単語分散表現獲得法の縮約モデリング

鈴木 潤　　永田 昌明
NTT コミュニケーション科学基礎研究所
{suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

1 はじめに

SkipGram, CBoW, GloVe, vLBL といった，いわゆる単語分散表現獲得法は，最も単純なニューラル言語モデルという位置づけで提案された [14, 10, 12, 11]．本稿では，区別するためこれらの方法を総称して特別に「ニューラル単語埋め込み法 (neural word embedding methods)」と呼ぶこととする．また，ニューラル単語埋め込み法によって得られる単語に割り当てられるベクトルを特別に「単語埋め込みベクトル (word embedding vector)」と呼ぶこととする．ニューラル単語埋め込み法の共通の特徴として，単語埋め込みベクトルの内積を用いて，特定の単語共起情報を近似するモデルを採用している点が挙げられる．この単純なベクトル内積モデルで単語共起情報を近似する仕組みは，ニューラル言語モデルの弱点であった，大規模データを扱う際の計算コストと，語彙数に対するスケーラビリティを劇的に向上させることを主な目的として考案された．事実，ニューラル単語埋め込み法では，数百万規模の語彙数と数十億規模の学習データ量を，単一の計算機で，かつ，現実的な実行時間での実行が可能である．また，得られた単語埋め込みベクトルの利用方法は，機械学習法での追加の素性として利用する方法，ニューラルネットの良い初期値として利用する方法，意味的な距離尺度として利用する方法など，多岐にわたる．また，翻訳，言語解析，質問応答，対話など自然言語処理分野の広範な領域で利用されている．

このように，計算リソースの観点での扱いやすさと単語間類似度としての利用価値の高さを高いレベルで両立可能である点が注目され，ニューラル単語埋め込み法は，ここ数年で自然言語処理分野の多くの場面で利用されるようになった．現在，単語埋め込みベクトルは，辞書やシソーラスのように，自然言語処理全般を支える基盤の言語資源と同等の位置付けになったと言っても過言ではないと思われる．

本稿では，このように今後の自然言語処理分野に大きな影響力があると考えられる単語埋め込みベクトルの利便性を更に向上させることを考える．ここでは特に，単語埋め込みベクトルを自然言語処理システム内で扱う際に必要となるメモリ量（または必要保存容量）を削減することに焦点をあてる．これは前述のように，単語埋め込みベクトルが自然言語処理の基盤資源であると仮定すると，その基盤資源のためだけに多くのメモリを占有するのは利便性を損なうので，極力軽減したいというモチベーションから，この題材を取り上げた．

2 ニューラル単語埋め込み法

ニューラル単語埋め込み法をニューラルネットの枠組みに従って解釈すると，入力単語に対する単語埋め込みベクトルと，出力単語に対する単語埋め込みベクトルの二つを用いて定式化される．一般的には，入力と出力単語の語彙は同じものを用いる場合があるが，同じでなければいけない理由はないため，ここでは，入力単語の語彙を U，出力単語の語彙を V と記述する．また，$|U|$ を入力単語の語彙数，$|V|$ を出力単語の語彙数とする．次に，i 番目の入力単語に割り当てる単語埋め込みベクトルを $\mathbf{e}_i$ と記述する．同様に，j 番目の出力単語に割り当てる単語埋め込みベクトルを $\mathbf{o}_j$ と記述する．この時，$\mathbf{e}_j$ と $\mathbf{o}_j$ 共に D 次元ベクトルとし，$1 \leq i \leq |U|$，$1 \leq j \leq |V|$ とする．

ここで簡単のため，$\mathbf{E} = (\mathbf{e}_1, \cdots, \mathbf{e}_{|U|})$，$\mathbf{O} = (\mathbf{o}_1, \cdots, \mathbf{o}_{|V|})$ とおく．また，$\mathcal{D}$ を学習用のデータとする．この時，ニューラル単語埋め込み法では，以下の最適化問題を解くことで，単語埋め込みベクトル $\hat{\mathbf{E}}$，$\hat{\mathbf{O}}$ を獲得する．

$$(\hat{\mathbf{E}}, \hat{\mathbf{O}}) = \arg\min_{\mathbf{E},\mathbf{O}} \big\{ \Psi(\mathbf{E}, \mathbf{O} \mid \mathcal{D}) \big\} \tag{1}$$

