単語埋め込みの決定的縮約

仲村 祐希 [1]　鈴木 潤 [1,2]　高橋 諒 [1,2]　乾 健太郎 [1,2]
[1] 東北大学　[2] 理化学研究所
yuki.nakamura.r1@dc.tohoku.ac.jp,
{jun.suzuki, ryo.t, inui}@ecei.tohoku.ac.jp

1 はじめに

自然言語処理分野の多くのタスクで，深層学習による方法論が成功を収めている．自然言語処理タスクで主に処理対象となる文章を深層学習により処理する場合，文字や単語などの離散的な記号を連続的な数値に変換する必要がある．その橋渡しとして用いられるのが単語埋め込みの技術である．つまり，深層学習に基づく自然言語処理において単語埋め込みは，必要不可欠な要素技術に位置付けられる．また，それ故に単語埋め込みの機能拡張や性能向上は様々な方法論に寄与する波及効果の高い研究成果になることが期待できる．

単語埋め込みの改善の代表的な研究トピックとして，必要記憶容量の削減に関する研究がこれまで多く取り組まれてきた [1, 2, 3, 4, 5, 6, 7]．これは，語彙数が大きい場合に，単語埋め込みを保持するために必要な記憶容量が比較的大きくなるため，結果として深層学習のモデルの中でも非常に大きな容量を占めることがしばしばおこることに起因する．単語埋め込みの必要記憶容量削減には，似た単語には似たベクトルが割り当てられているという性質を利用して，似ている単語間で効果的にパラメータを共有する方法論が現在主流である [2, 4, 5]．その代表的な方法論として，単語埋め込みを離散符号とコードブックに分解し記憶容量を削減する手法が知られている．さらに，この離散符号と基底ベクトルを深層学習により同時学習する手法が提案されており，高い圧縮率を達成している．これらの方法論は記憶容量の削減の観点では成功しているが，学習時の最適化関数は局所解があることやサンプリングを必要とすることなどの複数の要因により，完全に同じデータ，同じ手順を用いたとしても，得られる離散符号が毎回異なるという性質を持つ．

そこで，本研究では離散符号を決定的アルゴリズムにより獲得する方法を提案し，その性能を検証する．元の単語埋め込み行列の特徴を保持したまま離散符号を獲得することで，似た単語には似た離散符号が割り当てられるように獲得する．この方法で獲得できる離散符号は乱数によらず決定的に求まり，再現性の面で優れている．また，この方法によって獲得する離散符号に適した効果的な基底ベクトルの合成方法も合わせて提案する．

2 関連研究

単語埋め込み行列を離散符号とコードブックに分解し記憶容量を削減する手法について様々な研究がされてきている．Shu らは Gumbel-softmax トリックを用いることで，離散符号とコードブックを深層学習を用いて同時学習する手法を提案した [2]．また，Tissier らは単語埋め込み行列の記憶容量の削減と演算の高速化を目指し，オートエンコーダで単語埋め込み行列をバイナリベクトルにエンコードする手法を提案した [4]．さらに，Kim らは下流タスクの各単語の重要度に応じて，離散符号長を合わせて学習することで効率よく記憶容量を削減する手法を提案した [5]．これらの既存研究は，深層学習ベースで離散符号を獲得するため離散符号は乱数のシードによって異なる．本研究での提案法は，決定的アルゴリズムで乱数のシードによらず，一意に定まる離散符号を獲得するという点でこれらの手法とは異なる．

3 提案手法

提案手法は離散符号の獲得（3.1 節）とコードブックの学習（3.2 節）の二つの処理で構成される．離散符号を決定的に獲得するためのアイデアは，1 次元 K 平均法の最適解は一意に定まり，かつ，動的計画法により多項式時間で求まる [8]，という特徴を離散符号の獲得に利用することである．すなわち，(1) 単語埋め込み行列をある 1 次元で切り出し，(2) 1 次元 K 平均法により各単語のクラス番号としての離

Algorithm 1 Calculating discrete code

Input: Word embeddings: $\boldsymbol{E} \in \mathbb{R}^{|V|\times H}$
Input: Discrete code length: M
Input: Kind of code: K
Output: Discrete codes: $\boldsymbol{C} \in \{1, 2, \ldots, K\}^{|V|\times M}$

```
C^(0) ← ∅
Ē ← Normalization(E)
Ẽ ← PCA(Ē)
for d = 1, ..., M do
    classes ← 1dKmeans(Ẽ_{:,d})
    C^(d) ← appendVec(C^(d−1), classes)
end for
return C^(M)
```

