単語の重要度に応じてパラメタ数可変な単語分散表現の学習

露木 浩章　　小川 哲司　　小林 哲則　　林 良彦
早稲田大学理工学術院
tsuyuki@pcl.cs.waseda.ac.jp

1 はじめに

本研究では単語分散表現の圧縮を目的として，単語の重要度に応じてパラメタ数を可変とするゲート機構を提案しその有効性を評価する．

ニューラルネットワークを用いた自然言語処理モデルにおいて，単語ベクトルは単語の意味を表す特徴量の役割を果たす．しかし，モデルの精度向上のために単語ベクトルの次元サイズや語彙サイズを大きくすると，必要メモリ量が膨大となりモバイル端末上での処理が困難である．メモリコストの問題を解決しモバイル端末上で実行できれば，データの送受信コストやユーザデータの情報漏洩リスクを低減できる．また既存研究より，単語ベクトルのパラメタ冗長性を排除することは，必要メモリ量の削減だけでなく，過学習の抑制に基づくモデルの精度向上にも寄与することが分かっている [1].

単語ベクトルのパラメタ数が膨大な場合，出現回数が少ない統計的に信頼性の低い単語を<unk>に置換する前処理が用いられる．しかし，感情分析における感情語のようにタスクにおいて重要な役割を持つ単語 (重要語) を置換するとモデルの精度低下につながる．例えば，IMDB コーパス [2] を用いた感情分析の予備実験では出現回数が 2 回以下の語を<unk>に置換すると，置換しない場合と比べて感情語が 1098 個減り，0.9 ポイント識別精度が低下した．本研究では，識別に寄与する重要語に多くのパラメタを割り振り，寄与しない非重用語には少ないパラメタを割り振ることで，モデルの精度を落とすことなく単語ベクトルのパラメタ数を削減する手法を提案する．

パラメタ数の少ない単語同士は似たベクトルになりやすいため，提案法は感情語を<unk>に置換せずに残したまま，感情分析に寄与しない非重要語を<unk>に置換する機能を果たす．実験の結果，提案法はモデルの精度を落とすことなく従来法を上回る圧縮率を達成した．また，感情分析タスクで学習すると長いコードが割り当てられた語彙の中に感情語が多く存在することを確認した．

2 関連研究

単語ベクトルに要するメモリ量を削減する手法は語彙サイズを減らす手法と単語あたりの必要メモリ量を減らす手法の 2 つに大別できる．

語彙サイズを減らす手法として Byte Pair Encoding (BPE) [3] が存在する．BPE は目的コーパスにおける出現回数に応じてサブワード分割する．BPE は OOV 問題を解決でき，語彙サイズも低減できる強力な手法だが，文中に含まれるトークン数が増加することから，文ベクトルを構成する際に用いられる LSTM や Self-attention Network における計算量や必要メモリ量が増大しやすい．

単語あたりの単語ベクトルに要するメモリ量を削減する手法として量子化を用いた重み共有法が用いられる．全単語間で共有されるパラメタの内，入力単語に適切なパラメタを選択する離散値のコードを学習する．Neural Compressor (NC) [4] は最大値 K の自然数からなる長さ M のコードに基づき，M × K のコードブック (Decoder) から M 個の基底ベクトルを抜き出し，その和によって単語ベクトルを再構成する手法である．各単語に割り当てられた離散値コードに要する bit 数が小さくなり，単語ベクトルの高い圧縮率を達成した．Near-loss-Binarization (Bin) [5] では離散値コードをバイナリ化し，さらに高い圧縮率を達成した．NC と Bin ではいくつかのコーパスで元の単語ベクトルよりも圧縮語の単語ベクトルの方がタスクの精度が高くなることが報告されている．これは単語間で重みを共有することで低頻度語の学習が容易になったためであると考えられる．本研究と同様にバイナリコードの長さを単語ごとに可変とする手法として AdaComp [6] が存在する．AdaComp では (32,16,6,4) のように長さの異なるバ