ただし．Ψ は目的関数とする．

例えば，‘SkipGram with negative sampling (SGNS) [11]’ の場合は，以下の目的関数を用いて単語埋め込みベクトルを学習していることになる．

$$\Psi(\mathbf{E}, \mathbf{O} \mid \mathcal{D}) = \sum_{(i,j)} \Big(c_{i,j} L(x_{i,j}) + c'_{i,j} L(-x_{i,j}) \Big) \tag{2}$$

ただし，$x_{i,j} = \mathbf{e}_i \cdot \mathbf{o}_j$ であり，$L(x)$ は logistic 損失関数，つまり $L(x) = \log(1 + \exp(-x))$ とする．また $c_{i,j}$ は，i 番目の入力単語と j 番目の出力単語が学習用データ $\mathcal{D}$ 上に出現した回数を表し，$c'_{i,j}$ は negative サンプリングデータ $\mathcal{D}'$ 上に出現した回数を表すこととする．

同様に，logistic 損失関数 $L(x)$ の代わりに最小二乗誤差損失関数 $S_y(x)$ を利用する場合は，以下のようにかける．

$$\Psi(\mathbf{E}, \mathbf{O} \mid \mathcal{D}) = \frac{1}{2} \sum_{(i,j)} \Big(c_{i,j} S_y(x_{i,j}) + c'_{i,j} S_y(-x_{i,j}) \Big)$$

ただし，$S_y(x) = (y - x)^2$ である．実際，$m_{i,j} = \frac{c_{i,j} - c'_{i,j}}{c_{i,j} + c'_{i,j}} y$ および $\beta_{i,j} = c_{i,j} + c'_{i,j}$ とおくと，文献 [15] で提案された Global Vector (GloVe) の目的関数の形と一致する．

$$\Psi = \frac{1}{2} \sum_{(i,j)} \boldsymbol{\beta}_{i,j} (\mathbf{e}_i \cdot \mathbf{o}_j - m_{i,j})^2 \tag{3}$$

このように手法によって，目的関数の違いはあれど，ベクトルの内積を用いて単語の共起情報を近似する方法である点は共通している．

3 ニューラル単語埋め込み法の縮約

以下本稿での提案法を述べる．前述の通り本稿では，単語埋め込みベクトルの必要メモリ量（または必要保存容量）を削減する方法論を提案する．

3.1 基本のアイディア

提案法の基本は，文献 [18, 7, 19] で考案した縮約モデリング法をベースとし，ニューラル単語埋め込み法へ適用するための拡張を行った．

提案法を説明するために，まず，D 次元の単語埋め込みベクトルを，B 個の固定長のベクトルに分割することを考える．分割した固定長ベクトルの次元が D' とすると，$D = B \times D'$ の関係が必ず成り立つと仮定する．例えば，$D = 256$，$B = 8$ なら $D' = 32$ である．本稿では，便宜上，単語埋め込みベクトルから分割したベクトルを「分割ベクトル」と呼ぶ．また，$\mathbf{e}_i$ に対して B 個に分割した際の b 番目の分割ベクトルを $\mathbf{e}_{i,b}$ と記述する．更に，分割ベクトルを順番に連結すると元の単語埋め込みベクトルと一致することを，演算子 $\bigodot$ を導入して，$\mathbf{e}_i = \bigodot_{b=1}^{B} \mathbf{e}_{i,b}$ で表す．これは，ある集合 $\mathcal{A}$ とその部分集合族 $\mathcal{A}_b$ が，和集合の演算子 $\bigcup$ によって $\mathcal{A} = \bigcup_{b=1}^{B} \mathcal{A}_b$ などと記述されるのと同様の意図で用いている．

次に，任意の D' 次元のベクトルが S 本あると仮定する．ここでは便宜上，このベクトルを「基底ベクトル」と呼び，s 番目の基底ベクトルを $\mathbf{w}_s$ と表す．また，全ての基底ベクトルの集合を $\mathbf{W}$ で表す．つまり $\mathbf{W} = \{\mathbf{w}_s \mid s = \{1, \ldots, S\}\}$ である．本稿では，全ての入力および出力ベクトル $\mathbf{e}_i$，$\mathbf{o}_j$ に対する分割ベクトル $\mathbf{e}_{i,b}$，$\mathbf{o}_{j,b}$ は，基底ベクトルの一つと必ず一致することを制約条件として追加したニューラル単語埋め込み法を提案する．

