COMPOSITIONAL CODING CAPSULE NETWORK WITH K-MEANS ROUTING FOR TEXT CLASSIFICATION

Hao Ren, Hong Lu

School of Computer Science, Fudan University
Shanghai, China

ABSTRACT

Text classification is a challenging problem which aims to identify the category of texts. Recently, Capsule Networks (CapsNets) are proposed for image classification. It has been shown that CapsNets have several advantages over Convolutional Neural Networks (CNNs), while, their validity in the domain of text has less been explored. An effective method named deep compositional code learning has been proposed lately. This method can save many parameters about word embeddings without any significant sacrifices in performance. In this paper, we introduce the Compositional Coding (CC) mechanism between capsules, and we propose a new routing algorithm, which is based on k-means clustering theory. Experiments conducted on eight challenging text classification datasets show the proposed method achieves competitive accuracy compared to the state-of-the-art approach with significantly fewer parameters.

Index Terms— Text Classification, Machine Learning, CapsNet, K-means Routing, Compositional Code

1. INTRODUCTION

Recurrent Neural Networks (RNNs, [1], [2], [3]), including Long Short Term Memory networks (LSTMs, [4], [5]) and Gated Recurrent Units (GRUs, [6], [7]), have been increasingly applied to many problems in Natural Language Processing (NLP). Text classification is one of the most basic and important tasks in this field.

However, NLP models often require a massive number of parameters for word embeddings, resulting in a large storage or memory footprint. To reduce the number of parameters used in word embeddings without hurting the model performance, the work by Shu et al. [8] proposed to construct the embeddings with few basis vectors. For each word, the composition of basis vectors is determined by a hash code.

On the other hand, Hinton et al. [9] presented capsule, which is a small group of neurons. The activities of neurons are used to represent the various properties of an entity. The work by Sabour et al. [10] applied this concept to neural network firstly, a novel routing algorithm called dynamic routing was adopted to select active capsules. The experiments of CapsNet showed capsules could learn a more robust representation than CNNs in image classification domain.

In this paper, we aim to reduce the number of parameters used in word embeddings by introducing compositional coding for text classification, while maintaining a competitive accuracy compared to the state-of-the-art approach. To do so, we apply CapsNet to the classification of texts, and propose a novel and robust routing algorithm named k-means routing to determine the connection strength between lower-level and upper-level capsules.

Our compositional coding significantly differs from the method proposed by Shu et al. [8]. The work by Shu et al. [8] selects exclusive codeword vector in each codebook, while our method uses all the codeword vectors in each codebook to form the word embedding. To distinguish with the Compositional Coding embedding (CC embedding) method by Shu et al. [8], we call our compositional coding method as Compositional Coding capsule (CC capsule). And our k-means routing uses cosine similarity to obtain the coupling coefficient between lower-level and upper-level capsules, while the dynamic routing proposed by Sabour et al. [10] uses dot product value to determine the coupling coefficient. Furthermore, the coefficient update strategies are also different.

The main contributions of this work are three-folds. First, we propose a new compositional coding approach for constructing the word embeddings with significantly fewer parameters. Second, we propose a novel routing method named k-means routing to decide the credit attribution between lower-level and upper-level capsules, it is more stable and robust than dynamic routing. Third, we construct an end-to-end CapsNet with Bidirectional Gated Recurrent Units (BiGRUs, [11], [12]) to text classification and achieve comparable results to the state-of-the-art method.

2. PROPOSED METHODS

In this section, we describe the compositional coding approach, k-means routing algorithm and the end-to-end model in details.
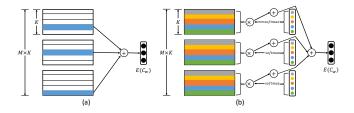
Fig. 1. Comparison of embedding computations between the CC embedding layer (a) and CC capsule layer (b) for constructing embedding vectors.

2.1. Compositional code capsule layer

Unlike the CC embedding layer, which selects exclusive codeword vector in each codebook, CC capsule layer uses all codeword vectors in each codebook to form the word embedding. And CC embedding layer restricts code must be integer number, our CC capsule layer eliminates this limitation.

