiki）	livedoor（学習データ）	Reuters（Wiki）	Reuters（学習データ）
W_{IN}（SVM）[%]	87.11(±0.091)	91.41(±0.077)	94.39(±0.062)	96.20(±0.095)
W_{OUT}（SVM）[%]	87.11(±0.10)	91.48(±0.035)	94.25(±0.058)	96.10(±0.057)
W_{CONC}（SVM）[%]	**87.38**(±0.14)	91.78(±0.072)	94.45(±0.025)	96.20(±0.089)
W_{ADD}（SVM）[%]	87.36(±0.099)	**92.78**(±0.074)	**94.98**(±0.060)	**96.64**(±0.053)
W_{IN}（CNN）[%]	—	—	—	95.83(±0.12)
W_{ADD}（CNN）[%]	—	—	—	96.05(±0.14)

は内積値の降順インデックスである．よって，Rank 1 は同一単語どうしの内積値の平均値を表している．表 5 より，同一単語どうしの内積値は，他と比べて突出していることが分かる．この観測的事実を考慮すると，式 (6) の第 2 項，第 3 項は，全体の結果に補正を加えるバイアス項であるととらえることができる．つまり，W_{IN}，または W_{OUT} において，$\vec{y}$ が正解単語ベクトルに類似した結果である場合，式 (6) の第 2 項，または第 3 項により極端な重みがかかるため，W_{ADD} は W_{IN} と W_{OUT} それぞれの分散表現の強みを利用することで，精度の向上を達成していると考えられる．

5. 加算ベクトル W_{ADD} の実タスクへの応用

本章では，4 章で示した各々の分散表現を実タスクへ応用した際の，精度への貢献を比較・検証する．従来研究 [13] では，分散表現自体の精度比較に主眼が置かれており，実際のタスクに近いモデルへ応用した際の精度比較は行われていない．そこで，文書分類タスクにおける精度の比較を行うことで，W_{ADD} の有効性を検証し，また，分散表現の意味関係性能との関連性について考察する．

5.1　文書分類

文書分類とは，与えられた文書をあらかじめ定められたクラスのいずれかに分類することである．これは，スパムメール分類や Web 記事分類などに広く実用化されている．分散表現の応用例として，分散表現を文書の素性とした文書分類手法が報告されている．分類器に SVM を用いた手法では，文書内出現単語の単語ベクトルを用い，加算平均ベクトルで文書ベクトルを定義し，分類器の特徴量として利用している [22], [23]．また Kim [24] は，文書を単語ベクトルの 2 次元マトリクスととらえることで，CNN の適用を可能とした分類手法を提案している．本実験では，これらの方法を用いて SVM と CNN の 2 種を実装した．

5.2　分類対象データセット

4 章では，データセットの都合上，英語コーパスのみでの実験を行った．本実験では，日本語と英語のデータセットを利用することで，言語の違いに対する W_{ADD} の適用可能性を検討する．実験で使用した分類対象データセットを**表 6** に示す．livedoor ニュースコーパスは日本語のテキストデータとなっており，ウェブサイト[*3]からダウンロードして利用できる．また，Reuters 21578 は英語のテキストデータとなっており，文献 [25] の著者のウェブサイト[*4]からダウンロードして利用できる．

5.3　Word2Vec 学習コーパス

Word2Vec の学習コーパスには，日本語 Wikipedia と英語 Wikipedia，さらに，各データセットの学習データを用いた．これにより，コーパスの違いによる精度に対する貢献を確認する．

5.4　分類器

分類器には SVM と CNN を用いた．SVM は RBF カーネルを用い，ハイパーパラメータ C と γ はグリッドサーチで決定した．また，特徴量とする文書ベクトルは，文書内出現単語の加算平均ベクトルを利用し，L^2 ノルムが 1 になるように正規化した．また，CNN を用いる場合，文書ごとに出現する単語数が異なるため，出現単語数が最大の文書に合わせ，0 パディングすることで文書長を揃えた．また，畳み込み層では 1-gram，2-gram，3-gram に対応するフィルタを用い，結果をプーリング処理した．

