Word Embeddingモデル再訪

堅山　耀太郎

近年，特に深層学習を利用した統計的自然言語処理の発展により，自然言語処理技術を利用したアプリケーションはわれわれの身近なものとなりつつある．自然言語処理は幅広いトピックを含み，それぞれの分野において課されるタスクは大きく異なるが，単語の Embedding（埋め込み）はさまざまなタスクで共通して用いられる自然言語の標準的な特徴量となっており広く使われている．本稿では広く知られている Word2Vec 以降の Embedding アルゴリズムの進歩にも触れつつ，実用上の利用方法を紹介する．

キーワード：自然言語処理，Word Embedding，分散表現，Word2Vec

1. 自然言語処理における Embedding

1.1 自然言語処理の活用の高まり

近年，本邦においては人材採用の難化や業務の効率化の必要性などから企業における「人工知能」（機械学習）の活用の機運が高まっている．そのため，社内業務から顧客接点まで幅広く，自然言語処理を用いたソリューションを多くの企業が採用し始めている．筆者が所属する株式会社 BEDORE では，深層学習技術と自然言語処理技術をコアとして，チャット上での日本語の自動応答などのプロダクトを企業に提供している．

自然言語処理に含まれるトピックは多岐にわたり，さまざまなタスクが存在している．応用を主眼においた場合の一例として，クエリ文から関連性の高い文書を提示する情報検索タスク，文書を規定のクラスに割り振る文書分類タスク，人と会話を行う対話生成タスクなどが挙げられる．近年，統計的機械学習技術の進歩とともに，自然言語処理においてもデータドリブンな統計的自然言語処理が主流になりつつあり，このようなタスクで機械学習アルゴリズムが広く用いられ，成功を収めている．

統計的自然言語処理においては，いかに自然言語のデータを機械学習アルゴリズムに投入できる形式に変換するかということが大きな問題となる．たとえば，単語を入力としてその単語の難しさを 0 から 1 の実数で判定するタスクがあるとする．この場合，データから「単語の長さ」のようにタスクに必要なデータの性質をうまく表現する数値をいくつか抽出し，ロジスティック回帰などの機械学習アルゴリズムを用いて予測するという形がまず考えられる．ここでいう，「データの性質をうまく表現する数値」を機械学習の文脈では特徴量と呼び，これをベクトルとして表したものを特徴量ベクトルと呼ぶ．しかし，ここで抽出した「単語の長さ」という特徴量はこのタスクにおいては有効だが，単語の意味などの多くの情報を捨ててしまっているために，ほかのタスクでは利用しづらいだろう．そこで，このようにタスクごとに特殊な特徴量を設計する必要がないように，より複雑な情報を表現し，汎用的に幅広いタスクに利用可能な特徴量抽出手法が求められてきた．

そのなかでも，本稿では単語の特徴量抽出の方法として非常に広く使われている Word Embedding を中心に，特に Embedding による特徴量の抽出に関して説明する．

1.2 Embedding とはなにか？

自然言語処理において Embedding（埋め込み）とは主に，文や単語，文字など自然言語の構成要素に何らかの空間における（実）ベクトルを与えることを指す．たとえば Word Embedding では，その名のとおりそれぞれの単語に対して固有のベクトルを割り当てることとなる．

自然言語は離散的かつ可変長の要素で階層的に構成されているのが特徴的であるが，一方で先述の例のように，多くの場合機械学習アルゴリズムに投入するにはデータごとに特徴量ベクトルの長さ（次元数）が変化してはならないので，自然言語の要素の特徴量としては何らかの固定次元のベクトルを用いる必要がある．Embedding では，同じ階層の要素すべて（Word Embedding なら単語すべて）が同じ空間内に配置されるため，要素の特徴量は常に同じ長さのベクトルで表現される．このため，機械学習アルゴリズムに容易に投入することができる．

かたやま　ようたろう
株式会社 BEDORE　取締役
〒 113–0033 東京都文京区本郷 2–35–10　本郷瀬川ビル 4F
y_katayama@bedore.jp

