深層コード学習による単語分散表現の圧縮

朱 中元　中山 英樹
東京大学　大学院情報理工学系研究科
{shu, nakayama}@nlab.ci.i.u-tokyo.ac.jp

平成 30 年 1 月 17 日

1　序論

ニューラルネットワークを用いた自然言語処理において，単語のベクトル表現（分散表現）は重要な役割を果たしている. しかし，従来の単語分散表現の問題として各単語に独立したベクトルが割り当てられるため，パラメータの数が膨大になるという点が挙げられる. 例えば，各ベクトルを 500 次元とすると，20 万個の語彙を表すために約 1 億個ものパラメータが必要になる. 実際，本稿で扱う簡単な感情分析モデルでは，単語ベクトルのパラメータ数は全パラメータ数の 98.8 %を占める.

この研究では，モデルの性能を下げずに，単語ベクトルで使用するパラメータ数を減らすことを目的とする. 既存研究によって，ニューラルネットワークモデルのパラメータには冗長性があることが既に分かっている. 本研究では，単語ベクトルはそれぞれ独立に学習されるため，非常に冗長性を含んでいると考える. 例えば，単語 “dog” と “dogs” はほぼ同じ意味を持ち，形態的にしか違いはない. この 2 つの単語を効率的に表現するためには，“dog” と “dogs” のベクトル表現の殆どの部分を共有するのが望ましい. 当然違いを捉えるために，それぞれのベクトルの一部は異なっている必要がある.

単語の分散表現を部分的に共有させるために，本研究では各単語 w に単語 ID の代わりに，コード $C_w = (C_w^1, C_w^2, ..., C_w^M)$ を割り当てる. 各サブコード C_w^i は $[1, K]$ に属する整数である. 理想的には，意味が近い単語同士は類似するコードが割り当てられる. 例えば，$C_{\text{dog}} = (3, 2, 4, 1), C_{\text{dogs}} = (3, 2, 4, 2)$ は良いコードの一例である. このように，本研究では各単語ごとに異なった分散表現を学習する代わりに，各サブコードの分散表現を学習する.

具体的には，モデルの中で M 個のコードブックを持ち，各コードブックは K 個の基底ベクトル（コードワード）を持つ. ある単語の分散表現を計算する際には，その単語の各サブコードに対応する基底ベクトルを次のように足し合わせる：

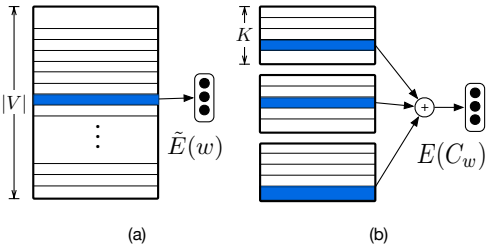

図 1: (a) 従来の単語分散表現の求め方 (b) 提案手法（合成的コードによって単語ベクトルを構築する）

$$E(C_w) = \sum_{i=1}^{M} E_i(C_w^i), \tag{1}$$

ここで，$E_i(C_i^w)$ は単語コード C_w の i 番目のサブコードのベクトル表現である. 図 1に示すように，単語をエンコードすることによって，語彙数によらずモデルが保持する必要のある基底ベクトルの数は常に $M \times K$ である. 実験の結果，基底ベクトルの数は 512 ∼ 1024 個で十分であることが分かっている. この数字は語彙数よりも遥かに少なく，高い圧縮率を期待できる.

次に，各単語のコードと基底ベクトルの学習方法を考案する. 圧縮したい分散表現 (ベースライン) を $\tilde{E}(w)$ とする. 最も素直な方法としては，$\tilde{E}(w)$ と距離が近い単語ベクトルが構築されるように基底ベクトル $\hat{E}$ とコードの集合 $\hat{C}$ を最適化することが考えられる：

$$(\hat{C}, \hat{E}) = \underset{C,E}{\operatorname{argmin}} \frac{1}{|V|} \sum_{w \in V} ||E(C_w) - \tilde{E}(w)||^2 \tag{2}$$

ここで，$|V|$ は語彙数を表す. ベースラインの分散表現としては，word2vec や GloVe による学習済みの単語分散表現を想定する. しかし，各単語のコード C_w は離散値であるため，通常直接的に式 (2) を最適化することは難しい. そこで，本研究では Gumbel-softmax

トリック [8] を利用し，ニューラルネットワークを用いて直接的に損失関数を最小化する方法を提案する．

2 関連研究

ニューラルネットワークを圧縮する既存手法としては，低精度計算 [11]，量子化 [3, 4]，枝刈り法 [5]，知識蒸留法 [6] の大きく 4 種類が存在する．

