RCapsNet: A Recurrent Capsule Network for Text Classification

Junfeng Hu
IEEE Student Member
School of Big Data & Software Engineering
Chongqing University
Chongqing, China
hjf@cqu.edu.cn

Jun Liao
School of Big Data & Software Engineering
Chongqing University
Chongqing, China
liaojun@cqu.edu.cn

Li Liu*
School of Big Data & Software Engineering
Chongqing University
Chongqing, China
dcsliuli@cqu.edu.cn
* corresponding author

Wenchao Ma
School of Big Data & Software Engineering
Chongqing University
Chongqing, China
20161642@cqu.edu.cn

***Abstract*—In this paper, we propose RCapsNet, a recurrent capsule network for text classification. Although a variety of neural networks have been proposed recently, existing models are mainly based either on RNN or on CNN, which are rather limited in encoding temporal features in these network structures. In addition, most of these models require to integrate prior linguistic knowledge into them, which is not practical for a non-linguistician to handcraft such knowledge. To address these issues on temporal relational variabilities in text classification, the RCapsNet is presented by employing a hierarchy of recurrent structure-based capsules. It consists of two components: the recurrent module considered as the backbone of the RCapsNet and the reconstruction module designed to enhance the generalization capability of the model. Empirical evaluations on four benchmark datasets demonstrate the competitiveness of the RCapsNet. In particular, it is shown that prior linguistic knowledge is dispensable for the training of our model.**

***Index Terms*—recurrent capsule network; text classification; temporal feature; dynamic routing**

I. Introduction

As an important and essential natural language processing (NLP) task, text classification is to categorize a phrase, a sentence, or sentences into distinct topics of interest, such as opinions, attitudes, and emotions [1], [2]. In the past, many traditional approaches are utilized to solve this problem. They leverage expert knowledge about words by employing logistic regression [3], support vector machine [4], and naive bayes [5], among others for classification. In recent years, the deep learning methods begin to dominate in this field, which can be grouped into two types, recurrent neural networks (RNNs) [6], [7] and convolutional neural networks (CNNs) [8].

With the great success being achieved, existing deep learning methods still bear several major issues. By introducing convolutional kernels crossing features maps, the family of CNN approaches are spatially sensitive, which unfortunately leads to the limitation of addressing semantic information in time domain [9]. On the other hand, RNNs are much more capable of representing temporal variabilities by not only using hidden states to keep historic context but also incorporating either simple multiplication (vanilla RNN) or gate mechanism (LSTM). However, they tend to be not robust to the outliers that one or few word vector(s) with enormous feature values (called aberrant vector) may destroy the structure of RNN models. Although the forget gate mechanism in LSTM contributes to discard such aberrant vectors, it merely can deal with part of such issue on outliers. In addition, a RNN model typically engages one hidden vector for representing semantic information, which may be rather limited in capturing intricate semantics [10], since phrases are extremely diverse under different context. Also, in NLP tasks, a RNN model is often required to manually encode linguistic knowledge such as negation words (e.g. no, never, not) and strong words (e.g. pretty, extremely) into learning models to improve prediction performance [11]. Therefore, it strongly relies on prior expertise, which remains a challenging issue for deep learning models in practice.

Recently, capsule network (or CapsNet in short) [12] is proposed to possess a parse tree like structure, in which each layer is divided into small groups of neurons rather than a signal neuron. In CapsNet, a vector of neurons, called *capsule*, is defined as the primary element, instead of single neurons in CNN or RNN models. By employing an iterative routing process, the CapsNet can establish probabilistic agreements among parent capsules over child capsule, which compensates the model for the limitation of max polling that ignores all but the most active features. It is well known that the CapsNet has achieved excellent performance on many visual tasks [13]. In the field of NLP, Wang et al. [14] applied

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

capsules as the output layer in the RNN model after a common recurrent of LSTM. Their structure is however still a traditional recurrent module that cannot handle the inherent robustness issue of RNNs. Zhao et al. [9] proposed a novel convolutional capsule network (ConvCaps) which possesses similar structure as CNNs to text classification. By stacking three ConvCaps of different scales, the ConvCaps obtains competitive results. However, this model is still CNN-based, which lacks the capability of modeling temporal features.

Inspired by the CapsNet, we propose a novel deep learning model, called recurrent capsule network or *RCapsNet* in short, to address the aforementioned issues for text classification. Briefly speaking, this is achieved by integrating a hierarchy of capsules in recurrent neural network. More specially, to equip the model with the capability of representing temporal context, hidden capsules are incorporated to remain the historic features. In each recurrent step, new input capsules and hidden capsules are coalesced to hidden capsules of the next recurrent step. Different from the vanilla RNN and LSTM, new hidden capsules are generated and estimated through an iterative voting process, called *routing*. After each iteration, the hidden capsules become more refined and precise on text classification, which inherently surpasses the performance of gate scheme. Comparing with RNNs, the occurrence of outlier have only limited impact on our model due to the iterative voting process. Also, the process is beneficial to capsule identification, and it can discard aberrant feature capsules with small voting score after routings. In this way, the model can characterize more temporal relational variabilities of texts than previous models, particularly without any manually encoded linguistic knowledge. The main contributions of our work are summarized as follows:

- To the best of our knowledge, the RCapsNet is the first capsule-based recurrent neural network model for text classification, which can explicitly model the temporal variabilities in texts by generating a hierarchy of capsules.
- In particular, the RCapsNet addresses the major weaknesses of the family of existing CNN and RNN models for text classification, including the limitation in encoding temporal features, and the requirement of prior linguistic knowledge manually encoded into them. Empirical experiments demonstrate the superior performance of the RCapsNet over the state-of-the-art models.