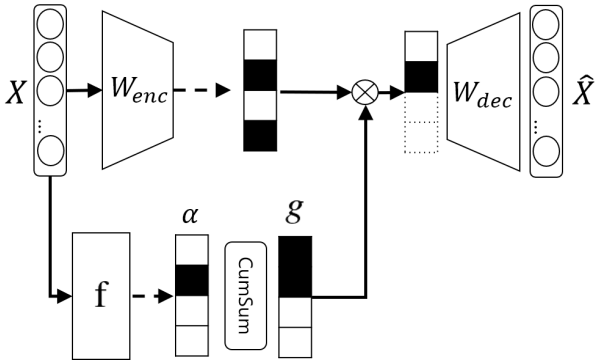

図 1 可変長コードを学習するニューラルネットワークの図．点線は gumbel softmax と Sign 関数の計算を表す．

イナリコードを扱う AutoEncoder を 4 つ用意し，単語ごとに使用する AutoEncoder を選択する手法である．しかし，異なるコード長を持つ単語間で異なるコードブック (Decoder) が用いられることから，NC や Bin で重要とされている重み共有が一部行われていない．また，コード長の種類を増やすと学習パラメタ数が増大し過学習しやすくなる．本研究では異なるコード長を持つ単語間でも同じコードブックを使用することで，重み共有を実現する．また，学習パラメタ数を増やさずにコード長の種類を増やす事が可能なためより細かいコード長の調整が可能である．

3 提案法

本節では，本研究で提案する手法について述べる．提案法は学習済みの単語ベクトル GloVe [7] からバイナリコードを得る AutoEncoder の機構 (3.1 節) と，入力単語ベクトルに応じて選択された長さ以上の要素をバイナリコードから削除するゲート機構 (3.2 節) からなる．

3.1 バイナリコードの学習

バイナリ制約のある AutoEncoder を使用する．学習済み単語ベクトルを入力とする AutoEncoder を学習した後，得られた中間特徴量を入力単語ベクトルのバイナリベクトル，Decoder を単語間で重み共有されるコードブックとして用いる．$x \in \mathbb{R}^d$ を入力単語ベクトル，m をバイナリコードの最大コード長とするとき単語ベクトルを以下のように再構成する．

$$\hat{x} = W_{dec}(\Phi(W_{enc}x + b_{enc}) \circ g) + b_{dec} \tag{1}$$

$W_{enc} \in \mathbb{R}^{d \times m}$ と $W_{dec} \in \mathbb{R}^{m \times d}$，$b_{enc}$，$b_{dec}$ は Encoder と Decoder の学習パラメタである．関数 $\phi(\cdot)$ と g は Encoder の出力をバイナリ化する関数とコード長を可変とするゲート機構である．関数 $\phi(\cdot)$ として以下の式 2 を用いる．

$$\Phi(h_i) = \mathrm{ReLU}(\mathrm{Sign}(h_i)) = \begin{cases} +1 & (x_i \geq 0) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

関数 $\phi(\cdot)$ は 1 と 0 からなるバイナリコードを出力するが関数 sign$(\cdot)$ の微分値は 0 になるため誤差逆伝播法による学習ができない．そこで上位から伝搬してきた微分値をそのまま下位の層に伝搬する Straight-Through Estimator (STE) [8] を適用する．

3.2 コード長を可変とするゲート機構

単語ベクトルを Encoder を通してバイナリコードに変換した後，入力単語に適切な長さまでバイナリコードを消去するゲート機構 g を適用する．

$$g = \mathrm{cumsum}(\alpha) = \sum_{i \leq m}^{m} \alpha_i \tag{3}$$

α はある 1 つの要素のみ 1 で他の要素が 0 な one-hot ベクトルであり，cumsum$(\cdot)$ は累積和の関数である．α において 1 がたつ要素の要素番号がその単語のコード長を表している．微分可能な形で one-hot ベクトルを学習するために Gumbel Softmax [9] を用いて，α を以下のように計算する．

$$\begin{aligned} \alpha_i &= \mathrm{softmax}_\tau(\log \beta_i + G_i) \\ &= \frac{\exp((\log \beta_i + G_i)/\tau)}{\sum_{j=1}^{m} \exp((\log \beta_j + G_j)/\tau)} \end{aligned} \tag{4}$$