5.5　実験結果

5.5.1　アナロジータスクと文書分類との関連性

表 7 に，各データセットと，Word2Vec 学習コーパス別に，個々の単語ベクトルを用いて文書分類を行った際の分類精度を示す．本実験では，5 分割交差検証を行った．また，Word2Vec の初期値ランダム性を考慮し，学習コーパスが Wikipedia の場合は 5 試行，学習データの場合は 10 試行行い，それぞれの平均値と標準偏差を表示している．なお，学習コーパスが英語の Wikipedia の場合の分散表現は，4 章で生成した分散表現をそのまま利用している．諸条件は 4.3 節と同様で，$window = 5$，$size = 300$，$negative = 5$ とした．ただし，CNN を用いた分類では，マシン性能の都合上，$size = 100$ とした．また，分類精度は，テスト文書に対して正しく分類された割合を表している．なお，CNN を用いた実験では，マシン性能の都合上，割愛

[*3] https://www.rondhuit.com/download.html
[*4] http://web.ist.utl.pt/acardoso/datasets/

表 8　各パラメータにおける分類精度（livedoor, $size = 100$）
Table 8　Classification accuracy in each parameter (livedoor, $size = 100$).

livedoor	$w=5,n=5$	$w=10,n=5$	$w=15,n=5$	$w=5,n=10$	$w=10,n=10$	$w=15,n=10$	$w=5,n=15$	$w=10,n=15$	$w=15,n=15$
W_{IN}[%]	90.48(±0.082)	91.39(±0.16)	91.93(±0.10)	89.95(±0.11)	90.93(±0.092)	91.43(±0.097)	89.63(±0.10)	90.62(±0.33)	91.18(±0.074)
W_{OUT}[%]	90.61(±0.073)	91.15(±0.11)	91.46(±0.076)	90.14(±0.13)	90.63(±0.14)	90.98(±0.074)	89.85(±0.13)	90.20(±0.26))	90.51(±0.085)
W_{CONC}[%]	91.01(±0.096)	91.70(±0.12)	92.07(±0.074)	90.66(±0.063)	91.19(±0.12)	91.62(±0.097)	90.45(±0.11)	90.92(±0.31)	91.40(±0.089)
W_{ADD}[%]	92.18(±0.087)	92.79(±0.10)	**93.11**(±0.11)	92.14(±0.052)	92.63(±0.11)	92.97(±0.10)	92.02(±0.065)	92.55(±0.21)	92.91(±0.065)

表 9　各パラメータにおける分類精度（livedoor, $size = 200$）
Table 9　Classification accuracy in each parameter (livedoor, $size = 200$).

livedoor	$w=5,n=5$	$w=10,n=5$	$w=15,n=5$	$w=5,n=10$	$w=10,n=10$	$w=15,n=10$	$w=5,n=15$	$w=10,n=15$	$w=15,n=15$
W_{IN}[%]	91.15(±0.097)	92.01(±0.092)	92.34(±0.11)	90.74(±0.12)	91.61(±0.12)	91.95(±0.095)	90.39(±0.10)	91.32(±0.098)	91.72(±0.14)
W_{OUT}[%]	91.16(±0.099)	91.55(±0.11)	91.76(±0.077)	90.78(±0.097)	91.13(±0.092)	91.40(±0.11)	90.51(±0.13)	90.86(±0.067))	91.05(±0.090)
W_{CONC}[%]	91.55(±0.077)	92.12(±0.10)	92.43(±0.054)	91.28(±0.056)	91.75(±0.070)	92.02(±0.099)	91.02(±0.072)	91.50(±0.073)	91.74(±0.078)
W_{ADD}[%]	92.60(±0.078)	93.13(±0.055)	**93.38**(±0.077)	92.58(±0.054)	93.08(±0.066)	**93.38**(±0.073)	92.56(±0.058)	93.09(±0.095)	93.34(±0.093)

表 10　各パラメータにおける分類精度（livedoor, $size = 300$）
Table 10　Classification accuracy in each parameter (livedoor, $size = 300$).