II. Related work

Text classification has been tackled with by a number of machine learning algorithm, including many traditional methods mentioned above. Most of the previous methods heavily rely on feature engineering techniques such as bag-of-words [15], n-grams [16] to extract salient features or sufficient statistics as input of the machine learning methods of interest. It has been observed that the practical performance of learning models highly depends on the feature representation of the sentences and phrases involved [17]. Despite being practically very effective, obtaining effective feature representation often requires tremendous engineering effort, yet the result might not be generalizable across various datasets.

Meanwhile, the recent development of deep learning techniques provide an excited alternative in text classification, where state-of-the-art performance is obtained with much less effort in feature engineering [18]. They can be roughly categorized into two groups: RNNs [19], [20] and CNNs [9], [21]. Kim et al. [21] carried out a series of experiments using CNNs on pre-trained word vectors, which demonstrated the superiority of training model with such pre-trained vectors. Kalchbrenner et al. [8] improved CNN structures by using dynamic k-Max pooling, which can deal with input sentences of varying lengths without resorting to a parse tree. In this way, it can be readily applied to a new language. Peng et al. [22] proposed a graph-based convolutional neural model to convert texts to graph-of-words over a set of predefined graph convolution operations being adopted to learn features. However, due to the shortcoming of modeling temporal features, the family of CNN models tend to fall behind the state-of-the-arts. Subsequently, LSTM starts to become the default choice for a range of NLP tasks. Tai et al. [7] proposed a tree-based LSTMs model by utilizing the structures of sentence on different NLP tasks. Yang et al. [23] integrated LSTM or GRU with attention mechanism by introducing a hierarchical structure and their models achieve competitive results over six benchmarks. Sachan et al. [24] presented bidirectional LSTM network in both supervised and semi-supervised ways. However, LSTMs may crash in the presence of aberrant values.

To address the problems in the existing models, we introduce capsules in our network model. The idea of capsules is first designed by Hinton et al. [25], where transformation matrices (both shared and non-shared parameters) are adopted to provide networks the abilities of learning information like part-whole relationships. Sabour et al. [12] proposed a capsule network to replace single neuron by vector-valued capsule, and substitute max pooling by dynamic routing. Their result shows great potential in a number of visual tasks. For instance, the results of CapsNet surpass all the state-of-the-arts on MNIST [26] benchmark. More recently, convolutional capsule network has been developed by Zhao et al. [9] for general classification tasks, which outperforms others on four out of six text classification benchmarks. This inspires us to present the following recurrent capsule network to explicitly model the temporal variabilities for text classification, which is achieved by generating a hierarchy of capsules in recurrent neural network with temporal dependencies. In this way, we can represent words and learn their temporal features more efficiently than other existing models.

III. Background

A. Word Embedding

Given a text dataset, a sentence is often represented by a sequence of n word vectors $\mathbf{X} = [x_1, x_2, ..., x_n]$, where x_i $(1 \leq i \leq n)$ can be a one-hot vector so that its length equals the number of distinct words in the dataset. This representation is however intrinsically sparse [27]. Another option which

becomes popular nowadays is to use a pre-trained embedding corpus such as GloVe [28] to map the word x_i into a low dimension space called embedding space. In this paper, we adopt GloVe to map each word into a 300-d embedding space.

B. Capsule Network

A capsule is a collection of neurons that represent the parameters of a specific variety of entity like an object or an object part. Formally, a capsule can be represented by a vector with one element for each neuron to hold that neuron's instantiation value. A capsule network or CapsNet [12] takes multiple capsules as input, and uses *dynamic routing* (or *routing* for short) as its backbone to obtain capsules of the next layer. Finally, the number of capsules in the output layer is equal to the number of categories for classification tasks. In what follows, we focus on the concept of dynamic routing, which will be adopted in our RCapsNet model.