式 4 の G_i は Gumbel 分布 $-\log(-\log(\mathrm{Uniform}[0,1]))$ からサンプリングされるノイズであり，τ は温度パラメタである．β は単語ベクトルを入力とする 2 層のニューラルネットワーク f$(\cdot)$ によって計算する．

$$\beta = \mathrm{f}(x) = \mathrm{softplus}(W \tanh(W'x + b') + b) \tag{5}$$

$W \in \mathbb{R}^{d \times d}$ と $W' \in \mathbb{R}^{d \times d}$，$b$，$b'$ は学習パラメタである．実験では $\tau = 0.1$ とした．

3.3 目的関数

単語ごとにバイナリコードの長さを可変とするモデルの学習によって，目的タスクにおける重要語には長いコードを，非重用語には短いコードを割り当てる．このとき，コーパスによってはすべての単語のコード長が最大長に収束するおそれがあるため，コード長に関する制約項を加えた．目的タスクにおける学習の目的関数を以下の式 6 に示す．

$$L = L_{task} + \lambda_2 \frac{1}{m} \sum_{i=0}^{m} g_i \tag{6}$$

表 1　各タスクに用いたコーパスのサイズ．

コーパス名	語彙サイズ	メモリサイズ (MB)
SST5 [10]	17k	20.5
TREC [11]	8.3k	8.69
IMDB [2]	60k	63.4
AG-NEWS [12]	61k	63.9
DBpedia [12]	228k	239.4

式 6 の第一項 L_{task} は各タスクにおける目的関数である．第二項はコード長に関する制約項である．λ_2 は定数であり，実験では 0.0001 とした．

先行研究 [6] から事前学習の導入によって目的タスクにおけるモデル精度が向上することが明らかになっている．本研究では再構成誤差に基づく事前学習をした後に目的タスクにパラメタを適応させた．事前学習の目的関数を以下の式 7 に示す．

$$L_{pre} = \frac{1}{m}\sum_{i=0}^{m}(x_i - \hat{x_i})^2 + \frac{\lambda_1}{2}\|W_{dec}W_{enc} - I\|^2 \quad (7)$$

λ_1 は定数であり，実験では 0.01 とした．式 7 の第一項は入力単語ベクトルの再構成誤差，第二項はバイナリコードの各要素が異なる観点の情報を Encode し，冗長性を減らすような正則化項である．事前学習のとき W_{dec} と W_{enc} は同じパラメタとなるようパラメタを共有した．また，事前学習時はモデルからゲート機構を取り除いた．

4　実験

4.1　実験設定

再構成誤差による事前学習と目的タスクにおける fine-tuning の 2 ステップの学習を行った．コードの最大長 m は 32，64，128 の 3 種を比較した．異なるコード長に設定された 4 つの AutoEncoder を用いる AdaComp [6] については (**128**,64,32,16)，(**64**,32,16,8)，(**32**,16,8,4) の 3 種を比較手法とした．

コードの事前学習　300 次元の単語ベクトル GloVe [1] を使用した．目的タスクにおける fine-tuning を前提としていない手法と公平な比較を行うため，実験では目的タスクに fine-tuning した GloVe を学習済みの単語ベクトルとして用いた．学習には Adam を使用し学習率は 0.001，バッチサイズは 64 とした．

タスク適応したコードの学習　評価に用いたタスクを表 1 に示す．学習には GloVe 中に存在する語彙のみを使用した．学習の高速化のため DBpedia [12] は単語ベクトルの平均を，それ以下のタスクでは 2 層からなる LSTM の各タイムステップ出力の平均を文ベクトルとした．LSTM のユニット数は 450，識別層として 1 層のニューラルネットワークを使用した．学習には Adam を使用し学習率は DBpedia のみ 0.0001，それ以外のタスクには 0.001 を用いた，バッチサイズは 64 とした．