近年では，モデルを圧縮するために，特に量子化と枝刈り法が多く使われている．量子化は，少数の真のパラメータのプールを用意し，このプールの中のパラメータを重み行列の複数の位置に割り当てる手法である．代表的な手法として，HashNet[3] と DeepCompression [4] が挙げられる．本研究は重み行列を行単位で量子化する手法として解釈できる．

枝刈り法は重み行列をスパースにするための方法である．代表的な手法である Iterative Pruning[5] は絶対値の小さいパラメータを強制的にゼロにする手法である．近年，この手法は機械翻訳モデルにも適用され [10, 12]，約 80 %の圧縮率が報告されている．本研究では，Iterative Pruning をベースラインとして，実験で比較対象とした．

3 提案手法

3.1 合成的コーディング

ここまでで説明した複数のコードブックを用いるコーディングアプローチは，合成的コーデング或いはマルチコードブックエンコーディングと呼ばれる．例えばコードブックが M 個存在し，各コードブックに K 個の基底ベクトルが含まれる場合，コードの長さは $M \log K$ ビットになる．この性質によって，通常のバイナリーコードと比べて，同数の基底ベクトルの組合せを表すのに，合成的コードでは必要なビット数がより少ないというメリットを持つ．

3.2 Gumbel-Softmax によるコード学習

圧縮したい単語ベクトルの重み行列を $\tilde{\mathbf{E}} \in \mathbb{R}^{|V| \times H}$ とする．ここで H は各単語ベクトルの次元数である．式 (2) で定義した損失関数を最小化することは，実際には近似行列分解 $\tilde{\mathbf{E}} \approx \sum_{i=0}^{M} \boldsymbol{D^i A_i}$ を解くことに等しい．ここで，$\boldsymbol{A_i} \in \mathbb{R}^{K \times H}$ は基底ベクトルで構成される基底行列である．$\boldsymbol{D^i}$ は $|V| \times K$ 行列であり，各行ベクトルは K 次元の one-hot ベクトルである．したがって，単語 w の i 番目のサブコードに対応する one-hot ベクトルを $\boldsymbol{d_w^i}$ にすることで，単語ベクトルの計算式

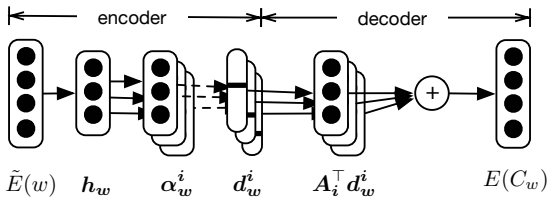

図 2: コード学習するためのニューラルネットワーク構造（Gumbel-softmax の計算は点線で示されている）

は以下の形になる：

$$E(C_w) = \sum_{i=0}^{M} \boldsymbol{A_i^\top d_w^i}. \tag{3}$$

よって，コードを学習する問題は one-hot ベクトルの集合 $\boldsymbol{d_w^1}, ..., \boldsymbol{d_w^M}$ と基底ベクトルの辞書 $\boldsymbol{A_1}, ..., \boldsymbol{A_M}$ を学習する問題に帰着する．ここで，one-hot ベクトルを学習するために，Gumbel-Softmax トリック [8] を用いて，$\boldsymbol{d_w^i}$ を以下のように計算する：

$$\left(\boldsymbol{d_w^i}\right)_k = \mathrm{softmax}_\tau(\log \boldsymbol{\alpha_w^i} + G)_k \tag{4}$$

$$= \frac{\exp((\log\ (\boldsymbol{\alpha_w^i})_k + G_k)/\tau)}{\sum_{k'=1}^{K} \exp((\log\ (\boldsymbol{\alpha_w^i})_{k'} + G_{k'})/\tau)}. \tag{5}$$

式 (5) の G_k 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 softmax の温度とする．本研究では，ベクトル $\boldsymbol{\alpha_w^i}$ を簡単な多層ニューラルネットワークによって計算する：

$$\boldsymbol{\alpha_w^i} = \mathrm{softplus}(\boldsymbol{\theta_i'}^\top \boldsymbol{h_w} + \boldsymbol{b_i'}), \tag{6}$$

$$\boldsymbol{h_w} = \tanh(\boldsymbol{\theta}^\top \tilde{E}(w) + \boldsymbol{b})\,. \tag{7}$$