よって，提案法では，一つの単語埋め込みベクトル $\mathbf{e}_i$ は，B 個の分割ベクトルが個々に選択した基底ベクトルの組み合わせによって表現されることになる．一つの単語埋め込みベクトルが取り得る可能な種類数は S^B 個である．例えば $S = 16$，$B = 8$ の場合，一つの単語埋め込みベクトルは，$16^8 = 4,294,967,296$ 通りの値を取ることができる．一般的にニューラル単語埋め込み法で扱う単語数は数百万規模，どんなに多くても数千万程度あれば自然言語処理には十分な数と言えるので，例で示した 43 億種類程度あれば，種類数としては十分であると考えられる．

3.2 必要メモリ容量の見積り

基底ベクトルは S 種類なので，何番目の基底ベクトルかを表すのには $\lceil \log S \rceil$bit 必要である．よって，一つの単語埋め込みベクトルは，B 個の分割ベクトルの連結で表せる，かつ，各分割ベクトルは一つの基底ベクトルと一致するので，$B \times \lceil \log S \rceil$bit で表現できる．結果，入力側の語彙数が $|U|$ 単語とすると，入力単語埋め込みベクトルは，全部で $|U| \times B \times \lceil \log S \rceil$bit になる．同様に，出力側の単語埋め込みベクトルも，$|V| \times B \times \lceil \log S \rceil$bit で表現できる．また，一つの基底ベクトルは，倍精度浮動小数点 (64bit) が D' 個で構成されるので，S 種類の基底ベクトルを表現するのに $64 \times S \times D'$bit となる．最終的に，$(|U| + |V|) \times B \times \lceil \log S \rceil + 64 \times S \times D'$bit で全語彙の単語埋め込みベクトルが格納可能である．

具体例として，単語埋め込みベクトルの次元数が 256 次元，語彙数が入力，出力共に 40 万語だった場合を想定する．つまり，$D = 256$，$|V| = |U| = 400,000$ である．この設定では，256 次元の単語埋め込みベクトルが合計 80 万本構築されることになる．このとき，提案法のパラメタとして $B = 8$，$S = 256$，$D' = 32$ を選択すると，最終的に，$(400,000 + 400,000) \times 8 \times 8 + 64 \times 256 \times 32 = 51,724,288$bit(約 6.5MB) で全語彙の単語埋め込みベクトルが格納可能である．

一方，従来のニューラル単語埋め込み法による単語埋め込みベクトルを無加工で取得すると，この設定では $(400,000 + 400,000) \times 256 \times 64 = 13,107,200,000$ であり，単純計算で約 1.6GB の容量を必要とする．ただし，ここでは一次元の実数値を倍精度浮動小数点として 64bit で換算している．単純な 8bit(1 バイト) 量子化を利用したとしても約 200MB となる．よって，提案法は従来法より 1/30 から 1/250 程度の削減ができる．

更に通常，大規模な単語埋め込みベクトルを構築する際には，語彙数 200 万語などの設定を用いる．その場合は，提案法の効果はさらに意味を増すと考えられる．

3.3 定式化

前節にて，提案法がうまく機能した場合には，原理的に大幅に必要メモリ量を削減できる可能性があることを示した．ここでは，実際にどのように所望の単語埋め込みベクトルを獲得するかの部分を述べる．提案法では，従来のニューラル単語埋め込み法の最適化問題である式 (1) を拡張し，以下の最適化問題を解く．

$$
\begin{aligned}
(\hat{\mathbf{E}}, \hat{\mathbf{O}}) = \mathop{\arg\min}_{\mathbf{E},\mathbf{O},\mathbf{W}} & \left\{\Psi(\mathbf{E}, \mathbf{O}|\mathcal{D})\right\} \\
\text{s.t.} \quad & \mathbf{e}_{i,b} \in \mathbf{W}\ \forall (i,b),\ \ \mathbf{o}_{j,b} \in \mathbf{W}\ \forall (j,b), \\
& \mathbf{e}_i = \bigodot_{b=1}^{B} \mathbf{e}_{i,b}\ \forall i,\ \ \mathbf{o}_j = \bigodot_{b=1}^{B} \mathbf{o}_{j,b}\ \forall j,
\end{aligned}
\tag{4}
$$