Given an *element vector* $\hat{u}_{j|i}$, which is generated from the i-th capsule of current l-th layer to the j-th capsule of its next layer $(l+1)$, its iterative coupling coefficient $\varphi_{j|i}$ in a routing is defined by

$$\varphi_{j|i} = \text{softmax}\left(\gamma_{j|i}, \gamma_i\right), \tag{1}$$

where $\gamma_{j|i}$ is the logits of coupling coefficients that are initialized to 0 at the first iteration. Then, the j-th capsule of the next layer, denoted by $\mathbf{o}_j$, is a weighted sum of squashed vectors $\hat{u}_{j|i}$:

$$\mathbf{o}_j = S\left(\sum_i \varphi_{j|i}\hat{u}_{j|i}\right), \tag{2}$$

where S is nonlinear squash function

$$S(z) = \frac{\|z\|^2}{1+\|z\|^2}\frac{z}{\|z\|}. \tag{3}$$

The last step of an iteration consists of updating each couple coefficient $\gamma_{j|i}$ by

$$\gamma_{j|i} = \gamma_{j|i} + \hat{u}_{j|i} \cdot \mathbf{o}_j, \tag{4}$$

where $\cdot$ refers to an inner product. All $\gamma_{j|i}$ in the current iteration can be regarded as the initial coupling coefficients of the next iteration. The whole process is repeated until the number of iterations, which can be set manually, is reached. The final outputs of a routing are all the capsules $\{\mathbf{o}_j\}$ in the last iteration. Note that the subscript $j|i$ indicates the parameter associated with the routing procedure from the i-th capsule in the current layer to the j-th hidden capsule in the next layer.

IV. Our Approach

Here we present the *RCapsNet* model, which endows capsule network with the capability of encoding temporal dependencies in sentences. We first briefly introduce the model, as illustrated in Fig. 1. The inputs of our model are obtained by mapping word vectors into an embedding space using a pre-trained corpus as described in section III-A. The RCapsNet mainly consists of a *recurrent module* and a *reconstruction module*. The recurrent module plays the backbone role in

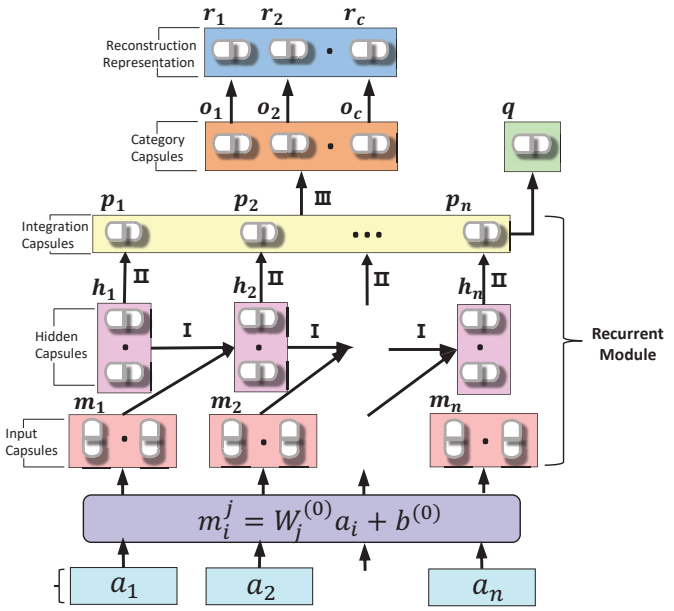

Fig. 1: **The structure of the RCapsNet model. The input sentence consists of n word, with each being represented as $a_i \in \mathbb{R}^v$ in an embedding space as described in section III-A. There are four levels of capsules, namely, input, hidden, integration and category capsules, and they are denoted by $\mathbf{m}$, $\mathbf{h}$, $\mathbf{p}$ and $\mathbf{o}$, respectively. $\mathbf{r}$ means the reconstruction representation of $\mathbf{o}$. I, II, and III refers to dynamic routing stages I, II, and III, respectively, which are responsible for generating capsules between two neighbouring levels. $\mathbf{q}$ refers to the averaged integration capsule.**

RCapsNet, which contains a hierarchy of four different levels of capsules. In the first input capsule level, each input vector multiplies a weighted tensor to attain *input capsules* (denoted by $\mathbf{m}$) for its layer. In the hidden capsule level, every layer contains multiple hidden capsules inherited from *hidden capsules* (denoted by $\mathbf{h}$) in previous layer as well as input capsules via dynamic routing stage I. In the integration capsule level, routing stage II is adopted to generate the layer of highly clustered capsules called *integration capsules* (denoted by $\mathbf{p}$). Finally, the output level (i.e. category capsule level) is composed of *category capsules* (denoted by $\mathbf{o}$), with each corresponding to a unique output class. These category capsules are generated from integration capsules in all layers via routing stage III. To improve the diverseness and robustness of RCapsNet, the reconstruction module is introduced to maintain the heterogeneous characteristics of input word vectors. Also, it helps to alleviate the outlier issue by constructing on a wide variety of features obtained from the entire sentences rather than relying on only a few salient features.