散符号を獲得する．(1) を異なる次元で M 回繰り返すことで，各単語に対して M 個の離散符号を得る．

また，単語埋め込みは冗長性が多く，全ての次元を使うのは記憶容量を削減する上で効率が悪い．そこで，主成分分析を用いて単語埋め込み行列の次元削減も合わせて活用する．最終的に，主成分分析で削減した行列の各列ベクトルに対して 1 次元 K 平均法を繰り返し適用し離散符号を獲得する．主成分分析も決定的な処理であるため，1 次元 K 平均法と合わせて一意に定まる離散符号が得られる．

3.1 離散符号の獲得：アルゴリズム

離散符号を獲得するアルゴリズムを Algorithm 1 に示す．まず，記憶容量の削減対象の単語埋め込み行列を $\boldsymbol{E} \in \mathbb{R}^{|V|\times H}$ とする．前述の通り，まず単語埋め込み行列 $\boldsymbol{E}$ の平均が 0，分散が 1 になるように規格化し (Normalization)，$\bar{\boldsymbol{E}}$ を得る．その後，$\bar{\boldsymbol{E}}$ に対して主成分分析 (PCA) を適用し主成分得点の行列 $\tilde{\boldsymbol{E}} \in \mathbb{R}^{|V|\times M}$ を得る．この行列 $\tilde{\boldsymbol{E}}$ の各列ベクトルを対象に 1 次元 K 平均法 (1dKmeans) を適用し列ベクトルに含まれる $|V|$ 個の要素 (スカラー値) のクラスタリングを行う．また，この処理を M 個の列ベクトルに対して先頭から順番に実行する．このとき，各 1 次元 K 平均法で得られたクラス番号を単語に割り振られた離散符号とみなすことができる．よって，最終的には各 1 次元 K 平均法で i 番目の要素に割り振られた M 個のクラス番号を並べたものを i 番目の単語に割り振られた離散符号とする．つまり，次元削減後の次元数を M，クラスタの数を K とすると離散符号は $\boldsymbol{C} \in \{1, 2, \ldots, K\}^{|V|\times M}$ と表現できる．

この手法は，各単語埋め込みを順序を保持したまま格子点上に埋め込み直していると考えることができる．そのため，十分に元の特徴を保持したまま離散符号化することができ，獲得した離散符号には，似たような単語には似たような離散符号が割り当てられていることが期待できる．

表 1 提案手法により獲得した離散符号の例

word	8 × 8 code							
king	7	7	4	6	5	5	0	3
man	7	7	1	4	7	4	1	6
woman	7	6	2	4	7	3	1	5
queen	7	6	2	6	5	4	0	6

3.2 コードブックの定義と学習

獲得した離散符号に対して，元の単語埋め込み行列を再現するようにコードブックを学習する．本手法で獲得した離散符号には各次元毎に順序関係が埋め込まれていると考え，コードブックとして基底ベクトルに離散符号に対応した重みをつけたものを用いる．具体的には，ある単語 w について離散符号 $\boldsymbol{C}(w) = [C_1(w), C_2(w), \ldots, C_M(w)](C_i(w) \in \{1, 2, \ldots, K\})$ が与えられているとき，単語埋め込みの近似行列 $\boldsymbol{E}' \in \mathbb{R}^{|V|\times H}$ を基底ベクトル $\boldsymbol{A} \in \mathbb{R}^{M\times H}$ と離散符号に対応する重み $\boldsymbol{\alpha} \in \mathbb{R}^{M\times K}$ を用いて次のように計算する．

$$\boldsymbol{E}'(w) = \sum_{i=1}^{M} \alpha_{i,c_i(w)} \boldsymbol{A_i} \tag{1}$$

この単語埋め込み近似行列と記憶容量削減対象の単語埋め込み行列の誤差が小さくなるように学習を行う．具体的には，基底ベクトル $\boldsymbol{A}$ と重み $\boldsymbol{\alpha}$ のパラメータを以下の誤差関数 L が最小となるように学習する．

$$L = \frac{1}{|V|} \sum_{w\in V} ||\boldsymbol{E}(w) - \boldsymbol{E}'(w)||^2 \tag{2}$$

4 実験

本研究では，単語埋め込みの内的評価にあたる単語類推タスク (Analogy)，単語類似性判定タスク (Similarity)，文穴埋めタスク (SentComp) の 3 つと，外的評価にあたる機械翻訳タスクを用いて実験を行った．実験では各タスクで一般的に用いられる評価指標の値とデータ圧縮比のトレードオフの観点で提案法を評価した．データ圧縮比は記憶容量削減前と記憶容量削減後の記憶容量の比である．なお，記