Suppose the vocabulary size is $|V|$, we create M codebooks $\boldsymbol{E}_1, \boldsymbol{E}_2, ..., \boldsymbol{E}_M$, each containing K codeword vectors. For CC embedding layer, the embedding of a word w is computed by summing up the codewords corresponding to all the components in the code as

$$\boldsymbol{E}(\boldsymbol{C}_w) = \sum_{i=1}^{M} \boldsymbol{E}_i(\boldsymbol{C}_w^i) \tag{1}$$

where $\boldsymbol{E}_i(\boldsymbol{C}_w^i)$ is the $\boldsymbol{C}_w^i$-th codeword in the codebook $\boldsymbol{E}_i$. For CC capsule layer, the embedding of word w is computed by summing up the weighted codewords corresponding to all the components in the code as

$$\boldsymbol{E}(\boldsymbol{C}_w) = \sum_{i=1}^{M} \sum_{j=1}^{K} \underset{j}{softmax}(\boldsymbol{C}_w^{ij}) \boldsymbol{E}_i(j) \tag{2}$$

where $\boldsymbol{E}_i(j)$ is the j-th codeword in the codebook $\boldsymbol{E}_i$, $\boldsymbol{C}_w^{ij}$ is the j-th code for the codebook $\boldsymbol{E}_i$. From the Formula (2), we can see the code needn't to be integer number. Fig.1 gives an intuitive comparison between the CC embedding layer and the CC capsule layer.

Moreover, M and K are hyper-parameters, they are designated by user in the CC embedding layer. However, in the CC capsule layer, only M need to be designated, K is determined as follows

$$K = \lceil \sqrt[M]{|V|} \rceil \tag{3}$$

because K^M is the total number of all the combination of codeword vectors, it makes sure $K^M \geq |V|$, which means each word can be assigned with an unique combination of codeword vectors.

2.2. K-means routing

The Fully Connected (FC) capsule layer receives lower-level capsules, which represent low-level features, then the rout-

ing algorithm clusters the low-level features to high-level features. We know that k-means clustering is an efficient method to cluster features, and produce a cluster centroid by using all the clustered features. Based on this, we propose k-means routing. We regard the k-means routing algorithm between l^{th} layer's capsules and $(l + 1)^{th}$ layer's capsules as a k-means clustering process. The $(l + 1)^{th}$ layer's capsules are cluster centers of the lth layer's capsules.

Therefore, we briefly review k-means clustering and its optimization procedure. Given n capsules $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ and the metric d, k-means clustering is to find k cluster centers $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ to minimize the following loss function:

$$L = \sum_{i=1}^{n} \min_{j=1}^{k} d(\boldsymbol{u}_i, \boldsymbol{v}_j) \tag{4}$$

we use the following metric:

$$d(\boldsymbol{u}_i, \boldsymbol{v}_j) = -\left\langle \frac{\boldsymbol{u}_i}{\|\boldsymbol{u}_i\|}, \frac{\boldsymbol{v}_j}{\|\boldsymbol{v}_j\|} \right\rangle \tag{5}$$

here $\langle \cdot, \cdot \rangle$ is the scalar product operation. For obtaining $\boldsymbol{v}_j$, we need to solve the equations $\partial L / \partial \boldsymbol{v}_j = 0$, which is non-linear mostly and can not be solved analytically. So we introduce an iterative process, suppose $\boldsymbol{v}_j^{(r)}$ is the result of $\boldsymbol{v}_j$ after r iterations. We can simply take

$$\boldsymbol{v}_j^{(r+1)} = \sum_{i=1}^{n} c_{ij}^{(r)} \boldsymbol{u}_i \tag{6}$$

here $c_{ij}^{(r)} = \underset{j}{softmax}\left(\langle \frac{\boldsymbol{u}_i}{\|\boldsymbol{u}_i\|}, \frac{\boldsymbol{v}_j^{(r)}}{\|\boldsymbol{v}_j^{(r)}\|} \rangle\right)$, it means $\boldsymbol{v}_j^{(r+1)}$ is the sum of those nearest $\boldsymbol{u}$s to $\boldsymbol{v}_j^{(r)}$.