A. Embedded Input Vector

Let n be the length of a sentence, and v the dimension of a word vector. Denote a word vector by $a_i \in \mathbb{R}^v$, the v-dimensional vector representing the i-th word in a sentence. Let $\mathbf{W}^{(0)} \in \mathbb{R}^{n_c \times d \times v}$ be a 3d-tensor to generate input capsules, with n_c being the number of input capsules, and d

the dimension of the input capsule. As depicted in Fig. 1, input capsules are obtained by respectively multiplying a 3d-tensor called $\mathbf{W}^{(0)}$ and the corresponding vectors of input words, namely $a_1, a_2, ..., a_n$. The the j-th input capsule of the i-th word, denoted by $\mathbf{m}_i^j \in \mathbb{R}^d$, is obtained by

$$\mathbf{m}_i^j = \mathbf{W}_j^{(0)} a_i + b^{(0)}, \tag{5}$$

where $b^{(0)}$ is a bias term, $\mathbf{W}_j^{(0)} \in \mathbb{R}^{d \times v}$ is the j-th matrix slice of the tensor $\mathbf{W}^{(0)}$. Thus, for $j = 1, 2, ..., n_c$, a total number of n_c input capsules of dimension d for word vector i are generated and can be formalized as

$$\mathbf{M}_i = [\mathbf{m}_i^1, \mathbf{m}_i^2, ..., \mathbf{m}_i^n] \in \mathbb{R}^{d \times n_c}. \tag{6}$$

B. Recurrent Module

The capsule recurrent module plays a role similar to that of the recurrent units in RNN. Normally, RNN adopts a directed cycle layer that multiplies a hidden vector and a input vector with different parameter matrixes separately and adds them to integrate the input vector and the hidden vector, which gives networks the ability to extract dynamic temporal information from a sentence. Due to the construction of hidden vectors, RNN can record feedforward information of word sequences, making it suitable for NLP tasks. Inspired by the structure of RNN, we design a recurrent module containing capsules with dynamic routing.

Let $\mathbf{h}_l^j \in \mathbb{R}^d$ be the j-th hidden capsule in the l-th layer. We suppose that the number of input capsules is equal to the number of hidden capsules. For simplicity, we denote the set of hidden and input capsules in the l-th layer by $\mathbf{K}_l = [\mathbf{h}_l^1, \mathbf{h}_l^2, ..., \mathbf{h}_l^{n_c}, \mathbf{m}_l^1, \mathbf{m}_l^2, ..., \mathbf{m}_l^{n_c}] \in \mathbb{R}^{d \times 2n_c}$.

Normally, the capsule network uses transformation matrixes to generate element vectors and learn part-whole relationships. In particular, there are three transformation matrixes defined in our model, i.e., $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{W}^{(3)}$, corresponding to the three stages of routings, respectively. It is worth to note that we also call the transformation matrix as transformation tensor. It can be catalogued into two types, i.e., shared transformation matrix and non-shared transformation matrix. We will elucidate them in the following definitions of the three transformation matrixes.

(**Routing stage I**) Let $\mathbf{W}^{(1)} \in \mathbb{R}^{n_c \times 2n_c \times d \times d}$ be a non-share transformation tensor in routing stage I. Figure 2 illustrates the process of dynamic routing with a non-shared transformation tensor. To generate the element vector $\hat{u}_{j|i}^{(1)} \in \mathbb{R}^d$, named *vote*, from the i-th capsule in the l-th layer to the j-th hidden capsule in the next layer $l+1$, each corresponding vote can be formalized as:

$$\hat{u}_{j|i}^{(1)} = \mathbf{W}_{j|i}^{(1)} \mathbf{k}_l^i + b_{j|i}^{(1)} \tag{7}$$

where $\mathbf{k}_l^i$ is an element of $\mathbf{K}_l$, $\mathbf{W}_{j|i}^{(1)} \in \mathbb{R}^{d \times d}$ is the $j|i$-th matrix of tensor $\mathbf{W}^{(1)}$, and $b_{j|i}^{(1)}$ is a capsule bias term. So far, we can compute out the hidden capsule $\mathbf{h}$ based on the Equation (2).

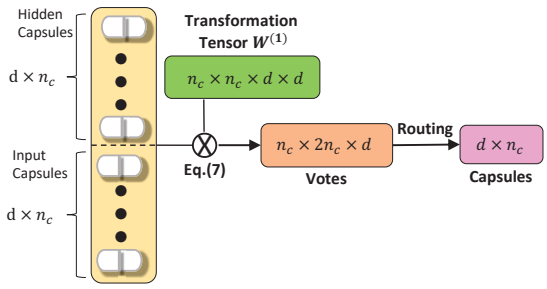

Fig. 2: **The illustration of dynamic routing process with non-shared transformation tensor. Element vectors are computed out by multiplying both hidden and input capsules with a transformation tensor. Then, the routing is adopted to iteratively generate n_c capsules of dimension d based on the Equation (2). The cross sign $\bigotimes$ refers to the Equation (7).**

(**Routing stage II**) We observe that generating the category capsule directly at the last level cannot obtain a prominent result, which is reported in our experimental results. This is because our model relies on recurrent structure where the hidden capsules sequentially reads input capsules of a word and updates its value, which leads to the overwhelming state estimation from the inputs in current steps. To this end, we design another routing stage to generate integration capsule from hidden capsules in each layer, which can remain the high-level information about hidden capsules from one recurrent step. Let $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times n_c \times d \times d}$ be the non-share transformation tensor used in routing stage II.

$$\hat{u}_{1|i}^{(2)} = \mathbf{W}_{1|i}^{(2)} \mathbf{h}_l^i + b_{1|i}^{(2)} \tag{8}$$

where $\mathbf{W}_{1|i}^{(2)} \in \mathbb{R}^{d \times d}$ is the $1|i$-th matrix of tensor $\mathbf{W}^{(2)}$. The element vectors obtained from above formula are fed into routing stage II to generate integration capsules $\mathbf{p}$. It is worth to mention that given the set of all the hidden capsules $\{\mathbf{h}_l^i\}$ from the same layer l this routing stage generates a unique integration capsule $\mathbf{p}_l$.