livedoor	$w=5,n=5$	$w=10,n=5$	$w=15,n=5$	$w=5,n=10$	$w=10,n=10$	$w=15,n=10$	$w=5,n=15$	$w=10,n=15$	$w=15,n=15$
W_{IN}[%]	91.38(±0.094)	92.32(±0.066)	92.65(±0.069)	91.11(±0.12)	91.87(±0.062)	92.19(±0.093)	90.77(±0.080)	91.72(±0.12)	92.00(±0.077)
W_{OUT}[%]	91.39(±0.081)	91.80(±0.079)	91.96(±0.082)	91.10(±0.073)	91.41(±0.086)	91.70(±0.083)	90.88(±0.10)	91.20(±0.086))	91.39(±0.093)
W_{CONC}[%]	91.81(±0.056)	92.30(±0.061)	92.67(±0 .069)	91.50(±0.086)	92.00(±0.078)	92.24(±0.088)	91.29(±0.090)	91.77(±0.10)	92.00(±0.059)
W_{ADD}[%]	92.79(±0.054)	93.25(±0.070)	**93.53**(±0.062)	**93.53**(±0.093)	93.30(±0.042)	93.51(±0.042)	92.72(±0.085)	93.22(±0.092)	93.46(±0.062)

した実験が存在するため，それらは空欄で表示している．

表 7 より，言語の違いに依存せず，W_{ADD} を用いた場合の精度が，従来の W_{IN} よりも高いことが確認できる．これら 2 種の分散表現に対して，ステューデントの t 検定（多重比較を考慮し，$\alpha = 0.05/4 = 0.0125$）を行ったところ，各データセットにおいて統計的有意差が確認された（livedoor（Wiki），livedoor（学習データ），Reuters（Wiki），Reuters（学習データ）について，それぞれ $p = 3.18 * 10^{-3}$，$2.57 * 10^{-11}$，$2.22 * 10^{-5}$，$3.30 * 10^{-11}$）．この結果より，アナロジータスクにおいて，意味関係性能の高い W_{ADD} を実タスクに適用することで，精度の向上が期待できるといえる．また，単語ベクトルの意味関係性能が高いことは，文書ベクトルを生成する際に文書をより的確に特徴づけることが可能であると考えられ，実際に精度の向上に寄与したと考えられる．さらに，Reuters（Wiki）の結果において，精度が $W_{OUT} < W_{IN} < W_{ADD}$ となっており，アナロジータスクにおける単語ベクトルの意味関係性能と一致していることが分かる．

Word2Vec の学習コーパスの違いに注目すると，学習コーパスを Wikipedia から学習データ（ニュース記事）にすることで，全体的に精度が向上していることが分かる．これは，分類対象のデータに学習コーパスをフィッティングすることで，文書特有の単語や表現を学習した分散表現を獲得することが可能となり，精度の向上を達成したと考えられる．よって，分類用のデータ以外にも，ニュース記事などのコーパスを Word2Vec の学習に利用することで，さらなる分類精度の向上が期待できる．また，分類器を CNN に変更した場合においても，従来の W_{IN} を用いた場合と比較して，W_{ADD} を分類器に用いることで，分類精度の向上が確認できる．これより，分類器の違いに対しても W_{ADD} が有効に働くことが示唆される．ただ，精度の面では SVM に劣っていることが分かる．原因として，次元数の違いが考えられる．ただしこれについては，パラメータなどのチューニングをすることで，さらなる精度の向上が見込めると思われる．

5.5.2　各パラメータにおける性能比較

次に，各パラメータの違いによる分類精度への貢献を検討する．**表 8**，**表 9**，**表 10** は，livedoor ニュースコーパスに対して，パラメータを変化させた際の分類精度であり，**表 11**，**表 12**，**表 13** は，Reuters 21578 に対して，パラメータを変化させた際の分類精度である．Word2Vec の学習には，分類用データセットの学習データを利用している．

まず，各分散表現の違いに注目する．W_{ADD} はその他の分散表現に対して，パラメータの変化によらず，つねに最も高い分類精度となっていることが確認できる．これより，W_{ADD} は文書分類タスクにおける利用価値が高いといえる．これは，4.4 節の結果と一致しており，分散表現の意味関係性能の向上により，他のタスクに応用した場合においても，W_{ADD} が精度の向上に寄与することが示された．また，W_{OUT} は W_{IN} に対して，分類精度が低い傾向が読み取れる．これは，4.4 節の知見から，意味関係性能の差に起因するものであると考えられる．加えて，W_{CONC} は W_{IN}

表 11　各パラメータにおける分類精度（Reuters, $size = 100$）
Table 11　Classification accuracy in each parameter (Rueters, $size = 100$).

