VOL 3 (2019) NO 4

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Efficient Processing of GRU Based on Word Embedding for Text Classification

Muhammad Zulqarnain[#], Rozaida Ghazali[#], Muhammad Ghulam Ghouse[#], Muhammad Faheem Mushtaq[#]

[#] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia*
E-mail: zulqarnainmalik321@gmail.com, rozaida@uthm.edu.my, apik.md@gmail.com, faheem.mushtaq88@gmail.com

***Abstract*— Text classification has become very serious problem for big organization to manage the large amount of online data and has been extensively applied in the tasks of Natural Language Processing (NLP). Text classification can support users to excellently manage and exploit meaningful information require to be classified into various categories for further use. In order to best classify texts, our research efforts to develop a deep learning approach which obtains superior performance in text classification than other RNNs approaches. However, the main problem in text classification is how to enhance the classification accuracy and the sparsity of the data semantics sensitivity to context often hinders the classification performance of texts. In order to overcome the weakness, in this paper we proposed unified structure to investigate the effects of word embedding and Gated Recurrent Unit (GRU) for text classification on two benchmark datasets included (Google snippets and TREC). GRU is a well-known type of recurrent neural network (RNN), which is ability of computing sequential data over its recurrent architecture. Experimentally, the semantically connected words are commonly near to each other in embedding spaces. First, words in posts are changed into vectors via word embedding technique. Then, the words sequential in sentences are fed to GRU to extract the contextual semantics between words. The experimental results showed that proposed GRU model can effectively learn the word usage in context of texts provided training data. The quantity and quality of training data significantly affected the performance. We evaluated the performance of proposed approach with traditional recurrent approaches, RNN, MV-RNN and LSTM," the proposed approach is obtained better results on two benchmark datasets in the term of accuracy and error rate.**

***Keywords*— RNN, GRU, LSTM, Word embedding, Text classification, Natural language processing**

I. INTRODUCTION

With the rapid development of social and technical information area, the continuous increasing the numbers of digital text format, means that the advent of huge text data age [1]. Text classification as an electronic basis for information retrieve, digital libraries and other fields, has a best application prospects [2]. Therefore, in this context how to organize and use these large amounts of text information becomes particularly important. The automatic text classification has become fundamental problem for big organization to manage the large amount of data. Automatic TC is a key technology to organizing and process large scale online resources. The lastly few years have realized remarkable attention in deep learning approaches that learn layered, hierarchical representations of high-dimensional data [3]. These deep learning approaches have been effectively applied to pattern recognitions [4], sentiment analysis [5] and computer vision [6], but not widely to text classification. The traditional text representation that there are dimension of disaster, data sparse and other issues has become a bottleneck in the performance of a large number of natural language processing tasks.

The advent of deep learning approaches has increase to a number of new techniques for sentiment analysis. The availability of huge unlabelled textual data can be used to learn the meanings of words and the structure of sentence formation. This has been attempted by word2vec [1] which learns word embeddings from unlabelled text samples. It learns both by predicting the word given its surrounding words (CBOW) and predicting surrounding words from given word (SKIP-GRAM). These word embeddings are applied for creating dictionaries and act as dimensionality reducers in existing method like Tf-idf, etc. More approaches are found capturing sentence level representations like recursive neural tensor network (RNTN) [7]. Convolution neural network which has primarily been used for image related task has been shown effective in text classification [8].

The main problem is the variable length of the natural language. Some of it is solved by fixed size context windows but it fails to extract semantics which extend longer than the context window. Recurrent neural network have the

capability to take variable size of text sequence but they are extremely tricky to learn. Hence new types of RNN were employed like LSTM and GRU. LSTM was proposed in 1997 by Hochreiter et al. [9] and is making news in many NLP task like sentiment analysis, translation and sequence generation. GRUs is quite recent development proposed by K. Cho [10] in 2014. GRU are much simpler in structure and probably more practical than LSTM. We attempt to show its advantages over LSTM in sentiment analysis in this work. Different approaches have used by other researcher included are Tf-idf, Word2Vec (vector average), Word2Vec (k – means dictionary), and Ensemble model. It was found that among all the single models GRU outperformed all of them, whereas this result of GRU model was further improved after using ensemble model.

In this research, we have to investigate the effectiveness of GRU network based on pre-trained word embedding method such as Glove for text classification. Applied GRU network for the goal of overcome the issue of gradient exploding or vanishing in an existing RNNs. In this work, we conduct the experimental setup on two benchmark datasets, Google snippets and TREC and compared the performance of GRU model with three existing RNNs models namely are, Recursive Neural Network (RNN), Matrix-Vector Recurrent Neural Network (MV-RNN) and Long Short Term Memory (LSTM). We found that GRUs is effective in the task of text classification because of their capability to remember long time dependencies and efficiently capture the semantics between words. GRU approaches are especially useful for sequential datasets. Experimental results demonstrate that the (our approaches) GRU model performs better on both datasets in the term of accuracy and error rate

II. Related Work

Classification is one of the hot topic and has been widely used in natural language processing. Texts mining which does not only contain of classification also included unstructured data analysis such as topics detection, spam filtering, sentiment analysis, sematic role labelling and documents classification. It is usually recognized that deep learning architectures have obtained great achievement in natural language processing Unstructured data passes through a series of processes while it is being converted into structured form; preprocessing, feature selection, term weighting and finally obtaining document vectors respectively. There are several deep learning models that have been used in various tasks of NLP such as web news classification, chunking, semantically related words and language modelling. A natural language model especially estimates the expectation of the next word being *w* in a sequence [11]. In instance, the deep recursive neural networks model have been applied for parsing and sentiment analysis [12] and question answering [13] and logical inference [14]. Recurrent neural networks have been successfully used in language modelling [15], speech recognition, and sentence generation from images [16].