(**Routing stage III**) As the last part of recurrent module, category capsules can be generated by using integration capsules. Let $\mathbf{W}^{(3)} \in \mathbb{R}^{c \times d_o \times d}$ be the shared transformation tensor in routing stage III, where c is the number of categories and d_o is the dimension of category capsules. The element vector $\hat{u}_{j|i}^{(3)}$ in this routing stage is calculated by

$$\hat{u}_{j|i}^{(3)} = \mathbf{W}_j^{(3)} \mathbf{p}_i + b_j^{(3)} \tag{9}$$

where $\mathbf{W}_j^{(3)}$ is the j-th matrix of tensor $\mathbf{W}^{(3)}$ corresponding to the j-th category. The category capsule with the largest vector L2-normalization value is regarded as the category of sentence. We use a separate margin loss function [12] to estimate the optimal category capsules:

$$\begin{aligned} L_1 = \sum_{i=1}^{c} & (\mathbf{1}_i \times \max(0, \beta^+ - ||\mathbf{o}_i||)^2 \\ & + \eta(1 - \mathbf{1}_i \times \max(0, ||\mathbf{o}_i|| - \beta^-)^2), \end{aligned} \tag{10}$$

where $\mathbf{o}_i$ is the i-th category capsule and $||\mathbf{o}_i||$ is its L2-norm. $\mathbf{1}_i$ is an indicator function where $\mathbf{1}_i = 1$ if the sentence belongs to the i-th category; otherwise, $\mathbf{1}_i = 0$. β^+ and β^- are tuning parameters. η is a regularization parameter. In this work, we set $\beta^+ = 0.9$, $\beta^- = 0.1$ and $\eta = 0.5$.

C. Reconstruction Module

A reconstruction module can encourage the capsules to encode the representation of the input word vectors, which can improve the robustness of the model [12]. To this end, we design a reconstruction module based on the capsules of sentence representations obtained from recurrent module, instead of reconstructing the whole part of the raw inputs. In our model, the category capsules obtained from the integration capsules are a high-level encoding of the input word sequence, assembling the most important features and discard others. In this way, it is possible to reconstruct representations by leveraging category capsules. Inspired by [10], [29], we generate the reconstruction representation $\mathbf{r}_i$ of the category capsule $\mathbf{o}_i$ by multiplying $\mathbf{o}_i$ and its L2 norm:

$$\mathbf{r}_i = ||\mathbf{o}_i||\mathbf{o}_i. \tag{11}$$

On the other hand, we define the averaged integration capsules to be the representation of a sentence:

$$\mathbf{q} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i \tag{12}$$

where $\mathbf{p}_i$ is the i-th integration capsule obtained from routing stage II.

Now, the goal of reconstruction module is to ensure that the reconstruction representation r_i is similar to its corresponding sentence representation $\mathbf{q}$. Meanwhile, other category capsules are far from the sentence representation. To this end, it can be formulated by using the hinge loss:

$$L_2 = \max(0, 1 + \sum_{i=1}^{c}\mathbf{1}_i\mathbf{q}^\intercal\mathbf{r}_i). \tag{13}$$

Finally, the parameters $\mathbf{W} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}\}$ and $b = \{b^{(0)}, b^{(1)}, b^{(2)}, b^{(3)}\}$ in our capsule-based model can be estimated by optimizing the following objective:

$$\underset{\mathbf{W},b}{\operatorname{argmin}} L = \underset{\mathbf{W},b}{\operatorname{argmin}}(L_1 + \lambda L_2), \tag{14}$$

where λ is a tuning parameter and we set $\lambda = 0.01$ in this work.

V. Experiments

A. Datasets

Experiments are carried out on four benchmarks. Movie reviews (MR) [30] is a collection of movie reviews, containing 5,331 positive and 5,331 negative reviews. Subjectivity datasets (SUBJ) [31] contains 5,000 subjective and 5,000 objective sentences. In TREC question dataset (TREC) [31], each sentence is annotated with one of the six classes: Desc, Enty, Abbr, Hum, Loc, and Num. Finally, AG's news corpus [32] covers more than one million articles from more than two thousands newspapers and magazines, which are divided into four categories: World, Sports, Business, and Sci/Tec. Table I provides the detailed descriptions of the four benchmarks.

Dataset	Sentences	Classes	Training	Testing
MR	10,662	2	7463	3199
Subj	10,000	2	7000	3000
TREC	15,952	6	7463	3199
AG's news	127,600	4	7463	3199

TABLE I: **Summary of the four publicly available datasets.**

Besides, we evaluate our model on two challenging NLP tasks, i.e,, natural language inference and multi-label text classification.