定式化的には従来法との差分は非常に明快で，制約式が増え，かつ，最適化変数として基底ベクトル $\mathbf{W}$ が増えた，という違いになる．この制約の追加と，基底ベクトルの導入によって，前述の単語埋め込みベクトルの縮約をニューラル単語埋め込み法の学習の中で実現する．学習問題で定義されるいわゆる損失関数に相当する部分は，本提案法に直接影響を与えないので，例えば式 (2) に示した SGNS の目的関数といった既存手法と同一のものを用いることを想定する．

3.4 解法

式 (4) の最適化問題を解く方法はいくつも考えられるが，ここでは，既存手法との親和性を担保する意図で双対分解の概念を利用した解法を提案する．まず，式 (4) の最適化問題を簡単な問題に分解するために補助変数 $\mathbf{u}_i$ と $\mathbf{v}_j$ を導入し，$\mathbf{U} = (\mathbf{u}_i)_{i=1}^{|U|}$，$\mathbf{V} = (\mathbf{v}_j)_{j=1}^{|V|}$ とする．このとき，$\mathbf{e}_i$ および $\mathbf{o}_j$ と基底ベクトル間の制約に関する直接依存関係を排除するための式変形を行う．

$$
\begin{aligned}
(\hat{\mathbf{E}}, \hat{\mathbf{O}}) = \mathop{\arg\min}_{\mathbf{E},\mathbf{O},\mathbf{U},\mathbf{V},\mathbf{W}} & \left\{\Psi(\mathbf{E}, \mathbf{O}|\mathcal{D})\right\} \\
\text{s.t.} \quad & \mathbf{e}_i = \mathbf{u}_i\ \forall i,\ \ \mathbf{u}_{i,b} \in \mathbf{W}\ \forall (i,b), \\
& \mathbf{o}_j = \mathbf{v}_j\ \forall j,\ \ \mathbf{v}_{j,b} \in \mathbf{W}\ \forall (j,b), \\
& \mathbf{u}_i = \bigodot_{b=1}^{B} \mathbf{u}_{i,b}\ \forall i,\ \ \mathbf{v}_j = \bigodot_{b=1}^{B} \mathbf{v}_{j,b}\ \forall j,
\end{aligned}
\tag{5}
$$

式 (4) と式 (5) は等価である．次に，$\mathbf{e}_i = \mathbf{u}_i$，$\mathbf{o}_j = \mathbf{v}_j$ の等式制約を拡張ラグランジュ緩和法を用いた際に得られる緩和問題の目的関数 Φ は以下のようになる [3]．

$$
\begin{aligned}
& \Phi(\mathbf{E}, \mathbf{O}, \mathbf{U}, \mathbf{V}, \mathbf{A}, \mathbf{B}; \mathcal{D}) \\
& = \Psi(\mathbf{E}, \mathbf{O}|\mathcal{D}) \\
& \quad + \sum_i \boldsymbol{\alpha}_i \cdot (\mathbf{e}_i - \mathbf{u}_i) + \sum_j \boldsymbol{\beta}_j \cdot (\mathbf{o}_j - \mathbf{v}_j) \\
& \quad + \frac{\rho}{2} \sum_i ||\mathbf{e}_i - \mathbf{u}_i||_2^2 + \frac{\rho}{2} \sum_j ||\mathbf{o}_j - \mathbf{v}_j||_2^2 \\
\text{s.t.} \quad & \mathbf{u}_{i,b} \in \mathbf{W}\ \forall (i,b),\ \ \mathbf{v}_{j,b} \in \mathbf{W}\ \forall (j,b), \\
& \mathbf{u}_i = \bigodot_{b=1}^{B} \mathbf{u}_{i,b}\ \forall i,\ \ \mathbf{v}_j = \bigodot_{b=1}^{B} \mathbf{v}_{j,b}\ \forall j,
\end{aligned}
\tag{6}
$$