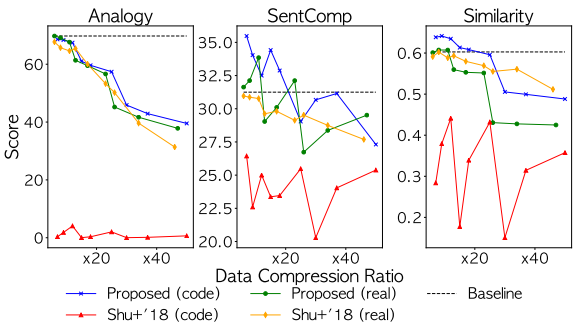

図 1　データ圧縮比に対する単語埋め込み評価タスクの各評価指標の値の変化. 評価指標は Analogy と SentComp は正答率，Similarity はスピアマンの順位相関係数である．Code の結果はコサイン類似度で評価しているため中心化を行った結果を報告する．

憶容量は理論的に計算を行った．また，比較対象のベースラインには Shu ら [2] の手法を用いた．

4.1　離散符号の獲得とコードブックの学習

提案法，および，比較手法である Shu らの方法では離散符号を獲得するために，あらかじめ単語埋め込み行列を用意しておく必要がある．単語埋め込みの評価タスクでは学習済み単語埋め込みである GloVe，機械翻訳タスクでは学習した深層学習モデルの埋め込み層のパラメータを記憶容量削減対象として用いた．コードブックの学習は，学習率は 0.0001，バッチ数は 256，1000 エポックで Adam を用いてパラメータの更新を行った．また，コードブックの学習の安定性のために基底ベクトルの重みの初期値には離散符号を獲得したときのクラスの重心を用いた．

4.2　単語埋め込みの評価タスク

このタスクでは単語埋め込み行列から獲得した離散符号が単語埋め込み行列の情報を含んでいるかどうか調べるために，単語類推タスク，単語類似性判定タスク，文穴埋めタスクの実験を行った．実験には語彙数 400000，300 次元の学習済み単語埋め込みの GloVe.6B.300d を用いた．

データセット：　単語類似性判定タスクでは 9 つ [9, 10, 11, 12, 13, 14, 15, 16]，単語類推タスクでは 3 つ [17, 18]，文穴埋めタスクでは 1 つ [17] のデータセットを用いて実験を行った．

実験結果：　実験の結果を図 1，表 2 に示す．Code は離散符号のみ，Real はコードブックを使って単語埋め込みを再構築した際の結果を示している．また，実際に獲得した離散符号を表 1 に示す．まず，全体の実験結果として提案法の Code がデータ圧縮比を上げたときの評価指標の低下が小さく良い結果が得られている．提案法の Code と Real を比較したときに Code の方が良い結果が得られているのは，離散符号の方がコードブックに対して占めている記憶容量が大きく情報量が大きいためだと考えられる．次に，Shu らの Code の Analogy が低い理由は，離散符号がどの基底ベクトルを使用するかしか表現しておらず，コードの足し引きに意味がないためである．一方で，提案法の離散符号には順序関係が存在するため Analogy で良い結果が得られている．これらのことから，提案手法により得られる離散符号には元の単語埋め込み行列の情報が十分に含まれていることが確認された．また，提案手法が SentComp と Analogy でベースラインよりも高い結果が得られている理由について，似た単語に似た離散符号を割り当てたことにより正則化の効果が得られるためだと考えられる．

表 2　単語埋め込み評価タスクにおける評価指標の値を落とさずに達成できるデータ圧縮比．acc は正答率，ρ はスピアマンの順位相関係数，ratio はデータ圧縮比を表す．

Model	Analogy		SentComp		Similarity	
	acc	ratio	acc	ratio	ρ	ratio
GloVe baseline	69.76	x1	31.25	x1	0.60	x1
Shu+'18 (Code)	4.97	x12	26.44	x7	0.44	x12
Shu+'18 (Real)	67.72	x6	30.77	x11	0.60	x8
Proposed (Code)	68.60	x7	31.15	x37	0.60	x25
Proposed (Real)	69.75	x6	32.12	x23	0.61	x11