B. Parameters

The GloVe vectors [28] are adopted to provide a mapping of raw word vector to a 300-dimensional space of embedded vectors ($v = 300$) based on a pre-trained model over 840 billion unlabeled corpus. All the dimensions of hidden capsules, integration capsules and category capsules are set to 64 (i.e. $d = d_o = 64$). The number of hidden capsules is set to 3 on TREC, AG's news and SUBJ, and 4 on MR.

Regarding the RCapsNet parameters, our models are trained with a mini-batch of 128 samples. The embedding dropout is 0.7, while the dropout for capsules in each recurrent layer is set to 0.8. Adam [33] is utilized as the optimization tool with its parameters β_1 and β_2 being set to 0.9 and 0.999, respectively. The learning rate is set to 0.001. All the model parameters are initialized by Xavier initializer [34]. The experimental results are averaged over 7-fold cross-validations. Finally, the RCapsNet is implemented on Tensorflow 1.2.

C. Baseline Methods

The classification performance of our RCapsNet model is compared against 11 established deep network-based models:
LSTM/Bi-LSTM [20]: LSTM is designed to solve the problem of gradient vanishing or exploding on standard RNNs. Bi-LSTM is a variant of LSTM.
Tree-LSTM [7]: It generalizes LSTM to tree-structured neural network using memory cells and gates.
LR-LSTM/LR-Bi-LSTM [11]: They are variants of LSTMs using linguistical regularization.
CNN-rand/-static/-non-static/-multichannel [21]: They use convolution and pooling operations to generate sentence representation and category. CNN-multichannel and CNN-non-static utilize different strategies on training and initialization.
VD-CNN [32]: It is a very deep CNN with up to 29 convolutional layers.
ConvCaps [9]: It adopts the rationale of capsules and convolutional operations.

VI. Results and Discussion

A. Evaluation and Comparison

Table II shows the averaged accuracy results over 7-fold cross-validations. Overall our RCapsNet model clearly out-

Method	MR%	TREC%	AG's%	Subj%
RCapsNet	81.1	**94.1**	**92.4**	**94.0**
LSTM	77.4	86.8	86.1	89.3
Bi-LSTM	79.3	89.6	88.2	90.5
Tree-LSTM	80.7	91.8	90.1	91.3
LR-LSTM	81.5	-	-	89.9
LR-Bi-LSTM	**82.1**	-	-	90.4
CNN-rand	76.1	91.2	91.2	89.6
CNN-multichannel	81.1	92.2	-	93.2
CNN-static	81.0	92.8	91.4	93.0
CNN-non-static	81.5	93.6	92.3	93.4
VD-CNN	-	85.4	91.3	88.2
ConvCaps	81.3	91.8	92.1	93.3

TABLE II: **Accuracies on the four evaluation datasets.**

Method	MR	Subj
RCapsNet	**81.5%**	**94.0%**
RCapsNet(-NRR)	79.9%	92.0%
RCapsNet(-L1)	80.9%	92.3 %
RCapsNet(-L2)	81.0%	92.2%

TABLE III: **Comparison of RCapsNet with different reconstruction settings. NRR, L1 and L2 refers to non-reconstruction regularization, L1 regularization, and L2 regularization, respectively.**

Method		MR	Subj
RCapsNet	Accuracy	**81.5%**	**94.0%**
	Iteration epochs	**12**	**10**
RCapsNet(-OCC)	Accuracy	78.0%	86.8%
	Iteration epochs	50	53

TABLE IV: **Comparison of RCapsNet with integration capsules and without them.**

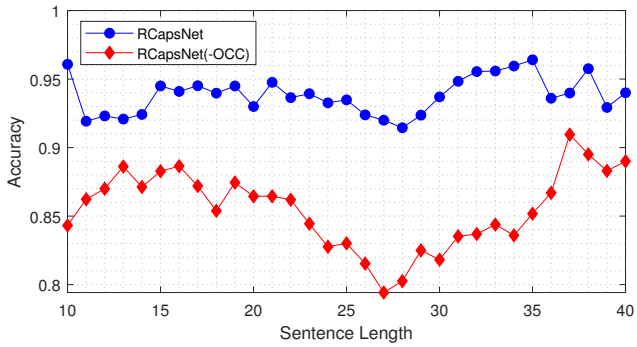

Fig. 3: **Accuracy comparison by varying the length of sentences.**

perform other competitive models with a large margin on the four benchmarks. This is mainly due to its abilities to take advantage of the rich temporal dependency information between words by recurrently generating a hierarchy of capsules. Notably, although ConvCaps also produces capsules through convolution operation, during training it does not encode temporal information among words. This might explain why RCapsNet has the best performance on the four benchmarks (94.1% vs 91.8%, 92.4% vs 92.1%, 94.0% vs 93.3%) expect MR. On the MR dataset, RCapsNet sightly underperforms LR-Bi-LSTM due to the linguistic regularizer and bidirectional structure in LR-Bi-LSTM, which are manually encoded. However, bidirectional structure, which contains tremendous parameters and linguistic regularizer, has to be handcrafted over prior linguistic knowledge that is impractical for most of the non-linguistician.