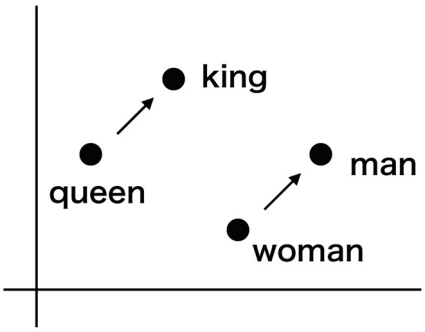

図 1　Word2Vec におけるアナロジータスク

2013 年に発表された Word2Vec [1] をご存知の読者は多いだろう．Word2Vec は Word Embedding を行う手法で，“queen” − “woman” + “man” = “king” というような，単語の特徴量ベクトル（単語ベクトル）の四則演算が人間の感覚と一致するような Embedding を実現したことで話題となった．図 1 に Word2Vec におけるこの種の計算（アナロジー）の概念図を示した．Word2Vec は実応用においても幅広く用いられている．

現在でも Word2Vec は使われているが，Embedding に関連した研究・手法は 2013 年からさまざまなアップデートがある．2 節では，Word2Vec 以降を重点的に，さまざまな Embedding のアルゴリズムとその応用について紹介する．3 節では日本語の実応用における Embedding の利用について触れる．弊社所有のデータにおける各種アルゴリズムのパフォーマンス比較などを通して，Embedding の実世界でのチューニングの例なども紹介したい．

1.3　さまざまなレベルでの Embedding

先ほど自然言語の階層性について言及したが，解きたい問題によってどのレベルの要素を埋め込むべきかが異なってくる．たとえば文を分類したいならば，単語ではなく文をベクトルとして表現し，埋め込む方法が必要だ．文や文書は単語の系列として存在しているので，単語より大きい要素の Embedding は単語または単語以下の Embedding を用いて表現することが多い．たとえば，Skip-thought Vectors [2] では系列データを扱うのに適したニューラルネットワークの一種である Recurrent Neural Network (RNN) に文を Word2Vec のベクトルの系列として入力し，文の Embedding を実現している．したがって，本稿では基礎となり，また実践的なタスクでも広く用いられる単語の Embedding に関する話題を主に取り扱う．

2.　Embedding の各種アルゴリズム

2.1　One-hot 表現

一番シンプルな Embedding の手法として，まず One-hot 表現が挙げられる．One-hot 表現ではまず表現する語彙のリストを作る．この語彙数の次元のベクトルを用意し，各単語を各次元に割り当てる．各単語を表現するベクトルは対応する次元の要素のみが非零の 1 の値をもつ基底ベクトルとなる．

One-hot 表現は離散的であり，語の意味の近さなどを表現することはできず，単語の一致・不一致しかわからない．そのため，この表現はむしろ文のベクトルを作る Bag-of-words 表現（BoW 表現）に使われる．BoW 表現では文のベクトルは単純に各単語の One-hot 表現のベクトル和として表され，シンプルな文書分類の特徴量として広く使われている．2 文で構成した One-hot 表現と BoW 表現を表 1 に示した．

One-hot 表現はさらに，未知語を扱うことができない，次元数が膨大となるという大きな欠点がある．特に次元数の増加は深層学習をはじめとする機械学習アルゴリズムでの利用に際してメモリ使用量などの観点からネックになる．

2.2　共起関係の利用

One-hot 表現と異なる，語の類似性などを反映させた連続的な単語表現の多くは単語の共起関係を利用したものである．意味が似ている単語は類似した文脈（=周囲の単語の出現分布）で出現するだろうという分散仮説 (Distributional Hypothesis) [3] が古くから存在しており，これを利用しているためである．実際に日本語で大規模な単語の共起関係を調査したデータベースが単語共起頻度データベース [4] として NICT から公開されているが，たとえば「海外旅行」との共起頻度（Dice 係数）が最大となる単語は「国内旅行」であり，分散仮説の妥当性を確認できる結果となっている．

分散仮説に関連して開発されたアルゴリズムは大きく count-based な手法と predictive な手法に分類することができる [5].

count-based な手法は，さまざまな「文脈」における単語の出現頻度を数えるところから始まる．「文脈」とは，何か意味のある単語の集合として広く捉えられ，たとえば，Wikipedia を例にとるならば，個々の記事を文脈と考えてもよいし，各記事各文中から取り出せる n 個の連続した単語（n-gram）をすべて文脈と考えてもよい．ここで，単語数を v，文脈数を c とし，文脈の中で登場する単語の頻度を表した $v \times c$ 次元の単語-

表 1　2 文のみで構築した One-hot 表現の例

対応単語	The	pen	is	mightier	than	the	sword	BoW
the	1	0	0	0	0	1	0	2
pen	0	1	0	0	0	0	0	1
be	0	0	1	0	0	0	0	1
mighty	0	0	0	1	0	0	0	1
than	0	0	0	0	1	0	0	1
sword	0	0	0	0	0	0	1	1
I	0	0	0	0	0	0	0	0
wear	0	0	0	0	0	0	0	0
my	0	0	0	0	0	0	0	0
as	0	0	0	0	0	0	0	0
others	0	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0	0
their	0	0	0	0	0	0	0	0