Reuters	$w=5,n=5$	$w=10,n=5$	$w=15,n=5$	$w=5,n=10$	$w=10,n=10$	$w=15,n=10$	$w=5,n=15$	$w=10,n=15$	$w=15,n=15$
W_{IN}[%]	95.97(±0.061)	96.02(±0.071)	96.06(±0.060)	95.83(±0.055)	96.00(±0.080)	96.07(±0.13)	95.73(±0.054)	95.98(±0.10)	95.99(±0.15)
W_{OUT}[%]	95.88(±0.048)	95.88(±0.055)	95.68(±0.050)	95.68(±0.11)	95.70(±0.052)	95.72(±0.11)	95.59(±0.099)	95.60(±0.057)	95.60(±0.13)
W_{CONC}[%]	95.94(±0.047)	95.96(±0.043)	95.76(±0.053)	95.76(±0.096)	95.86(±0.085)	95.89(±0.079)	95.68(±0.079)	95.75(±0.077)	95.75(±0.15)
W_{ADD}[%]	96.48(±0.069)	96.69(±0.074)	96.54(±0.080)	96.54(±0.073)	96.75(±0.069)	**96.83**(±0.11)	96.52(±0.068)	96.77(±0.065)	96.81(±0.11)

表 12　各パラメータにおける分類精度（Reuters, $size = 200$）
Table 12　Classification accuracy in each parameter (Rueters, $size = 200$).

Reuters	$w=5,n=5$	$w=10,n=5$	$w=15,n=5$	$w=5,n=10$	$w=10,n=10$	$w=15,n=10$	$w=5,n=15$	$w=10,n=15$	$w=15,n=15$
W_{IN}[%]	96.15(±0.068)	96.16(±0.081)	96.16(±0.054)	95.99(±0.059)	96.02(±0.046)	96.05(±0.028)	95.93(±0.045)	95.99(±0.050)	96.00(±0.032)
W_{OUT}[%]	95.98(±0.090)	95.99(±0.085)	95.93(±0.040)	95.78(±0.060)	95.70(±0.033)	95.67(±0.045)	95.69(±0.038)	95.59(±0.052)	95.55(±0.034)
W_{CONC}[%]	96.13(±0.061)	96.10(±0.053)	96.09(±0.048)	95.89(±0.057)	95.87(±0.052)	95.86(±0.047)	95.82(±0.034)	95.75(±0.025)	95.73(±0.031)
W_{ADD}[%]	96.55(±0.039)	96.74(±0.044)	96.84(±0.052)	96.60(±0.023)	96.73(±0.056)	**96.87**(±0.039)	96.58(±0.057)	96.77(±0.039)	96.84(±0.037)

表 13　各パラメータにおける分類精度（Reuters, $size = 300$）
Table 13　Classification accuracy in each parameter (Rueters, $size = 300$).

Reuters	$w=5,n=5$	$w=10,n=5$	$w=15,n=5$	$w=5,n=10$	$w=10,n=10$	$w=15,n=10$	$w=5,n=15$	$w=10,n=15$	$w=15,n=15$
W_{IN}[%]	96.20(±0.095)	96.30(±0.11)	96.39(±0.075)	96.09(±0.084)	96.27(±0.098)	96.35(±0.089)	96.04(±0.057)	96.09(±0.049)	96.33(±0.13)
W_{OUT}[%]	96.10(±0.057)	96.18(±0.10)	96.25(±0.11)	95.92(±0.091)	96.01(±0.096)	95.97(±0.10)	95.79(±0.062)	95.65(±0.046)	95.89(±0.16)
W_{CONC}[%]	96.20(±0.089)	96.23(±0.090)	96.31(±0.062)	95.98(±0.077)	96.15(±0.088)	96.21(±0.086)	95.92(±0.054)	95.85(±0.042)	96.09(±0.15)
W_{ADD}[%]	96.64(±0.053)	96.89(±0.12)	97.06(±0.11)	96.67(±0.064)	96.98(±0.087)	**97.08**(±0.080)	96.68(±0.089)	96.82(±0.032)	**97.08**(±0.13)