1）　http://nlp.stanford.edu/data/glove.6B.zip

4.2　実験結果

最大バイナリコード長を変えて実験したときの識別精度と圧縮率を表 2 に示す．表 2 より，DBpedia 以外のタスクにおいタスクで圧縮前の単語ベクトル GloVe finetune よりも圧縮語の単語ベクトルの方がタスクの識別精度が高い．同様の結果は先行研究 [4, 5, 6] でも報告されており，強い正則化による効果だと考察されている．コードブック (Decoder) の重み共有により低頻度語の学習が簡単になった点，識別タスクにおいて区別する必要のない単語 (Tom と Mary 等) がバイナリ化によってより近いベクトルとなり後段の LSTM の学習が容易になった点が理由として考えられる．DBpedia は多くの学習データを用いた学習が行えるため，正則化が必要なかったと考えられる．小規模コーパスにおけるモデルの学習において過学習を防ぐ正則化は重要であり，単語ごとにコード長を可変とする提案法はこの正則化の効果をタスクの非重要語に対してより強めるものである．

コード長を可変とする手法 (AdaComp，Ours) はコード長固定の手法 Bin よりも高い精度を示した．特に提案法は AdaComp よりも精度が高く，圧縮率も高くなった．次の 2 つの点から精度が向上したと考えられる．1 つ目はより多様なコード長を選択できる点である．AdaComp はコード長ごとに AutoEncoder を用意するために，コード長の種類を増やすとパラメタが増えて学習が難しくなる．提案法はパラメタ数はそのままに最大コード長と同じ数だけコード長の種類を用意できるため，より細かく冗長性を排除できる．2 つ目は重み共有により学習が容易になった点である．異なるコード長を持つ単語間での重み共有により低頻度語学習の効率化が精度向上につながったと考えられる．DBpedia コーパスにおいて，コード長を可変とする提案法は Bin と比較して圧縮率が大きく向上した．語彙サイズが大きいコーパスほど，多くの非重要語が出現するためより多くのパラメタを消去できためである．

表 2　各タスクにおけるバイナリコードを用いる手法の実験結果.

	SST5		TREC		IMDB		AG NEWS		Dbpedia	
	acc	ratio	acc	ratio	acc	ratio	acc	ratio	acc	ratio
GloVe finetune	42.6	×1	91	×1	89.2	×1	91.7	×1	98.5	×1
Bin (32) [5]	38.8	×181	89.4	×132	86.9	×245	91	×253	96.7	×288
Bin (64) [5]	39.1	×95	90.8	×66	87.7	×120	91.6	×122	97.3	×139
Bin (128) [5]	40.1	×48	91.4	×35	87.9	×60	91.8	×61	97.6	×69
AdaComp (32) [6]	42.5	×135	92.4	×88	87.4	×226	91.3	×212	97	×431
AdaComp (64) [6]	43	×69	93.2	×45	88.2	×115	91.9	×102	97.3	×139
AdaComp (128) [6]	43.6	×35	92.6	×23	88.6	×57	91.9	×53	97.7	×67
Ours (32)	42.1	×198	93.2	×141	88	×317	91.6	×269	97.4	×1085
Ours (64)	43.6	×103	93.6	×74	88.6	×166	92.1	×130	97.8	×309
Ours (128)	42.5	×55	93.6	×38	89	×75	92.2	×69	97.8	×138

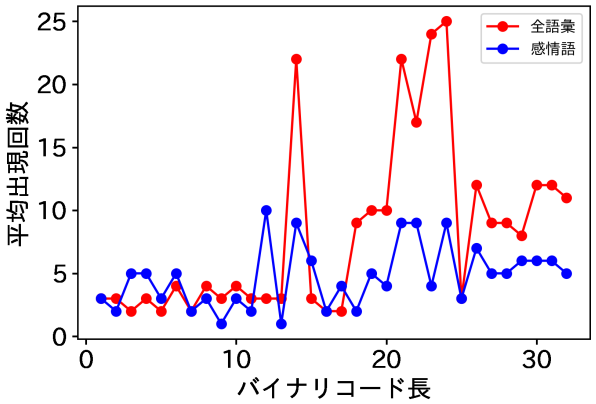

図 2　各コード長が割り当てられた語彙と感情語の平均出現回数.