B. Ablation Study

To analyze the effect of different components in RCapsNet, we carry out the ablation test in this subsection to answer two intuitive questions: (1) What will happen if RCapsNet does not have reconstruction module? (2) Are integration capsules redundant?

To answer the first question, we remove the reconstruction module and train the model on MR and Subj datasets. Instead of reconstruction module, we adopt other two commonly used regularization approaches, such as L1 regularization and L2 regularization. The regularization weight λ is set to 0.01, keeping the same as it in reconstruction module. In addition, we also compare it with non-reconstruction regularization where no reconstruction module is designed. As shown in Table III, it is obvious that RCapsNets with different reconstruction settings (i.e. RCapsNet, RCapsNet(-L1) and RCapsNet(-L2)) outperform RCapsNet without it (i.e. RCapsNet(-NRR)). RCapsNet surpasses 1.6% of accuracy than RCapsNet(-NRR) on MR dataset and 1.8% on Subj dataset. Also, it is clear that RCapsNet with reconstruction module is more robust than those with L1 regularization or L2 regularization.

For the second question, we remove the integration capsules as well as routing stage III. Instead, we use another routing process called OCC, which obtains category capsules directly from hidden capsules. The accuracy and its corresponding iteration epoches are shown in Table IV, which suggest that RCapsNet preforms remarkably more accurate than RCapsNet(-OCC) on both MR and Subj datasets. Also, RCapsNet converges much faster than RCapsNet(-OCC). Additionally, it can be seen from Figure 3 that RCapsNet is significantly more robust than RCapsNet(-OCC) for sentences of various lengthes.

C. Experiments on Other NLP Tasks

To evaluate the performance of our model for more sophisticated natural language process takes, we carry out two additional experiments on natural language inference and multi-label text classification.

1) Natural Language Inference (NLI): NLI aims to find the semantic relationship between a premise sentence and a corresponding hypothesis sentence. We carry out an experiment on a widely used benchmark dataset called Stanford Natural Language Inference (SNLI) dataset [35]. This dataset contains three relationships (i.e. *entailment*, *neutral*, *contradiction*) and consists of $549,367$/ $9,842$/ $9,824$ premise-hypothesis pairs on train/dev/test set. The dataset has been annotated manually.

We constructed all the model structure designed in [35], launching two separated sentence encoding RCapsNet for

premise sentence and hypothesis sentence. We feed the integration capsules of each encoding model into a new non-shared parameter dynamic routing to get final category capsules. Note that in this experiment we abnegated the reconstruction module because it is only suitable for text classification task so far. For hyperparamters, the number of integration capsules of each model is set to be 5 with its dimension of 128. The number of category capsules with dimension of 64 is 3, which is equivalent to the number of categories.

The performance of RCapsNet and other typically competitive models are showed in Table V. It can be seen that RCapsNet outperforms most of the pervious model. In details, RCapsNet performs more accurate than the LSTM encoder model with around 7.7% (100D LSTM), 4.7% (300D LSTM), 2.0% (600D BiLSTM), 1.1% (600D BiLSTM with intra-attention) and 0.7% (memory-based NSE encoder network) boost. It is worth to notice that the 300D CAFE model slightly outperforms our model about 0.3% in accuracy. This is mainly because semantics are required to manually encode in its structure, which are more capable to handle the long sentence than ours.

2) Multi-Label Text Classification: The previous experimental results have shown that RCapsNet is capable of achieving competitive performance on single-label NLP tasks. However, an important problem in NLP field is associated with multiple classes or labels, leading to more challenges than single-label NLP tasks. For instance, in multi-label tasks, the label space is expanded from n to 2^n, which makes it difficult for a model to get accurate results in a reasonable time. To this end, we evaluate the capability of our RCapsNet on multi-label text classification in the following. Note that there is not sufficient datasets of multi-label text classification due to the difficulty and extremely high cost of collection and labelling. In this experiment, we evaluate our model on one publicly available dataset, Reuters-21578 dataset [42], which is a collection of documents with financial news articles. The original corpus has $10,788$ documents and a vocabulary of $29,930$ words, with each sentence either belonging to one single label or multiple labels. To make our model comparable, we followed the steps in [9] to preprocess the dataset. We utilized all the single-label and multi-label documents during training stage, and only multi-label samples were used during testing stage. We adopt precision, recall and F1 score as evaluation metrics on this dataset.

As shown in Table VI, it is clear that RCapsNet outperforms all other competitive models. In particular, RCapsNet and ConvCaps are more accurate than LSTM-based networks and CNN-based networks on multi-label text classification. This is mainly due to their abilities to take advantage of capsules that can catch rich temporal dependencies among words. RCapsNet outperforms ConvCaps with around 0.6% accuracy boost. This may be due to the fact that RCapsNet generates hierarchically organized capsules which contains more temporal relations. Overall, the experimental results demonstrate RCapsNet is capable of handling multi-label NLP tasks.