に対して，日本語の livedoor では分類精度が高く，英語の Reuters では分類精度が低い傾向が確認できる．W_{CONC} は，次元数が他の分散表現の 2 倍ある点に優位性があるが，意味関係性能の点では劣っている．よって，分類対象のデータの特徴により，精度の優劣が変化するものだと考えられる．

次に，パラメータの違いに注目する．*window* と *size* は，大きくなるほど，分類精度が向上している傾向が確認できる．*window* は大きくなるほど，Word2Vec 学習時に，ターゲットの単語に対する正解単語の範囲が広がるため，共起情報により単語の意味を学習するのに役立つと考えられる．また，*size* は大きくなるほど，単語の表現能力が高まるため，文書ベクトルもより詳細に表現することができ，分類精度の向上に寄与していると考えられる．しかし，表 2 では，必ずしも上述の傾向があることが確認できない．これは，学習コーパスの違いや試行回数の少なさに起因している可能性があると考えられる．また，W_{IN}，W_{OUT}，W_{CONC} では，*negative* が大きくなるほど，分類精度が低下している傾向が確認できる．一方で，W_{ADD} では，*negative* の影響をほとんど受けていないことが分かる．これは，W_{ADD} では，*negative* のパラメータを考慮する必要性がないことを意味し，実利用においてメリットとなりうるだろう．

6. まとめ

本論文では，Word2Vec における，加算型単語ベクトル W_{ADD} に焦点を当て，アナロジータスクを用いて意味関係性能の比較を行い，その原因解析を観測的事実を用いて行った．また，従来では検証されていなかった実タスクにおける W_{ADD} の有効性を示すため，文書分類タスクを用いて精度の評価を行い，結果について考察した．実験より，W_{ADD} は意味関係性能の向上と文書分類精度の向上に貢献することが確認された．これにより，意味関係性能と文書分類精度との強い関連性が確認でき，文書分類タスク以外のアプリケーションにおいても，W_{ADD} が有効に働く可能性は高いと考えられる．ただし，パラメータの組合せに対する意味関係性能の高さの違いの関係が，必ずしもそのまま文書分類タスクにおいての精度の高さの違いに当てはまっていない場合が一部存在することが確認できた．特に，W_{ADD} については，*negative* の影響を受けにくい傾向があることから，パラメータチューニング対象からの除外が可能であると考えられる．

W_{ADD} の生成には W_{IN} と W_{OUT} が必要であるが，それらは 1 度の学習で同時に生成されるため，W_{ADD} の生成コストはきわめて安価である．つまり，既存の分散表現で容易に利用することが可能でありながら，精度の向上が期待できる優れた手法であるといえる．単語のベクトル化手法は，自然言語処理分野の根本を支える技術であり，今後ますますの発展が期待される．

参考文献

[1] Golson, S.: One-hot state machine design for FPGAs, *3rd PLD Design Conference*, pp.1–6 (1993).