Step1 従来の最適化変数 $\mathbf{E}$, $\mathbf{O}$ に対する最小化:

$$(\hat{\mathbf{E}}, \hat{\mathbf{O}}) = \arg\min_{\mathbf{E},\mathbf{O}} \{\Phi(\mathbf{E}, \mathbf{O}, \mathbf{U}, \mathbf{V}, \mathbf{A}, \mathbf{B}; \mathcal{D})\} \quad (7)$$

Step2 補助変数 $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ に対する最小化:

$$(\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{W}}) = \arg\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \{\Phi(\mathbf{E}, \mathbf{O}, \mathbf{U}, \mathbf{V}, \mathbf{A}, \mathbf{B}; \mathcal{D})\} \quad \text{s.t.} \quad \mathbf{u}_{i,b} \in \mathbf{W}\ \forall(i,b),\ \mathbf{v}_{j,b} \in \mathbf{W}\ \forall(j,b), \quad (8)$$

Step3 双対変数 $\mathbf{A}$, $\mathbf{B}$ に対する最大化:

$$(\hat{\mathbf{A}}, \hat{\mathbf{B}}) = \arg\max_{\mathbf{A},\mathbf{B}} \{\Phi(\mathbf{E}, \mathbf{O}, \mathbf{U}, \mathbf{V}, \mathbf{A}, \mathbf{B}; \mathcal{D})\} \quad (9)$$

図 1　ADMM[3] に基づく最適化アルゴリズムの概略

ただし，$\boldsymbol{\alpha}_i$，$\boldsymbol{\beta}_j$ は共にラグランジュ未定乗数（双対変数）とし，$\mathbf{A} = (\boldsymbol{\alpha}_i)_{i=1}^{|U|}$，$\mathbf{B} = (\boldsymbol{\beta}_j)_{j=1}^{|V|}$ とおく．

拡張ラグランジュ緩和法で得た式 (6) に対して，ADMM[3] に基づく最適化アルゴリズムを用いる．ADMM では一定の最適化変数を固定し残りの最適化変数のみで構成された部分問題を最適化し，次に，固定する変数と最適化に用いる変数の役割を入れ替えて最適化を行うといった処理を任意の収束条件を満たすまで繰り返して最適化する方法である．提案法の場合は図 1 に示す 3 つの部分最適化問題を繰り返し解くことで最終的な解を得る．

式 (7) の部分最適化問題に対して，最適化変数となる $\mathbf{E}$, $\mathbf{O}$ に関連する項のみで構成された目的関数 Φ_1 は以下のようにかける．

$$\Phi_1 = \Psi(\mathbf{E}, \mathbf{O}|\mathcal{D}) + \frac{\rho}{2}\sum_i ||\mathbf{e}_i - \mathbf{e}'_i||_2^2 + \frac{\rho}{2}\sum_j ||\mathbf{o}_j - \mathbf{o}'_j||_2^2 \quad (10)$$

ただし，$\mathbf{e}'_i = \mathbf{u}_i - \frac{1}{\rho}\boldsymbol{\alpha}_i$，$\mathbf{o}'_j = \mathbf{v}_j - \frac{1}{\rho}\boldsymbol{\beta}_j$ である．目的関数の形からもわかるように，従来のニューラル単語埋め込み法の目的関数 Ψ に，バイアスあり L_2 正則化項に相当するものが追加された形となる．L_2 正則化項は強凸関数なので，勾配法に基づく最適化アルゴリズムには悪影響を与えない．また，文献 [3] にあるように，ADMM による最適化では，3 つの部分最適化問題を必ずしも厳密に解かなくても，経験的によい解が得られることが知られている．そこで，本稿では式 (7) の部分最適化問題は従来法と同じオンライン最急降下法 (SGD) を用いて最適化することとする．

次に，式 (8) の部分最適化問題に対して，最適化変数となる $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ に関連する項のみで構成された目的関数 Φ_2 は，同様に以下のようにかける．

$$\Phi_2 = \frac{\rho}{2}\sum_{(i,b)} \left\|\mathbf{u}'_{i,b} - \mathbf{u}_{i,b}\right\|_2^2 + \frac{\rho}{2}\sum_{(j,b)} \left\|\mathbf{v}'_{j,b} - \mathbf{v}_{j,b}\right\|_2^2 \quad \text{s.t.} \quad \mathbf{u}_{i,b} \in \mathbf{W} \quad \forall(i,b), \quad \mathbf{v}_{j,b} \in \mathbf{W} \quad \forall(j,b), \quad (11)$$