対応単語	I	wear	my	pen	as	others	do	their	sword	BoW
the	0	0	0	0	0	0	0	0	0	0
pen	0	0	0	1	0	0	0	0	0	1
be	0	0	0	0	0	0	0	0	0	0
mighty	0	0	0	0	0	0	0	0	0	0
than	0	0	0	0	0	0	0	0	0	0
sword	0	0	0	0	0	0	0	0	1	1
I	1	0	0	0	0	0	0	0	0	1
wear	0	1	0	0	0	0	0	0	0	1
my	0	0	1	0	0	0	0	0	0	1
as	0	0	0	0	1	0	0	0	0	1
other	0	0	0	0	0	1	0	0	0	1
do	0	0	0	0	0	0	1	0	0	1
their	0	0	0	0	0	0	0	1	0	1

文脈の共起行列 X を作ることができる．この行をそのまま c 次元の単語ベクトルとみなすこともできるが，多数の文脈を考慮に入れる場合はスパースになってしまい扱いにくく，またベクトルの次元数を制御しづらい．そこで，これをなんらかの手法で縮約してやれば，固定長の単語を表現するベクトルが得られるだろうというのが count-based な手法のアイデアである．一般に行列分解が縮約の手法として用いられる．

predictive な手法は，前後の文脈の単語のベクトルから目的の単語ベクトルを予測する（またはその逆を予測する）というタスクを通して，単語ベクトルを学習する手法である．単語ベクトルの次元数は予測のアルゴリズム内で自由に決めることができる．

これらの方法を利用すると，次元圧縮された潜在変数の空間に単語がマッピングされることとなる．したがって，One-hot 表現のように一つの要素（≒次元）が一つの概念（≒単語）に対応しているような表現ではなく，複数の要素が一つの概念を構成し，一つの要素が複数の概念を構成するために用いられる “many-to-many” の表現 となる．深層学習で有名な Hinton は，ニューラルネットワークによる概念の表現を論じた際に，前者を局所表現 (Local Representation)，後者を分散表現 (Distributed Representation) と呼んだ．正確には，Distributional Hypothesis を前提として主に count-based な手法により導かれる表現を Distributional Representation と呼び，predictive な手法のうち特にニューラルネットワークを用いて求められるものを想定した Distributed Representation とは異なる歴史的経緯があるが [6]，どちらも日本語では「分散表現」と呼称され，また本稿では両者に言及するために区別は行わない．

2.2.1　LSI

count-based な手法のうち，代表的な手法の一つである Latent Semantic Indexing (LSI) [7] をここでは説明したい．もともとは情報検索のために考えられた手法で，さまざまな文書を潜在変数空間の点として表し，関連度を検索クエリ文書との近さで計測するといった目的をもっている．共起行列 X は「文脈」として「文書」をとる．したがって，X_{ij} は単語 i が文書 j に出現する回数を表す．LSI のアイデアは，この X を特異値分解することで単語・文書のベクトルを得ることができるというものだ．

行列 X のランクを r とすると，特異値分解により，$v \times r$ 直交行列 U，$c \times r$ 直交行列 V，$r \times r$ 対角行列 Σ を用いて，

$$X = U\Sigma V^{\intercal}$$

と分解でき，得られた r 次元の行ベクトル $\boldsymbol{u}_i$ と，$\boldsymbol{v}_j$ がそれぞれそのまま単語ベクトル，文書ベクトルとなる．しかし，ランク r はデータ数が増大するほど大きくなっていくため，これを固定次元 m のベクトルに変換したい．X を分解したものであるので，X を最もよく近似できる r 次元中の m 次元を選択するのが簡単である．特異値分解の場合には，特異値の大きな m 次元を採用すればよいことが知られている．上記によって LSI における単語ベクトル，文書ベクトルが得られた．

LSI の発展形として，確率モデルによる probabilistic LSI (pLSI) [8]，Latent Dirichlet Allocation (LDA) [9] などが存在する．これらの確率モデルでは，文書生成を各単語を独立に選択する過程だと考える．トピックと呼ばれる，単語上の多項分布を潜在変数として導入するため，これらのモデルはトピックモデルと呼ばれる．トピックはそのトピックにおける各単語の選ばれやすさを表している．文書はトピック上の確率分布で特徴づけられ，文書ごとのトピックの選択されやすさを表している．トピックが m 個の