ここで，隠れ層 $\boldsymbol{h_w}$ のユニット数は常に $MK/2$ である．実験では，温度は $\tau = 1$ で固定する．図 1 に示すように，コード学習のモデルは実際には一種の auto-encoder として見なせる．このモデルは合計 5 種類のパラメータを持つ（$\boldsymbol{\theta}, \boldsymbol{b}, \boldsymbol{\theta'}, \boldsymbol{b'}, \boldsymbol{A}$）．コード学習のモデルが学習されると，各単語のハッシュコード C_w は中間層 $\boldsymbol{d_w^1}, ..., \boldsymbol{d_w^M}$ それぞれに argmax を適用することで簡単に得られる．学習したパラメータ $\boldsymbol{A_1}, ..., \boldsymbol{A_M}$ は基底行列としてそのまま使われる．

4 実験

本章では，本稿の提案手法による圧縮率を検証する．自然言語処理の典型的なタスク（感情分析及び機械翻訳）において，モデルの性能を悪化させずに達成できる最大圧縮率を評価した．単語の分散表現のサイズを測る際，numpy のダンプファイルのサイズを報告するが，Iterative Pruning によって得られたスパース行列については scipy.csc_matrix で保存したファイルのサイズを報告する．

	ベクトル数	ベクトルサイズ	コード長	コードサイズ	合計サイズ	スコア
感情分析タスク / スコア：判別精度						
GloVe baseline	75102	78 MB	-	-	78 MB	87.18
prune 80%	75102	21 MB	-	-	21 MB	86.25
16×32 coding	512	0.52 MB	80 bits	0.71 MB	**1.23 MB**	87.37
32×16 coding	512	0.52 MB	128 bits	1.14 MB	1.66 MB	87.80
64×8 coding	512	0.52 MB	192 bits	1.71 MB	2.23 MB	**88.15**
独英機械翻訳タスク / スコア：**BLEU**(%)						
baseline	40000	35 MB	-	-	35 MB	29.45
prune 90%	40000	5.21 MB	-	-	5.21 MB	29.34
prune 95%	40000	2.63 MB	-	-	2.63 MB	28.84
32×16 coding	512	0.44 MB	128 bits	0.61 MB	1.05 MB	29.04
64×16 coding	1024	0.89 MB	256 bits	1.22 MB	**2.11 MB**	**29.56**
英日機械翻訳タスク / スコア：**BLEU**(%)						
baseline	80000	274 MB	-	-	274 MB	37.93
prune 90%	80000	41 MB	-	-	41 MB	38.56
prune 98%	80000	8.26 MB	-	-	8.26 MB	37.09
32×16 coding	512	1.75 MB	128 bits	1.22 MB	**2.97 MB**	38.10
64×16 coding	1024	3.50 MB	256 bits	2.44 MB	5.94 MB	**38.89**

表 1: 感情分析タスクと機械翻訳タスクにおける実験結果

4.1 コード学習

各単語のコードと基底ベクトルの辞書を学習するために，まず各タスクにおいて圧縮対象となる単語分散表現を用意する. 感情分析タスクではGloVe，機械翻訳では学習済みのニューラル機械翻訳モデルのEmbedding層を利用する. 図 2に示すコード学習モデルを訓練する際には，圧縮対象の分散表現の行列からランダムに単語ベクトルのバッチをサンプリングし，式 (2) の損失関数を最適化することで学習を行う. バッチサイズは 128 とした. パラメータの更新アルゴリズムとしてAdam を用い，学習率は 0.0001 として，20 万イタレーション実行した.4 つの GPU(GTX 1080Ti) を使用する場合，実時間では約 15 分で学習が可能である.

4.2 感情分析タスク

データセット: 本実験は IMDB 映画レビューデータセット [7] を用いる. 学習データとテストデータはそれぞれ 25,000 個のレビューテキストから構成される. 前処理として *nltk* で形態素解析し，小文字化した後，最初の 400 単語を残した. 圧縮対象の分散表現は，42B トークンの Common Crawl データセットで学習された GloVe の 300 次元の単語ベクトルを IMDB データセットに含まれる 7.5 万語彙に絞ったものである.

モデル構造: Embedding 層を除いて，本実験の全てのモデルは同一の計算グラフを持つ. モデルは一層の LSTM と，ロジスティック回帰のための softmax 層で構成される.LSTM 層のユニット数は 150 とした. ベースラインモデルの Embedding 層のパラメータ数は $75,000 \times 300$ である. 提案手法を適用した場合，32×16 の合成的コードを使用するならば，Embedding 層のパラメータ数は 512×300 になる.Iterative Pruning の場合，枝刈りしたスパース行列はモデルの学習と同時にチューニングされる.

モデル学習: すべてのモデルは Adam（学習率 0.0001）を使って 15 エポック訓練する.