Finally, to achieve a complete routing algorithm, we need to solve these three problems: how to initialize the cluster centers, how to identify capsules at different position, how to guarantee the cluster centers keep the main information of input features. They all can be solved by inserting transformation matrix $\boldsymbol{W}_{ij}$:

$$\boldsymbol{v}_j^{(r+1)} = \sum_{i=1}^{n} c_{ij}^{(r)} \boldsymbol{W}_{ij} \boldsymbol{u}_i \tag{7}$$

here $c_{ij}^{(r)} = \underset{j}{softmax}\left(\langle \frac{\boldsymbol{W}_{ij} \boldsymbol{u}_i}{\|\boldsymbol{W}_{ij} \boldsymbol{u}_i\|}, \frac{\boldsymbol{v}_j^{(r)}}{\|\boldsymbol{v}_j^{(r)}\|} \rangle\right)$. For the simplicity of this iterative process, we assign the sum of $\boldsymbol{u}_i$ averagely to each cluster center as $\boldsymbol{v}_j^{(0)}$. Because we want to use the length of capsule to represent the probability that a category's entity exists, a $squash$ function has been introduced:

$$squash(\boldsymbol{v}_j) = \frac{\|\boldsymbol{v}_j\|}{1 + \|\boldsymbol{v}_j\|^2} \boldsymbol{v}_j \tag{8}$$

The whole procedure is summarized on Algorithm 1. Inserting $\boldsymbol{W}_{ij}$ is a beautiful trick, which induces different cluster centers by one same initialization method. In addition,

W_{ij} can keep the position information and increase or decrease dimension of capsule, which means the cluster centers have enough representation ability.

Algorithm 1 K-means Routing

1: **procedure** ROUTING($\boldsymbol{u}_i, r$)
2:      Initialize $\boldsymbol{v}_j \leftarrow \frac{1}{k} \sum\limits_{i=1}^{n} \boldsymbol{W}_{ij} \boldsymbol{u}_i$
3:      **for** r iterations **do**
4:          $b_{ij} \leftarrow \langle \frac{\boldsymbol{W}_{ij}\boldsymbol{u}_i}{\|\boldsymbol{W}_{ij}\boldsymbol{u}_i\|}, \frac{\boldsymbol{v}_j}{\|\boldsymbol{v}_j\|} \rangle$
5:          $c_{ij} \leftarrow \underset{j}{softmax}\, b_{ij}$
6:          $\boldsymbol{v}_j \leftarrow \sum\limits_{i=1}^{n} c_{ij} \boldsymbol{W}_{ij} \boldsymbol{u}_i$
7:      **return** $squash(\boldsymbol{v}_j)$

K-means routing is similar to dynamic routing in general, but it has three differences. First of all, we don't apply the $squash$ function to capsule $\boldsymbol{v}_j$ in the period of iteration, we just squash it after iteration. Secondly, b_{ij} is replaced by new b_{ij}, however, in dynamic routing, b_{ij} is replaced by new b_{ij} plus old b_{ij}. This is the biggest difference between our routing algorithm and dynamic routing. Finally, the cosine similarity is computed between $\boldsymbol{v}_j$ and $\boldsymbol{W}_{ij}\boldsymbol{u}_i$ instead of dot product.

According to the b_{ij} update step as described in dynamic routing, after r iterations:

$$\boldsymbol{v}_j^{(r)} \sim squash\left(\sum_i \frac{e^{r\hat{\boldsymbol{u}}_i \cdot \boldsymbol{v}_j}}{Z_i} \hat{\boldsymbol{u}}_i\right) \qquad (9)$$

which

$$Z_i = \sum_j e^{\hat{\boldsymbol{u}}_i \cdot \boldsymbol{v}_j}, \quad \hat{\boldsymbol{u}}_i = \boldsymbol{W}_{ij} \boldsymbol{u}_i \qquad (10)$$

if $r \rightarrow +\infty$, we find the result of $softmax$ will be either 0 or 1. In other words, each lower capsule is linked to sole upper capsule. This is unreasonable, we know there are common characteristics among different categories, so we hope the common characteristics can be linked to all those categories. That's why we don't plus old b_{ij} when we update b_{ij}.