場合，$v \times m$ 行列 U の列ベクトルがトピックを表し，$c \times m$ 行列 V の行ベクトル $\boldsymbol{v}_j$ が j 番目の文書のトピック分布であるとすれば，LSI と同様の形で pLSI モデルのトピックに関するパラメータを表現できる．U と V は LSI と同様に，単語ベクトルと文書ベクトルに相当するパラメータになるが，LSI における単語ベクトルであった U の行ベクトル $\boldsymbol{u}_i$ は，pLSI では一義的には「単語 i が各トピックにて選ばれる確率」を表す．

pLSI においては，データの文書群の生成は (i) 文書を選択し，(ii) 文書のトピック分布からトピックを選択し，(iii) トピック分布から単語を選択する，というプロセスが独立に，文書群の総単語数回行われたと考える．(i) で各文書が選択される確率を表す $\boldsymbol{d}$ をパラメータに加えて，データから最尤となる $\boldsymbol{d}, U, V$ を求めることで pLSI における文書・単語の表現を得ることができる．LDA はさらに U, V に事前分布を仮定するなどしたモデルである．これらのモデルではあくまで U はトピックの表現であり，単語の表現を意図したものではないため本稿ではこれ以上触れない．実応用においても pLSI や LDA の結果をそのまま単語ベクトルとして用いることは少ない．

2.2.2 Word2Vec

次に，predictive な手法で実用上現在最も広く用いられているであろう Word2Vec について触れたい．Word2Vec は 2013 年に Mikolov らによって考案されたアルゴリズムで，ここでは特に Skip-gram with Negative Sampling (SGNS) というアルゴリズムについて触れる．まず，Skip-gram モデルでは，ある単語の単語ベクトルを入力として，前後の単語の単語ベクトルを予測する．単語の系列である文 $\boldsymbol{w}$ があったとすると，以下の目的関数を最大化する．前後の文脈 c 単語の中での順序などは考慮されない．

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-c\le j\le c\\ j\neq 0}}\log P\left(w_{t+j} \mid w_t\right) \tag{1}$$

このとき各単語 w に対して 2 種類のベクトルが定義され，v_w を input ベクトル，v'_w を output ベクトルと呼ぶ．input ベクトルは単語が前後の単語の予測の入力として使われるときの単語ベクトル，output ベクトルは単語が前後の文脈の単語として用いられるときの単語ベクトルである．なお，Python における自然言語処理のスタンダードなライブラリの一つである gensim [10] などでは input ベクトルをデフォルトで出力する単語ベクトルとしている．

Skip-gram では語彙数 V のとき，文書中のある単語 w_I の文脈で単語 w_O が出現する確率は，

$$P\left(w_O \mid w_I\right) = \frac{\exp\left(v'^{\mathsf{T}}_{w_O} v_{w_I}\right)}{\sum_{w=1}^{V}\exp\left(v'^{\mathsf{T}}_{w} v_{w_I}\right)} \tag{2}$$

とモデリングされる．単語ベクトルが類似している単語は同じ文脈での出現率が高くなるようになっている．

h 次元の単語ベクトルを推定する Skip-gram は 2 層のフィードフォワード型ニューラルネットワークとして表現できる．フィードフォワード型ニューラルネットワークは線形変換のパラメータを (W, b)，非線形変換を f としたとき，それらを組み合わせた $f(W\boldsymbol{x}+\boldsymbol{b})$ の形の関数（層）を 1 回以上関数合成したものである．以下では $\boldsymbol{b}=\boldsymbol{0}$ とする．

まず第 1 層は単語を One-hot 表現で入力するとその単語の input ベクトルが出力されるように設定する．W を One-hot 表現に対応するように input ベクトル v を並べた $h \times V$ 次元の行列とし，非線形変換は省略する．このとき第 1 層の出力は w_k が One-hot 表現として入力された場合 v_{w_k} となる．第 2 層では W を第一層と同様に output ベクトル v'^{T} を並べた $V \times h$ 次元の行列とし，非線形変換には以下の softmax 関数を利用する．softmax 関数により最終出力のベクトルが確率分布の表現となる．

$$softmax\left(\boldsymbol{x}\right)_i = \frac{\exp x_i}{\sum_i \exp(x_i)}$$

このとき，このニューラルネットワーク全体を g と表すと，$g(w_k)_l$ は

$$g(w_k)_l = \frac{\exp\left(v'^{\mathsf{T}}_{w_l} v_{w_k}\right)}{\sum_{w=1}^{V}\exp\left(v'^{\mathsf{T}}_{w} v_{w_k}\right)}$$