実験結果: 様々な M と K の組合せにおける識別性能と圧縮率を表 1に示す. ベースラインモデルの精度は 87.16%であるが，Embedding 層では約 78MB を消費する.Iterative Pruning を使用した場合，圧縮率を 80%以上にするとモデルの性能が悪化することがわかる. 提案手法については，16×32 のコードを使用する場合，98.4%のモデル圧縮率を達成できた. 圧縮率を少し犠牲にした場合，強い正則化の効果によって，モデルの識別性能は約 1%向上することが確認された.

4.3 機械翻訳タスク

データセット: このタスクでは，IWSLT 2014 独英翻訳タスク [2] 及び ASPEC 英日翻訳タスク [9] で実験を行う. 独英翻訳タスクの前処理では，moses toolkit によって形態素解析を行った後，語彙数が 2 万になるように byte-pair encoding (以下 BPE) を適用した. 評価尺度としては *tokenized BLEU* を報告する. 英日翻訳では，先頭の 1,500 万対訳ペアを学習データとして用いた. 英語側を moses, 日本語側を kytea で前処理を行い，語彙数が 4 万になるように BPE を適用した. 評価については，このタスクの公式サイトで提供されている後処理の Kytea ベースのスクリプトを用いた.

モデル構造: ニューラル機械翻訳モデル（以下，NMT）の性能を向上させるための複数の手法を用いた.NMT のエンコーダ側は Bahdanau ら [1] と同様に標準的な双方向 LSTM によって構築し，デコーダ側は 2 層の LSTM によって構築した. またデコーダ LSTM には Residual Connection を適用した.LSTM のユニット数については，独英翻訳タスクでは 256，英日翻訳タスクでは 1000 とした.LSTM 層内部の計算を除き，すべての層の前にドロップ率 0.2 の Dropout を適用した. デコーダ LSTM の 1 層目においては Key-Value Attention を用いた.Attention 層のキーとして線形変換したエンコーダ隠れ状態を用いた.

モデル学習: すべてのモデルは Nesterov 勾配降下法によってパラメータの更新を行い，初期学習率は 0.25 とした.7,000 イタレーション毎に検証用データセットでモデルの性能を測り，パラメータを保存した. 学習率のスケジューリングについては，3 回の検証でモデルの性能の向上が観測されない場合，学習率を 10%まで減少させた. 学習の高速化のために 4GPU での分散学習を行い，各 GPU は 16 バッチを学習する.

実験結果: 翻訳タスクの結果を表 1の後半に載せる. 全ての翻訳結果の出力にはビームサーチ（ビーム幅 5）を用いた. 枝刈り法では，英日翻訳タスクにおいて翻訳性能を維持したまま 90%の圧縮率を達成できるが，独英翻訳では性能の悪化が観測された. 一方，提案手法を用いた場合には，それぞれのタスクで性能を悪化させずに 94%，99%の圧縮率を達成できた. 感情分析タスクと同様に，圧縮率を僅かに犠牲にすることで翻訳性能を向上させることができる. ここで，Embedding 層の圧縮後もリカレント層と Softmax 層の影響で NMT モデルのサイズは依然として膨大であることに注意が必要である. それらの部分を圧縮するためには，枝刈り法を適用した既存研究 [12] をなどが有効であると考えられ，参考にされたい.

5 結論

本研究では，単語ベクトル間で部分的に情報を共有させ，単語分散表現の冗長性を低減させるアプローチを提案した. 実験結果から，提案手法はモデルの性能を悪化させずに，感情分析タスクでは 98%，機械翻訳タスクでは 94%の圧縮を達成できることが確認できた. 本論文の主な貢献は，Gumbel-softmax トリックを用いて，ニューラルネットワークによるハッシュコードと基底ベクトル辞書の同時学習手法を提案したことである. 本研究では，単語分散表現の圧縮のみを対象としたが，提案手法は Softmax 層の重みを含むその他の重み行列の圧縮にも適用が可能であると考える.

参考文献

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[2] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, 2014.

[3] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *ICML*, 2015.

[4] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[5] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, Vol. abs/1506.02626, , 2015.

[6] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, Vol. abs/1503.02531, , 2015.

[7] 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.

[8] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, Vol. abs/1611.00712, , 2016.

[9] Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. Aspec: Asian scientific paper excerpt corpus. In *LREC*, 2016.

[10] Abigail See, Minh-Thang Luong, and Christopher D. Manning. Compression of neural machine translation models via pruning. In *CoNLL*, 2016.

[11] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, Vol. 1, p. 4, 2011.

[12] Xiaowei Zhang, Wei Chen, Feng Wang, Shuang Xu, and Bo Xu. Towards compact and fast neural machine translation using a combined method. In *EMNLP*, 2017.