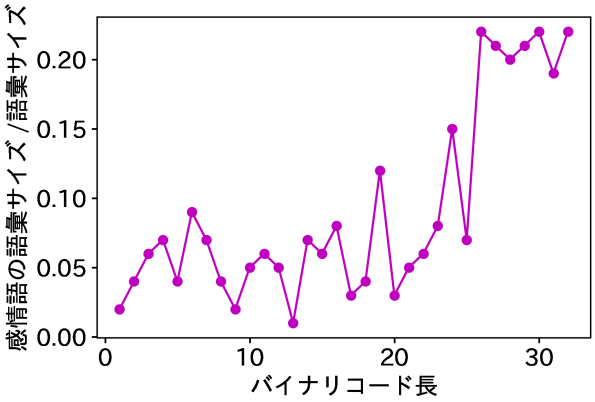

図 3　各コード長が割り当てられた語彙のうち，感情語の比率.

5　コード長の分析

提案法を学習した結果得られたバイナリコードについて，2 つの分析をした．提案モデルの学習には SST5 コーパスを用いた．

コード長と語彙の出現回数　各コード長が割り当てられた語彙と感情語について，SST5 コーパスにおける出現回数の平均を図 2 に示す．感情語は感情語辞書 [13] に含まれている語のうち SST5 コーパスに出現した語を使用した．図 2 より，出現回数の多い語には長いコードが割り当てられる傾向があるが，出現回数の多い語がコード長 14 や 20 から 24 あたりに特に多く存在していることがわかった．個々のバイナリコードを見ると"the"や"on"のように文の感情極性に依存しないが出現回数の多い語がコード長 14 や 20 から 24 に多く存在していた．対して，コード長ごとの感情語の平均出現回数の違いに傾向は確認できなかった．

コード長と感情語の比率　各コード長が割り当てられた語彙のうち，その語が感情語である比率を図 3 に示す．長いコードほど感情語の比率が高くいことがわかった．提案法によって感情語のようなタスクにおける重要語に長いコードが割り当てられる．

6　おわりに

本研究では単語分散表現の圧縮を目的として，単語ごとのパラメタ数を可変とするゲート機構を提案した．提案法はモデルの精度を落とすことなく従来法を上回る圧縮率を達成した．感情分析タスクにおいて感情語に多くのパラメタ (長いコード) が割り当てられていたことから，提案するゲート機構の導入によって，期待通りにタスクにおける重要語に多くのパラメタを割り振られ非重要語には少ないパラメタを割り振られることが分かった．

参考文献

[1] Maximilian Lam. Word2bits - quantized word vectors. *arXiv:1803.05651v3*, 2018.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, 2011.

[3] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.

[4] Raphael Shu and Hideki Nakayama. Compressing word embeddings via deep compositional code learning. In *ICLR*, 2018.

[5] Julien Tissier, Christophe Gravier, and Amaury Habrard. Near-lossless binarization ofword embeddings. In *AAAI*, 2019.

[6] Yeachan Kim, Kang-Min Kim, and SangKeun Lee. Adaptive compression ofword embeddings yeachan. In *ACL*, 2020.

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word rep- resentation. In *EMNLP*, 2014.

[8] Geoffrey E. Hinton. Neural networks for machine learning.coursera, video lectures. 2013.

[9] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[10] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[11] Ellen M Voorhees and Dawn M Tice. The trec-8 question answering track evaluation. In *TREC*, 1999.

[12] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[13] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD*, 2004.