解の安定性：　既存手法の安定性を調べるために，Shu らの手法を用いてデータ圧縮比を固定して複数回単語埋め込み行列の記憶容量の削減を行い評価を行った．実験はデータ圧縮比 11 倍でコードブックを使って単語埋め込み行列を再構築したものを用いた．Shu らの手法で測定した評価指標の最大値と最小値の差は，Analogy では 1.29，SentComp では 1.82，Similarity では 0.2 であった．具体的な測定結果は付録 A に示す．このように深層学習ベースによる記憶容量の削減は乱数のシードによって結果が大きく変化してしまう．一方で提案手法の離散符号を用いた場合，乱数のシード値によらず得られる離散符号は決定的であるため結果の変化はない．よって離散符号の個数 M と離散符号の種類数 K を決めてしまえば，離散符号が決定的に獲得できるという提案手法の特徴は大きな利点である．

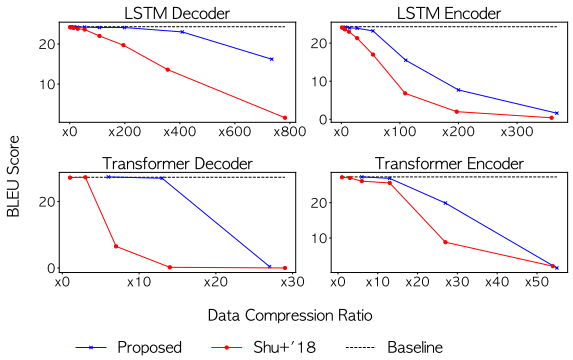

図 2　データ圧縮比に対する BLEU スコアの変化.

4.3　機械翻訳タスク

ニューラルネットワークの埋め込み層に対して本手法の記憶容量の削減が有効であるかを確認するために，機械翻訳モデルの埋め込み層のパラメータに対して記憶容量の削減を行い，データ圧縮比と翻訳精度の関係を調査した.

データセット：　WMT 2016 英独翻訳タスクのデータセットを利用した．検証データは newstest2013，テストデータは newstest2014 を用いた．Byte pair encoding [19] を適用し，語彙数は英語で 35856 語，独語で 35456 語となった.

実験設定：　機械翻訳モデルとして Transformer に基づくモデル [20] と，LSTM のエンコーダ部分を双方向にしたモデル [21] を用いた．基本的な学習設定は Ott ら [22] に従う．詳細なハイパーパラメタは付録 C を参照されたい．ただし，モデルのエンコーダとデコーダのそれぞれの埋め込み層のパラメータのデータ圧縮比を測定するため語彙の共有は行わなかった．各モデルの学習後に，提案法および Shu らの手法によって埋め込み層の記憶容量の削減を行った.

翻訳結果の出力にはビーム幅 10 のビーム探索を用いた．評価指標としてテストデータに対する sacreBLEU [23] を報告する.

実験結果：　実験の結果を図 2，表 3 に示す．まず，LSTM の結果についてエンコーダ，デコーダ共に提案手法の方がデータ圧縮比を上げても BLEU スコアの低下が抑えられていることがわかる．特にデコーダについては BLEU スコアをほぼ落とさずに (24.3 → 24.1) 記憶容量を 199 分の 1 にすることができた．次に，Transformer の結果についても同様に提案手法の方がデータ圧縮比を上げても BLEU スコアの低下が抑えられていることがわかる．LSTM と Transformer では LSTM の方がより記憶容量を削減できることが確認された.

表 3　機械翻訳タスクにおけるスコアを落とさずに達成できる圧縮比

Model	LSTM				Transformer			
	Encoder		Decoder		Encoder		Decoder	
	BLEU	ratio	BLEU	ratio	BLEU	ratio	BLEU	ratio
Baseline	24.3	x1	24.3	x1	27.3	x1	27.3	x1
Shu+'18	23.6	x6	23.6	x54	27.0	x3	27.3	x3
Proposed	23.9	x27	24.1	x199	26.9	x13	27.0	x13

今回用いた LSTM の総パラメータ数は約 2.4 億，Transformer の総パラメータ数は約 2.9 億であり，各埋め込み層のパラメータの占める割合は LSTM では約 15%，Transformer では約 13% である．そのため，埋め込み層の記憶容量削減後でもモデルの必要記憶容量は依然として大きい．そこで，モデルのさらなる記憶容量の削減手法として 8-bit 量子化などがあり，併用して適用することでさらにモデル全体の記憶容量を削減することが期待できる.

解の安定性：　提案手法と Shu らの手法を用いて，複数回データ圧縮比 13 倍で記憶容量を削減し BLEU スコアの変化を測定した．実験結果としてスコアの最大値と最小値の差は Shu らの手法は 0.7，提案手法は 0.1 であり，提案手法の方が安定性が高かった．しかし，データ圧縮比を上げると提案手法でもスコアが変動し，安定性の低下が見られた．なお，具体的な実験結果は付録 B に示す.