2.3. Model architecture

Our architecture consists of a CC capsule layer, a BiGRU and a FC capsule layer. The task of CC capsule is to obtain the embedding of word. BiGRU extracts lower features and feeds them into FC capsule, which takes advantage of them to form the higher features and correctly classify the texts.

We design a simple model to validate the effectiveness and test the performance of our approach. The structure is illustrated in Fig.2. The number of codebook is 8, and the embedding dimension is 64. The hidden size of BiGRU is 128, and the number of recurrent layers is 2. We introduce a Dropout layer [13] on the outputs of each RNN layer except
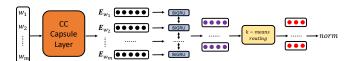

Fig. 2. Our model structure, we use 2-norm of capsule to represent the probability that a category's entity exists.

the last layer, with dropout probability equal to 0.5. The dimension of lower capsules is 8, and the dimension of upper capsules is 16. As the number of categories is related to specific dataset, so the number of output capsules in FC capsule layer is specified with particular dataset.

3. EXPERIMENTS

In this section, we conduct extensive experiments on the 8 benchmark text datasets. We first introduce the details of the 8 datasets, and present the experimental settings. Then we conduct experiments with our model and compare with the state-of-the-art methods.

3.1. Datasets

We use publicly available datasets from Zhang et al. [14] to evaluate our model. These datasets are AG's News (AG's.N), DBPedia, Yahoo! Answers (Yahoo!.A), Sogou News (Sogou.N), Yelp Review Polarity (Yelp.P), Yelp Review Full (Yelp.F), Amazon Review Polarity (Amazon.P) and Amazon Review Full (Amazon.F). Table.1 shows summary of corporas main features.

The datasets used in this paper not only contain English words, but also contains digital numbers, punctuations, Chinese words, etc. So we need to preprocess them to make sure the preprocessed samples can be tokenized by white space easily. To achieve this goal, we turn the sentences to be lower case firstly, then add white space character before and after the words or characters which are not belong to English words. Additionally, we just take the first 5000 words into our model.
For example, the original sequence of words is

> Skye Bank Skye Bank Plc. commonly known as Skye Bank is a commercial bank based in Nigeria. It is one of the twenty-six (26) commercial banks licensed by the Central Bank of Nigeria the country's banking regulator.

after preprocessing, it turns to

> skye bank skye bank plc . commonly known as skye bank is a commercial bank based in nigeria . it is one of the twenty - six (2 6) commercial banks licensed by the central bank of nigeria the country ' s banking regulator .

Table 1. Statistics of the benchmark text datasets.

Dataset	#Class	#Train	#Test	$|V|$
AG's.N	4	120,000	7,600	62,535
DBPedia	14	560,000	70,000	548,338
Yahoo!.A	10	1,400,000	60,000	771,820
Sogou.N	5	450,000	60,000	106,385
Yelp.P	2	560,000	38,000	200,790
Yelp.F	5	650,000	50,000	216,985
Amazon.P	2	3,600,000	400,000	931,271
Amazon.F	5	3,000,000	650,000	835,818

Table 2. Test set accuracy compared to other methods.

Dataset	VDCNN	W.C.region	Ours
AG's.N	91.33%	**92.89%**	92.39%
DBPedia	98.71%	**98.89%**	98.72%
Yahoo!.A	73.43%	73.66%	**73.85%**
Sogou.N	96.82%	**97.63%**	97.25%
Yelp.P	95.72%	96.39%	**96.48%**
Yelp.F	64.72%	64.90%	**65.85%**
Amazon.P	**95.72%**	95.23%	94.96%
Amazon.F	**63.00%**	60.93%	60.95%

Table 3. Model parameters compared to other methods.