ただし，$\mathbf{u}'_{i,b} = \mathbf{e}_{i,b} + \frac{1}{\rho}\boldsymbol{\alpha}_{i,b}$，$\mathbf{v}'_{j,b} = \mathbf{o}_{j,b} + \frac{1}{\rho}\boldsymbol{\beta}_{j,b}$ である．これは，ユークリッド距離 (L_2 ノルム) に基づいて S 個のクラスに分割するクラスタリング問題と等価である．よって，例えば，一般的な k 平均クラスタリング法を用いることで式 (11) を解くことができる．ただし，式 (8) は，S 個の基底ベクトルへの各分割ベクトルの割り当て問題であり，NP 困難な問題であることが知られているため，厳密解を得ることは困難である．本稿では，k 平均クラスタリング法の最適解に $O(\log k)$ の近似比率で解が保証され，経験的に収束性の速い k-means++ 法 [2] を利用して近似解を得る．

表 1　実験結果

methods	容量	Analogy	Similarity	SentComp
SGNS	1.6GB	65.0	62.1	40.1
(16bit)	400MB	62.3	61.5	33.9
(8bit)	200MB	54.2	58.7	28.4
(4bit)	100MB	31.4	53.9	26.5
(2bit)	50MB	29.4	53.2	25.8
(k-means)	6.5MB	44.1	54.6	27.8
DC-SGNS	6.5MB	63.8	61.5	39.5

最後に，式 (9) の双対変数に対する最適化問題に対して，最適化変数となる $\mathbf{A}$, $\mathbf{B}$ に関連する項のみで構成された目的関数 Φ_3 は以下のようにかける．

$$\Phi_3 = \sum_i \boldsymbol{\alpha}_i \cdot (\mathbf{e}_i - \mathbf{u}_i) + \sum_j \boldsymbol{\beta}_j \cdot (\mathbf{o}_j - \mathbf{v}_j) \quad (12)$$

よって，式 (9) は，単純な勾配法を用いて解くことが可能である．

このように，式 (7)，(8)，(9) を順番にかつ反復して解くことで，最終的な解を得る．ただし，本稿で取り扱うニューラル単語埋め込み法は，そもそも多くの近似や高速化法を用いた実装により学習が行われていることから，提案法でも各部分問題は厳密に解くのではなく，オンライン学習の枠組みに則って，個々の学習データを読み込んでそれぞれの変数を更新するような方法で学習を進める．

4　実験

以下，本稿で行った評価実験について述べる．

4.1　学習データ

学習データには，既存研究で最もよく用いられる英語 Wikipedia のデータを用いた．前処理として，英語 Wikipedia アーカイブから 2014 年 8 月のダンプをダウンロードし，xml 形式から本文部分を抽出，文区切り，標準的なトークン区切り，小文字化処理を行った．最終的に，18 億 (1.8B) トークンのデータとなった．

4.2　単語分散表現評価用ベンチマークデータ

本稿では，単語埋め込みベクトルの評価に用いられてきたベンチマークデータを用いて評価実験を行った．Word Similarity タスク (Similarity) として，MEM[4], MTK[16] M&C[13], RAR[9], R&G[17], SLex[5], SCWS[6], WSR[1], WSS[1] の 9 種類のデータを準備した．同様に，Word Analogy タスク (Analogy) として，GSE[10], GSY[10], MSY[12] の 3 種類のデータを準備した．最後に，Sentence Completion タスク (SentComp) 文として，MSC[20] を準備した．各ベンチマークデータの詳細は，各参考文献を参照とし，ここではスペースの都合で割愛する．既存手法での評価と基本的に同じ処理により評価を行った．

4.3　実験結果

まず，ベースラインとなるニューラル単語埋め込み法として，文献 [8] 等に示されているように，安定して良い結果が得られることが知られている SGNS を用いることとし，著名な `word2vec`[*1] の実装を用いて実験を行った．また，SGNS の人手設定のハイパーパラメタ