となる．$w_k = w_I, w_l = w_O$ となるように単語インデックスを設定すれば $g(w_k)_l = P(w_O|w_I)$ となり，式 (2) をフィードフォワード型ニューラルネットワークの枠組みで実装することができる．

ここで，最も計算コストがかかるのは V 回内積と指数関数を計算する softmax 関数の分母である．Negative Sampling [11] は，softmax 関数をすべての語に対して真面目に計算するのでなく，サンプリングによって得られた単語のみに対して内積を計算する Skip-gram の高速化の工夫の一つである．sigmoid 関数を $\sigma\left(x\right) = \frac{1}{1+\exp\left(-x\right)}$ として，$\log P\left(w_O|w_I\right)$ を以下に

置き換える．

$$\log \sigma\left(v'^{\mathsf{T}}_{w_O} v_{w_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma\left(-v'^{\mathsf{T}}_{w_i} v_{w_I}\right)\right] \quad (3)$$

$P_n(w)$ は雑音分布と呼ばれる分布で，Negative Sampling で用いる単語をサンプルするための分布である．原論文は一様分布，出現頻度の経験分布より，出現頻度の経験分布の 3/4 乗を正規化した分布が実験的によかったと主張している．式 (3) は同じ文脈に出る単語ベクトルの内積を大きくし，ランダムサンプルした通常の文脈の単語ベクトルとの内積を小さくするという目的関数である．これにより真面目に softmax 関数を計算しなくても，実用的な単語ベクトルが得られるようになった．k は通常 5〜20 程度で十分であると言われており，語彙数が 10^5 程度であることを考えると大幅なスピードアップである．

2.3 Word2Vec 以降の研究の進展

Word2Vec の発明により，Embedding の研究は大きく進展した．2017 年現在でも多くの自然言語処理の論文では Word2Vec の単語ベクトルを用いている．一方で，Embedding の研究は Word2Vec 以降も，理論，実践上大きな進展が続いている．

2.3.1 Glove

Word2Vec の翌年，2014 年に Glove [12] が発表された．Glove はグローバルな情報を用いる count-based な手法とローカルな文脈の情報を用いる predictive な手法を組み合わせたもので，

$$\sum_{i,j=1}^{V} f(X_{ij}) \left(v'^{\mathsf{T}}_{w_j} v_{w_i} + b_i + b'_j - \log X_{ij}\right)$$

を最小化する． b は各単語に設定されたバイアス項である．Word2Vec と同様に $'$ は文脈ベクトルであることを表す．f はいくつかの性質を満たす関数である．X は単語–単語の共起行列で，X_{ij} は単語 w_i の前後の window 内に w_j が表れる回数を表す．

2.3.2 理論的解析の進展

同年，いくつかの前提をおくと SGNS モデルが shifted PMI 行列の分解と等価であるということが示された [13]．つまり，大きく別れていると思われていた count-based な行列分解系の手法と，predictive な手法の繋がりが示されたのである．離散分布において，Pointwise Mutual Information (PMI) は以下のように定義される．

$$PMI(x,y) = \log \frac{P(x,y)}{P(x)P(y)}$$

つまり，PMI は独立を仮定した場合に比べて共起確率がどれくらい大きいかを示す指標となっている．shifted PMI 行列は PMI 行列に定数を足したものである．この PMI 行列の分解を応用した Embedding として LexVec [14] も考案されている．

2.3.3 fastText

今まで紹介した手法は単語をベースとしているために，依然として未知語に対応するのが難しい．これを克服するためのアイデアとして単語より小さな単位で Embedding を行うという方法がある．Facebook が 2016 年に発表した fastText [15] では文字レベルの n-gram (character n-gram) である“sub-word”を用いる．単語のベクトルは，その単語を構成する sub-word のベクトルの和で構成される．原論文では 3-gram から 6-gram の sub-word を利用している．fastText ではたとえば，“egg” という単語は，単語の開始記号＜と終端記号＞を合わせて，3-gram の “＜eg”，“egg”，“gg＞”，4-gram の “＜egg”，“egg＞”，5-gram の “＜egg＞” の sub-word のベクトル和で表される．fastText は含まれる sub-word が類似している，つまり単語の表記が類似した単語は同様の意味をもつというモデルである．原論文の例では “english-born” の類似語として fastText は “british-born”，“polish-born” を挙げるのに対し，skip-gram では “most-capped”，“ex-scotland” を挙げる．これは一致した接尾辞をうまく拾うことができている例で，fastText の意図がうまく表れている．筆者らは，fastText の単語ベクトルから一つの sub-word を除外した際に元の語のベクトルから大きくずれる sub-word をいくつかの単語について分析しており，実際に接頭辞，接尾辞となっている場合を確認している．