5　おわりに

1 次元 K 平均法および主成分分析を組み合わせた決定的アルゴリズムによる単語埋め込みに対する離散符号の獲得手法を提案した．また，得られた離散符号に適した効果的な基底ベクトルの合成方法も合わせて提案した．本手法は，既存の手法と比較し離散符号を乱数のシードによらず決定的に獲得することができるため，再現性に優れている．実験では本手法で獲得した離散符号には単語埋め込み行列の情報が十分に含まれていることを実験で明らかにした.

謝辞

本研究の一部は，JSPS 科研費 JP19H04162，及び JSPS 科研費 JP18J20936 の助成を受けたものです.

参考文献

[1] Jun Suzuki and Masaaki Nagata. Learning compact neural word embeddings by parameter space sharing. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 2046–2052. IJCAI/AAAI Press, 2016.

[2] Raphael Shu and Hideki Nakayama. Compressing word embeddings via deep compositional code learning. In *International Conference on Learning Representations*, 2018.

[3] Shota Sasaki, Jun Suzuki, and Kentaro Inui. Subword-based Compact Reconstruction of Word Embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3498–3508, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] Julien Tissier, Christophe Gravier, and Amaury Habrard. Near-Lossless Binarization of Word Embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 01, pp. 7104–7111, jul 2019.

[5] Yeachan Kim, Kang-Min Kim, and SangKeun Lee. Adaptive Compression of Word Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3950–3959, Stroudsburg, PA, USA, 2020. Association for Computational Linguistics.

[6] Sho Takase and Sosuke Kobayashi. All word embeddings from one embedding. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[7] Aliakbar Panahi, Seyran Saeedi, and Tom Arodz. word2ket: Space-efficient word embeddings inspired by quantum entanglement. In *International Conference on Learning Representations*, 2020.

[8] Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast Exact k-Means, k-Medians and Bregman Divergence Clustering in 1D. pp. 1–16, 2017.

[9] G. Miller and W. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, Vol. 6, pp. 1–28, 1991.

[10] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, Vol. 8, pp. 627–633, 1965.

[11] 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*, pp. 19–27, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[12] Elia Bruni, Nam-Khanh Tran, and M. Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, Vol. 49, pp. 1–47, 2014.

[13] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. New York, NY, USA, 2011. Association for Computing Machinery.

[14] Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 873–882, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[15] 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.

[16] Felix Hill, Roi Reichart, and Anna Korhonen. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, Vol. 41, No. 4, pp. 665–695, December 2015.

[17] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[18] 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.

[19] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008. Curran Associates, Inc., 2017.

[21] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[22] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[23] Matt Post. A call for clarity in reporting BLEU scores. In Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana L. Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pp. 186–191. Association for Computational Linguistics, 2018.

A　単語埋め込み評価タスクの安定性評価の実験結果

表 4　単語埋め込み評価タスクをランダムシードで複数回実験したときの評価指標の値の変化.

	1 回目	2 回目	3 回目	4 回目	5 回目	6 回目	7 回目	8 回目	9 回目	10 回目
Analogy(acc)	63.76	64.27	63.95	63.97	64.26	64.74	64.91	64.24	64.00	65.05
SentComp(acc)	30.38	30.67	31.63	31.44	31.44	30.58	31.63	29.81	30.19	31.25
Similarity(ρ)	0.59	0.59	0.60	0.58	0.59	0.59	0.59	0.58	0.59	0.59

B　機械翻訳タスクの安定性評価の実験結果

表 5　機械翻訳タスクをランダムシードで複数回実験したときのスコアの変化.

	1 回目	2 回目	3 回目	4 回目	5 回目	6 回目	7 回目	8 回目	9 回目	10 回目
Shu+'18	25.6	25.4	25.4	25.4	25.2	25.0	24.9	25.4	25.4	25.4
Proposed	27.0	26.9	26.9	26.9	26.9	26.9	26.9	26.9	26.9	26.9

C　機械翻訳タスクの学習設定

学習率は Transformer は 0.0005，LSTM は 0.001 とした．LSTM と Transformer 両方のモデルにドロップ率 0.3 のドロップアウトを適用した．GPU 4 台で分散学習を行い，GPU 1 台のバッチサイズは Transformer は 64，LSTM は 1 とした．また，パラメータの更新は Transformer は 10000 回，LSTM は 150000 回行った．LSTM には勾配のクリップを 1.0 に設定した．また，最終的に得られたパラメータの直近 10 エポックのチェックポイントの平均をとった値を最終的なパラメータとして用いた．