[2] Sivic, J. and Zisserman, A.: Efficient Visual Search of Videos Cast as Text Retrieval, *IEEE Trans. Pattern Analysis and Machine Intelligence* (*TPAMI*), Vol.31, No.4, pp.591–606 (2009).
[3] Deerwester, S., Dumais, S., Furnas, G., Landauer, T. and Harshman, R.: Indexing by Latent Semantic Analysis, *Journal of the American Society for Information Science*, Vol.41, No.6, pp.391–407 (1990).
[4] Blei, D., Ng, A. and Jordan, M.: Latent Dirichlet Allocation, *The Journal of Machine Learning Research*, Vol.3, pp.993–1022 (2014).
[5] Maas, A. and Ng, A.: A Probabilistic Model for Semantic Word Vectors, *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning* (2010).
[6] Mikolov, T., Yih, W. and Zweig, G.: Linguistic Regularities in Continuous Space Word Representations, *NAACL HLT* (2013).
[7] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, *Proc. ICLR Workshops Track* (2013).
[8] Xue, B., Fu, C. and Shaobin, Z.: A Study on Sentiment Computing and Classification of Sina Weibo with Word2vec, *Proc. IEEE International Congress on the Big Data* (*BigData Congress*), pp.358–363, IEEE (2014).
[9] Ma, L. and Zhang, Y.: Using Word2Vec to process big text data, *2015 IEEE International Conference on Big Data* (*Big Data*), pp.2895–2897 (2015).
[10] Ju, R., Zhou, P., Li, C. and Liu, L.: An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis, *Proc. IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp.2276–2283, IEEE (2015).
[11] Sien, S.: Adapting word2vec to Named Entity Recognition, *Proc. Nordic Conference of Computational Linguistics* (2015).
[12] Pennington, J., Socher, R. and Manning, C.: GloVe: Global Vectors for Word Representation, *Proc. Empiricial Methods in Natural Language Processing* (*EMNLP 2014*), No.12, pp.1532–1543 (2014).
[13] Levy, O., Goldberg, Y. and Dagan, I.: Improving Distributional Similarity with lessons Learned from Word Embeddings, *TACL*, No.3, pp.211–225 (2015).
[14] Harris, Z.: Distributional Structure, *Word, IO*, pp.140–162 (1954).
[15] Rong, X.: Word2vec Parameter Learning Explained, arXiv preprint arXiv:1411.2738 (2014).
[16] Le, Q. and Mikolov, T.: Distributed Representations of Sentences and Documents, *Proc. ICML* (2014).
[17] Moody, C.: Mixing Dirichlet Topic Models and Word Embeddings to Make LDA2vec, arXiv preprint arXiv:1605.02019 (2016).
[18] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L.: Deep Contextualized Word Representations, *Proc. NAACL* (2018).
[19] Yin, W. and Schutze, H.: Learning Word Meta-embeddings, *Proc. 54th Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), *Berlin, Germany*, pp.1351–1360, Association for Computational Linguistics (2016).
[20] Press, O. and Wolf, L.: Using the Output Embedding to Improve Language Models, *Proc. 15th Conference of the European Chapter of the Association for Computational Linguistics, Volume 2, Short Papers, Valencia, Spain*, pp.157–163 (2017).
[21] Mitra, B., Nalisnick, E., Craswell, N. and Caruana, R.: A Dual Embedding Space Model for Document Ranking, CoRR abs/1602.01137 (2016).
[22] Liu, R., Wang, D. and Xing, C.: Document Classification Based on Word Vectors, *ISCSLP '14* (2014).
[23] Xing, C., Wang, D. and Zhang, X.: Document Classification with Distributions of Word Vectors, *Signal and Information Processing Association Annual Summit and Conference* (*APSIPA*), pp.1–5 (2014).
[24] Kim, Y.: Convolutional Neural Networks for Sentence Classification, *Proc. EMNLP* (2015).
[25] Cardoso-Cachopo, A.: *Improving Methods for Single-label Text Categorization*, PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa (2007).

内田 脩斗　（学生会員）

2018 年 3 月名古屋大学工学部電気電子・情報工学科卒業．同年 4 月同大学大学院工学研究科博士課程前期課程情報・通信工学専攻に入学，現在に至る．主として自然言語処理に関する研究に従事．人工知能学会，IEEE 各会員．

吉川 大弘　（正会員）

1997 年名古屋大学大学院博士課程修了．同年カリフォルニア大学バークレー校ソフトコンピューティング研究所客員研究員．1998 年三重大学工学部助手．2005 年名古屋大学大学院工学研究科 COE 特任准教授．2006 年 10 月同研究科准教授，現在に至る，主としてソフトコンピューティングとその応用に関する研究に従事．博士（工学）．IEEE，人工知能学会，日本知能情報ファジィ学会，進化計算学会各会員．

古橋 武

1985 年名古屋大学大学院工学研究科博士後期課程電気系専攻修了．工学博士．2004 年名古屋大学大学院工学研究科計算理工学専攻教授，現在に至る．ソフトコンピューティング，感性工学に関する研究に従事．1996 年日本ファジィ学会論文賞受賞．IEEE，日本知能情報ファジィ学会，電気学会等の各会員．