2.3.4 Character-based Embedding

sub-word よりも小さい単位である文字ベースでの Embedding (Character-based Embedding) も存在しており，近年は特に RNN を用いた文章生成などで用いられている．英語であれば 10^2 程度の文字数であるために，たとえばそのまま ASCII コードを用いて 128 次元の One-hot 表現をすることも可能だが，日本語や中国語のような漢字文化圏では文字種数のオーダーが 2 桁以上異なるために個々の文字がより複雑な単語的意味をもっており，文字レベルでの Embedding が研

究されている．漢字の場合はその字形自体も意味をエンコードしているとも考えられ，字形を画像として捉え，Convolutional Neural Network を用いて視覚的特徴の Embedding を作成している例もある [16].

2.3.5 Word Embedding モデルのアンサンブル

今まで述べてきたように Embedding にはそれぞれ異なる目的関数を設定したさまざまな手法が存在しており，異なる情報をエンコードしていると考えられる．したがって複数の Embedding のパラメータを組み合わせ，アンサンブルするというのは自然な発想である．なお，機械学習の文脈においては学習済みのパラメータのことを指してモデルと呼称するため，これ以降では本稿でも“Embedding モデル”や“学習済み Word2Vec モデル”のような表現を用いる．

複数の Embedding モデルのベクトルを平均したり，結合させたりすることで精度が向上するということは以前より指摘されており [17]，異なる Word Embedding モデルを複数用いた Meta Embedding [18] が発表されている．また，WordNet のような人手で作られた概念のツリー構造を表すデータベースを利用して Embedding を計算する手法がいくつか提案されている．Word2Vec モデルに WordNet の情報を結びつけることで WordNet で定義されている抽象的な概念にベクトルを付与する AutoExtend [19] や，さらに WordNet に限らずツリー構造をポアンカレの円盤モデル上で表現した Poincaré Embeddings [20] のような手法も考案されている．

3. 実応用における Word Embedding

3.1 日本語環境下での Embedding

英語圏ではフリーの大規模コーパス（学習に利用する文章のデータセット）が多く存在し，それらを学習した Embedding モデルが数多く公開されている．さらに，本稿で紹介した新しい手法の多くの学習済みモデルも公開されている．一方で，日本語のコーパスで大規模でフリーなものは少なく，日本語における Embedding モデルの公開言語資源は英語圏と比べるとまだまだ少ないのが現状である．

しかし，いくつかのグループが主に Wikipedia のデータセットなどを用いて Word Embedding モデルを公開しており，これらを利用すれば手軽に Embedding の利用を始めることができる．モデルファイルは基本的には単語とそのベクトルのペアを記述した大きな辞書であるので，自分でパーサーを書くことも可能だが，gensim を利用するのが手軽だろう．深層学習に利用する際は，読み込んだベクトルをパラメータとして利用する深層学習ライブラリのモデルにコピーする必要がある．ただし，ここで気をつける必要があるのがモデルの単語の切り出し方である．「すももももももものうち」といった言葉遊びがあるが，日本語では単語の区切りは自明でなく，文を単語に分解するには形態素解析という処理を行う必要がある．著名な日本語形態素解析ツールには ChaSen や JUMAN，MeCab などがあるが，これらの出力結果は異なり，さらには同じ MeCab を利用していてもどの辞書を用いるかによっても変わる．MeCab 標準の辞書 (ipadic) を利用した場合と，固有名詞などを充実させた mecab-ipadic-neologd [21] では表 2 のような違いが出る．mecab-ipadic-neologd を使って形態素解析した文を ipadic を利用して作られた Word Embedding モデルでベクトル化しようとすると「ググった」の部分でモデルの語彙の違いでヒットせず未知語となってしまう．なお，近年では neologd が広く用いられている．また，変化形は原形に戻して学習していることが多い．したがって，表の例文の「見つから」は「見つかる」としてモデルを検索することになる．