Dataset	VDCNN	W.C.region	Ours
AG's.N	17,232,900	43,810,308	**2,456,800**
DBPedia	**17,362,030**	233,333,518	26,797,408
Yahoo!.A	**17,301,514**	370,613,514	37,516,352
Sogou.N	17,242,053	101,780,101	**4,713,640**
Yelp.P	17,228,978	118,065,410	**8,483,696**
Yelp.F	17,235,125	127,256,197	**9,137,640**
Amazon.P	**17,230,050**	403,850,498	45,153,616
Amazon.F	**17,236,485**	364,864,133	40,578,016

3.2. Experimental settings

We implemented our model with PyTorch library [15], all the experiments are performed on a single NVIDIA GeForce GTX 1080 GPU. The number of routing iterations is fixed to 3, and the batch size is 30. We train the model with 10 epochs. A lot of loss functions have been experimented, finally we decided using a compositional loss function of Margin loss [10] and Focal loss [16] to compute our model's loss:

$$L = \frac{1}{k} \sum_{j=1}^{k} \Big(T_j \max(0, 0.9 - \|\boldsymbol{v}_j\|)^2 + 0.5(1 - T_j) \quad (11)$$

$$\max(0, \|\boldsymbol{v}_j\| - 0.1)^2 - 0.25(1 - \|\boldsymbol{v}_j\|)^2 \log \|\boldsymbol{v}_j\| \Big)$$

where $T_j = 1$ iff a text of class j is present, it is optimized through ADAM scheme [17] with default learning rate and momentum. The code of our work is available on `https://github.com/leftthomas/CCCapsNet`.

3.3. Experimental results about test set accuracy

Our model is compared with two state-of-the-art used supervised text classification models. We report the VDCNN baselines from Conneau et al. [18] and the W.C.region of Qiao et al. [19]. The quantitative results are shown in Table.2, best results are marked with bold.

On six datasets of eight, our model and W.C.region beat VDCNN. As a result, the two methods win VDCNN on six datasets and lost in two of Amazon datasets. On the six datasets, our model and W.C.region obtain half of the best results. So we have come to the conclusion that the proposed method achieves comparable accuracy with the state-of-the-art approach.

3.4. Experimental results about model parameters

We also compare the number of parameters about our model and the two state-of-the-art methods. The experimental results are summarized in Table.3, the minimum number of parameters are marked with bold. From the results, we observe that our model and VDCNN require fewer parameters than

W.C.region on all the 8 datasets. For example, the number of parameters about Sogou.N dataset used in our model is $\sim 4.7M$, but the W.C.region needs more than $101M$ parameters, which is 21 times than ours. What's more, our model and VDCNN obtain the minimum parameters on the 4 datasets among this 8 datasets for each.

Take into account of the results about test accuracy and model parameters, we can get a corollary that VDCNN requires similar number of parameters with our method but can not obtain the similar accuracy compared to ours, the W.C.region can obtain the similar accuracy, but requires significantly more parameters than ours. Thus we can get a conclusion that our model achieves competitive accuracy compared to the state-of-the-art approach with significantly fewer parameters. It is obvious that our model has huge advantage in practical application.

4. CONCLUSION

In this paper, we proposed CC capsule layer, which required fewer parameters than traditional embedding layer. Then we developed a more robust and stable k-means routing algorithm, analyzed the limitation of dynamic routing. Extensive experiments demonstrated the effectiveness of CC capsule layer and k-means routing, our method achieved competitive performance on the task of text classification compared to the state-of-the-art methods with significantly fewer parameters.

5. REFERENCES

[1] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010, pp. 1045–1048.

[2] Jeffrey L Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533, 1986.

[4] Sepp Hochreiter and Júrgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 1999.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, Yoshua Bengio, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[7] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou, "Minimal gated unit for recurrent neural networks," *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, 2016.

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

[9] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*, 2011, pp. 44–51.

[10] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3859–3869.

[11] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[12] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[14] Xiang Zhang, Junbo Zhao, and Yann LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[15] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet, "Torch7: A matlab-like environment for machine learning," in *NIPS workshop on Big Learning*, 2011.

[16] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.

[17] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[18] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017, pp. 1107–1116.

[19] chao qiao, bo huang, guocheng niu, daren li, daxiang dong, wei he, dianhai yu, and hua wu, "A new method of region embedding for text classification," in *International Conference on Learning Representations*, 2018.