[*1] https://code.google.com/p/word2vec/ 各単語二本のベクトルを取得可能に機能拡張したバージョン

は基本的に文献 [8] での推奨値，或いは，デフォルトの値を用いた．

表 1 に実験結果を示す．DC-SGNS の行が提案法の結果である．提案法の各種縮約パラメタの設定は，第 3.2 節で示した通りの値を利用した．ここで注意点として，提案法は単語埋め込みベクトルの精度向上が目的ではなく，精度を保ったまま必要メモリ量を削減することが目的である．よって実験結果より，提案法は，SGNS の結果からほぼ精度を落とさずに必要メモリ量だけを大幅に削減できたことが確認できた．

次に，提案法と同じ必要メモリ量を削減する方法としての簡単なベースライン法として，SGNS で得られた単語埋め込みベクトルに対して，単純に $m$bit 量子化した結果 (2bit)，(4bit)，(8bit)，(16bit) の評価も行った．$m$bit 量子化の手順としては，各次元毎に最大値と最小値を取得し，最大値-最小値の間を $m$bit で均等に量子化点を設定して量子化する方法を採用した．また，$m$bit 量子化と同様の後処理法として，学習後の単語埋め込みベクトルに対して，一度だけ提案法の式 (11) に相当する処理を施した後の単語埋め込みベクトルを用いた場合 (k-means) の結果も合わせて示す．結果からわかるように，これら単純な方法では，精度が大幅に低下することが確認できる．これらの結果との比較からも，従来通り，より精度の高くなる方向へ学習する中で，同時に必要メモリ量を削減できる単語埋め込みを探索する提案法は有効であることがわかる．

最後に，提案法では，必要メモリを増加させる方向へ縮約パラメタを緩和させると，SGNS の結果へ徐々に近づいていく．よって，縮約パラメタを変更し，必要メモリ量を多くした設定では，実験結果はほぼ変化しないため，実験結果として提示するのはスペースの都合上割愛する．

5 まとめ

本稿では，ニューラル単語埋め込み法により獲得できる単語埋め込みベクトルに対して更なる利便性を高めるため，必要メモリ量を削減する方法について議論を行った．従来通りの単語埋め込みベクトルの能力を保って必要メモリ量を削減するために，各単語埋め込みベクトルは任意の基底ベクトルの組み合わせで表現されるという制約を加え，その制約の下で単語埋め込みベクトルを獲得するニューラル単語埋め込み法を提案した．また最適化アルゴリズムの観点でも，ADMM の枠組みを利用して，従来のニューラル単語埋め込み法の学習アルゴリズムに k 平均クラスタリングを加え，それらを交互に最適化していく形式で，制約を満たした単語埋め込みベクトルを獲得する方法を新たに考案した．本稿の実験では，単語埋め込みベクトルの評価でよく用いられるベンチマークデータでの精度を保ったまま，大幅に必要メモリ量を削減することが可能であることを示した．

参考文献

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pp. 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[2] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning, 2011.

[4] Elia Bruni, Nam Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *J. Artif. Int. Res.*, Vol. 49, No. 1, pp. 1–47, January 2014.

[5] F. Hill, R. Reichart, and A. Korhonen. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *ArXiv e-prints*, August 2014.

[6] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pp. 873–882. Association for Computational Linguistics, 2012.

[7] Yotaro Kubo, Jun Suzuki, Takaaki Hori, and Atsushi Nakamura. Restructuring output layers of deep neural networks using minimum risk parameter clustering. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pp. 1068–1072, 2014.

[8] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, Vol. 3, , 2015.

[9] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 104–113, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, Vol. abs/1301.3781, , 2013.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

[12] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[13] George A. Miller and Walter G. Charles. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes*, Vol. 6, No. 1, pp. 1–28, 1991.

[14] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pp. 2265–2273. Curran Associates, Inc., 2013.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[16] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pp. 337–346, New York, NY, USA, 2011. ACM.

[17] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, Vol. 8, No. 10, pp. 627–633, October 1965.

[18] Jun Suzuki and Masaaki Nagata. Supervised model learning with feature grouping based on a discrete constraint. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 18–23, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[19] Jun Suzuki and Masaaki Nagata. Fused feature representation discovery for high-dimensional and sparse data, 2014.

[20] Geoffrey Zweig and Christopher J.C. Burges. The microsoft research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft Research, December 2011.