表 2 MeCab による形態素解析の辞書による結果の違い

辞書	形態素
ipadic	ググ っ た けど 見つから ない
neologd	ググっ た けど 見つから ない

3.2 ファインチューニング

機械学習においては，まず一般的なデータで学習して得られたパラメータを初期値として，特定のタスクのデータや目的関数で再度学習することで，そのタスクにおける精度を向上させる手法をファインチューニングと呼ぶ．Embedding モデルのファインチューニングの方法は主に，目標とするタスクの目的関数に対して Embedding モデルもパラメータとして最適化することで達成される．前節で見たように，Word2Vec の各単語ベクトルは単語の One-hot 表現を入力としたニューラルネットワークの重みとして表現できるのだから，ニューラルネットワークが学習器として使われている場合，バックプロパゲーションが単語ベクトルに対しても行われるようにし，Embedding モデルのパラメータがタスクに対して最適化されるようにすればよい．

Embedding モデルを利用する際，試験的に利用する範囲では既存の公開モデルをそのまま使うだけでも十分だが，タスクの精度を向上させるにはやはり Embed-

ding モデルのファインチューニングは有力な選択肢の一つになる．もちろん，ファインチューニングという形でなく，特定のタスクのデータに対して Embedding モデルの学習を一から開始することもできるが，学習データが少量の場合，そのデータに含まれない単語はすべて未知語となってしまうなどさまざまな問題が発生し，うまくいかないことが多い．したがって，実用上は Wikipedia データのような大規模なデータセットで学習した Embedding モデルを，対象のタスクの精度を向上するように再学習するという形式がとられることが多い．

3.3 オンライン学習

Embedding のファインチューニングを行うとき，新しいコーパスを利用して Embedding モデル自体を更新したい場合がある．たとえば，特定の領域のコーパスを利用して，その領域の語彙を Embedding モデルに追加したいという場合である．この場合，Embedding モデルをオンライン学習するということになる．広く使われている SGNS のアルゴリズムは Negative Sampling がコーパス全体の単語分布に依存しており，近年までオンライン学習のアルゴリズムが確立していなかった．したがって，別のコーパスを利用して，既存の Embedding モデルを初期値とし SGNS によりファインチューニングを行おうとする場合，実タスクでのスコアを見てイテレーション回数を決めるといった方法しかなく，汎化性能などの点で不安が残るものだった．また，元のコーパスにない単語をどう扱うかという問題が存在していた．Kaji and Kobayashi [22] などによって，動的な語彙をもつ SGNS のオンライン学習アルゴリズムが考案されており，今後の普及が待たれる．

3.4 文書分類タスクによる実験

ここまでさまざまな Embedding 手法を紹介してきた．前述の日本語環境の問題で，Word2Vec と fastText が広く使われている．ここでは，弊社で作成した実験用データセットを用いてファインチューニングの実験を行ってみたい．400 クラスの短文分類問題に対して各クラス 10 文を学習データ，5 文をテストデータとし，3-fold クロスバリデーションを行い正解率を測定する．

BEDORE エンジンではユーザーの発言が質問の場合，用意されている FAQ から適切な回答を返す機能がある．このデータセットは 400 種類の FAQ それぞれに当てはまるべきユーザーの問い合わせを想定して作成した．この機能を実現するには一般に二つの方法

表 3　文書分類タスクの実験結果

手法	正解率
Word2Vec+LSTM	0.39
fastText+LSTM	0.41
fine-tuned Word2Vec+LSTM	0.43
fine-tuned fastText+LSTM	0.42
BoW+NN	0.32

がある．ユーザーの問い合わせに対して各 FAQ に何らかのスコア（一般には関連度など）をつけるランキングモデルによって回答を選定する方法，そしてあらかじめ分類先の FAQ を固定しクラス分類として解く方法だ．ここでは実験の設定を簡単にするために後者のモデルを採用する．

学習データの語彙数は 905 語であった．Wikipedia と BEDORE で作成した訓練データをコーパスとし，MeCab/mecab-ipadic-neologd で分かち書きした Word2Vec モデルと fastText モデルを 300 次元で作成した．Word2Vec や fastText を特徴量として用いる場合，文の長さという可変長の要素をいかに吸収するかという問題がある．たとえば，TF-IDF という文における語の重要性を評価する指標を重みとした単語ベクトルの加重平均をベクトルにするなどの方法がある．ここでは RNN の一種である Long Short-Term Memory (LSTM) に系列データとして投入することで固定長のベクトルを得る．LSTM を用いる手法では，分類する各文を同様に分かち書きし，Embedding モデルに存在しない語は無視して順に LSTM に入力し，最終的に全結合層 2 層を経由して 400 次元で出力する．参考に，BoW 表現（905 次元）を 3 層の全結合ニューラルネットワークにかけたモデル (BoW+NN) での精度も計測した．非線形変換は簡単のためすべて softmax 関数を用いた．BoW 表現においては，学習データにない語は表現できないため無視される．ファインチューニングは Word2Vec と fastText のそれぞれのモデルを単語ベクトルとして読み込み，文書分類タスクの正解率を最適化するように行った．