	Accuracy%	Recall	F1 Score
LSTM	86.7	54.7	63.5
BiLSTM	82.3	55.9	64.6
CNN-rand	88.6	56.4	47.1
CNN-static	91.1	59.1	69.7
CNN-non-static	92.0	59.7	70.4
ConvCaps	95.4	82.0	85.8
RCapsNet	**96.0**	**85.6**	**86.4**

TABLE VI: **Performance comparison on multi-label text classification.**

VII. Conclusion

In this paper, we present a recurrent capsule network for text classification called RCapsNet, which generate a hierarchy of capsules via dynamic routing to encode rich temporal dependencies in sentence sequences without any prior linguistic knowledge. Experiments on four benchmarks and two challenging NLP tasks show the effectiveness and practicability of our RCapsNet model on text classification. As for future work, we will consider improving reconstruction module, and we will further investigate the performance of our model on more NLP tasks. Transplanting the model to another fields like vision task is also a potential option.

Acknowledgement

This work was supported by grants from the National Major Science and Technology Projects of China (grant nos. 2018AAA0100703, 2018AAA0100700), the National Natural Science Foundation of China (grant no. 61977012), the Chongqing Provincial Human Resource and Social Security Department (grant no. cx2017092), the Central Universities in China (grant nos. 2019CDJGFDSJ001).

References

[1] Y. Y. Zhao, B. Qin, and T. Liu, "Sentiment analysis," *Journal of Software*, 2010.

[2] B. Liu, *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers, 2012.

[3] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, 2007.

[4] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *ECML*, 1998, pp. 137–142.

[5] A. Mccallum, "A comparison of event models for naive bayes text classification," in *AAAIW*, 1998.

[6] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *EMNLP*, 2011.

[7] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[8] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[9] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao, "Investigating capsule networks with dynamic routing for text classification," *arXiv preprint arXiv:1804.00538*, 2018.

[10] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment analysis by capsules," in *WWW*, 2018.

[11] Q. Qiao, M. Huang, J. Lei, and X. Zhu, "Linguistically regularized lstm for sentiment classification," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

Reference	Model	Number of parameters	Train Acc%	Test Acc%
Bowman et al., 2015 [35]	Unlexicalized features		49.4	50.4
Bowman et al., 2015 [35]	+ Unigram and bigram features		99.7	78.2
Sentence vector-based models				
Bowman et al., 2015 [35]	100D LSTM encoders	220k	84.8	77.6
Bowman et al., 2016 [36]	300D LSTM encoders	3.0m	83.9	80.6
Yang et al., 2016 [37]	600D BiLSTM encoders	2.0m	86.4	83.3
Yang et al., 2016 [37]	600D BiLSTM encoders with intra-attention	2.8m	84.5	84.2
Munkhdalai and Yu, 2017 [38]	300D NSE encoders	3.0m	86.2	84.6
Tao et al., 2017 [39]	300D Directional self-attention network encoders	2.4m	91.1	85.6
Choi, 2017 [40]	300D Gumbel TreeLSTM encoders	2.9m	91.2	85.6
Tay et al., 2018 [41]	300D CAFE (no cross-sentence attention)	3.7m	87.3	**85.9**
	RCapsNet	1.5m	91.3	85.6

TABLE V: **Accuracy comparison on the SNLI dataset. Train Acc and Test Acc refers to training and testing accuracies, respectively.**

[12] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," 2017.

[13] A. Jaiswal, W. Abdalmageed, and P. Natarajan, "Capsulegan: Generative adversarial capsule network," in *ECCV*, 2018.

[14] Q. Wang, J. Qiu, Y. Zhou, T. Ruan, D. Gao, and J. Gao, "Recurrent capsule network for relation extraction: A practical application to the severity classification of coronary artery disease," 2018.

[15] H. M. Wallach, "Topic modeling:beyond bag-of-words," in *ICML*, 2006, pp. 977–984.

[16] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text." *Science*, 1995.

[17] B. Pang, L. Lee *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, 2008.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119, 2013.

[19] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," 2013.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[21] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[22] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *WWW*, 2018.

[23] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *NAACL*, 2017.

[24] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Revisiting lstm networks for semi-supervised text classification via mixed objective function," in *AAAI*, 2019.

[25] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *ICANN*, 2011.

[26] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[27] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, 2018.

[28] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[30] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," 2005.

[31] P. Bo and L. Lee, "A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts," in *ACL*, 2004, p. 271.

[32] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*, 2016.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2014.

[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[35] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *EMNLP*, 2015.

[36] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, "A fast unified model for parsing and sentence understanding," in *ACL*, 2016.

[37] L. Yang, C. Sun, L. Lei, and X. Wang, "Learning natural language inference using bidirectional lstm model and inner-attention," *arXiv preprint arXiv:1605.09090*, 2016.

[38] T. Munkhdalai and H. Yu, "Neural semantic encoders," in *ACL*, 2017.

[39] S. Tao, T. Zhou, G. Long, J. Jing, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *AAAI*, 2017.

[40] Y. K. M. L. S.-g. Choi, Jihun, "Learning to compose task-specific tree structures," 2018.

[41] Y. Tay, A. T. Luu, and S. C. Hui, "Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference," in *EMNLP*, 2018.

[42] D. D. Lewis, "An evaluation of phrasal and clustered representations on a text categorization task," in *SIGIR*, 1992.