表 3 が利用した手法とその結果である．確かにこの実験設定ではファインチューニングによる精度向上が確認できた．元の Word2Vec と fastText では fastText のほうが高性能であるのに対し，ファインチューニングを行うと両者とも正解率が向上し，順位が逆転した．BoW 表現を用いた手法と Embedding を利用した手法の性能の差にはさまざまな原因が考えられるが，今回の実験設定では学習データが小さいためテストデータに出現する単語をカバーできず，未知語に対応でき

ない BoW 表現では精度が低くなっていると推測される．このタスクにおいては実際はユーザーの使用する語彙は制御不能であるため，Embedding の利用は合理的といえるだろう．この実験設定は簡単なものであり，アルゴリズムの性能はデータとタスクに依存するため一般化は難しいが，Embedding の利用は実応用において精度向上をもたらすことが多い．また，この実験で見られたファインチューニングにおける精度順位の逆転など，直感的でない精度変化も多々起きうるため，精度向上を考える際は網羅的な実験をするべきだろう．

4. 最後に

本稿ではさまざまな Embedding アルゴリズムを紹介したが，このほかにも紹介しきれないさまざまな研究が行われており，Embedding に関する研究は Word2Vec の登場以降も現在に至るまで着実な進展を遂げている．また，実応用における利用についても言及したが，現在でも自然言語処理のタスク性能向上には Embedding モデルのチューニングが気軽さと効果の点から有用であるといえるだろう．本稿が読者の Embedding モデルの利用の助けとなれば幸いである．

参考文献

[1] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," arXiv: 1301.3781, 2013.

[2] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba and S. Fidler, "Skip-thought vectors," In *Advances in Neural Information Processing Systems*, **28**, pp. 3294–3302, 2015.

[3] M. Sahlgren, "The distributional hypothesis," *Italian Journal of Linguistics*, **20**, pp. 33–53, 2008.

[4] ALAGIN, "単語共起頻度データベース," Advanced LAnGuage INfomation Forum, 2011.

[5] M. Baroni, G. Dinu and G. Kruszewski, "Don't count, predict!: A systematic comparison of context-counting vs. context-predicting semantic vectors," In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 238–247, 2014.

[6] J. Turian, L. Ratinov and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394, 2010.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, **41**, pp. 391–407, 1990.

[8] T. Hofmann, "Probabilistic latent semantic indexing," In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50–57, 1999.

[9] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, **3**, pp. 993–1022, 2003.

[10] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, 2010.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," In *Advances in Neural Information Processing Systems*, **26**, pp. 3111–3119, 2013.

[12] J. Pennington, R. Socher and C. D. Manning, "Glove: Global vectors for word representation," In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

[13] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," In *Advances in Neural Information Processing Systems*, **27**, pp. 2177–2185, 2014.

[14] A. Salle, A. Villavicencio and M. Idiart, "Matrix factorization using window sampling and negative sampling for improved word representations," In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 419–424, 2016.

[15] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, **5**, pp. 135–146, 2017.

[16] F. Liu, H. Lu, C. Lo and G. Neubig, "Learning character-level compositionality with visual features," arXiv: 1704.04859, 2017.

[17] J. Garten, K. Sagae, V. Ustun and M. Dehghani, "Combining distributed vector representations for words," In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pp. 95–101, 2015.

[18] W. Yin and H. Schütze, "Learning word meta-embeddings," In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1351–1360, 2016.

[19] S. Rothe and H. Schütze, "Autoextend: Extending word embeddings to embeddings for synsets and lexemes," arXiv: 1507.01127, 2015.

[20] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," arXiv: 1705.08039, 2017.

[21] T. Sato, "Neologism dictionary based on the language resources on the web for mecab," https://github.com/neologd/mecab-ipadic-neologd, 2015.

[22] N. Kaji and H. Kobayashi, "Incremental skip-gram model with negative sampling," arXiv: 1704.03956, 2017.