A pioneering work introduced by Bengio et al. [17] applied a neural network language model (NNLM) to learn word embeddings based on the preceding contexts of each word. The C&W model explored the concept of applying the preceding contexts alone to in-corporate both preceding and succeeding contexts into word embeddings using a convolutional network. Mikolov et al. [18] further proposed the CBOW and skip-gram approaches that apply a simple single-layer structure to allow effective processing of word embeddings from very big datasets. In [19] respectively presented that approaches developed word embeddings based exclusively on linear contexts and local contexts (typically a few words in the preceding and succeeding contexts), and thus proposed dependency-based word embeddings and global vectors (GloVe). The semantics-based word embeddings are handled the limitation of linear contexts by proposing syntactic con texts derived from a dependency parser, however the GloVe method addresses the limitation of local contexts by accounting for global word to word co-occurrence statistics.

Recent advances in recurrent neural network-based language models (RNN-LM) have presented the value of distributed representations and the capability to model arbitrarily long dependencies [20]. In [21] explains a simple variation of the RNN that can produce meaningful sentences by learning from a character-level corpus. More recently, [22] have illustrated that an RNN-LM is ability of producing image descriptions by conditioning the network model on a pre-trained convolutional image feature representation. Moreover, the ability to train deep networks provides a more sophisticated way of exploiting relations among labels and features, therefore making the prediction more accurately.

In addition, this feature enables that RNNs is applicable for sequential tasks such as text classification and named entity recognition. Furthermore, [23] proposed a tree-LSTM, a variant of RNN allowing for richer network topologies where each LSTM unit is able to incorporate information collected from multiple child units. In addition, Zhou et al. [24] achieved success in extracting meaningful features from documents automatically by combining bi-directional LSTM with an attention mechanism.

III. Text Classification Steps

Usually, text classification process contains four major steps: [25]. However, an initially steps regarding to collecting and preparing the datasets on which the work will be done.

A. Data Preprocessing

Preprocessing techniques plays important role to improve the performance of the models. It is the initial step to convert unstructured texts into structured form and increase the quality of the texts dataset by preprocessing technique in text mining studies and its applications. We discuss the three key steps of data preprocessing namely, tokenization, stop words removal, and stemming.

1) Tokenization

Tokenization is a technique of breaking a stream of texts into words, phrases, symbols, or other useful elements as known tokens. The purpose of the tokenization is the exploration of the words in a sentence.

2) Stop Word Removal

In this stage the removal of common meaningless words which are filtered out before or after processing of natural

language data (text). For few websites, mostly used common and short function words, such as 'the', 'an', 'is', 'at', 'of', 'but' and so on.

3) Stemming

Stemming is the technique of mislead the variants forms of word into a similar representation, the stems. For example, the text words: "presentation", "presented", "presenting" could all be reduced to a common representation "present". It's an extensively applied in text processing for information retrieval (IR) based on the assumption that posing a query with the term presenting implies attention in documents consisting the words presentation and presented.

B. Document representation

In this stage, before classification, data have to be illustrated in the format that the classification algorithm can identify Bag of Words (BOW) is one of the most applied techniques. It is a representation of text that defines the occurrence of words within a document.

C. Dimension Reduction

Sometimes, hundreds of thousands of words in a text corpus, so it is not possible to do the classifying for all those words as features; also, the computer could have issues to computing such amounts of data. That is why it is essential to choose the best illustrative features as inputs for the classification step.

D. Model Training

This is the major step of text classification. It consists selecting a portion of text from the dataset to determine the leaning set, implements the training on it, and after that produce the model.

IV. Model Illustration

Deep learning approaches were derived from artificial neural networks and currently it is a largest area of machine learning and has successfully applied to achieve an excellent performance in several fields, such as pattern recognition, signal processing, computer vision and natural language processing. In this section, we have decided to explore RNNs models for sequential data of modelling and text classification.

A. *Recurrent Neural Networks*

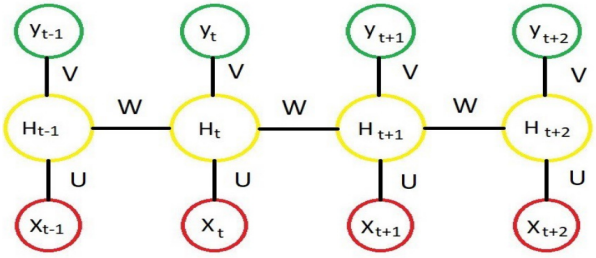

Fig. 1 The architecture of traditional RNNs

Recurrent neural networks (RNNs) are derived from artificial neural networks in which connections between the units a bidirectional cycle. RNN structure is designed to sequential events included word sequences; this architecture is usually appropriate for extracting the significant linguistic information through the long sequential of words within a corpus. At each time-step, the current input and the previous time steps *t* hidden state are add as inputs after passing through their weights matrix U, W and V respectively as presented in Fig. 1.

Equations included of RNN are:

$$h_t = f_h\,(U.x_t + W.h_{t-1} + b_{(h)}) \quad (1)$$

$$O_t = f_o(V.h_t) \quad (2)$$

Where equation 1 is to compute the hidden state value h_t and equation 2 to compute the output O_t. *W* is the recurrent weight matrix, U is the input to hidden layer matrices and V is the hidden layer to output matrices. However, RNNs are hard to train and suffer from vanishing and exploding gradients issue. Either the gradients become so small that learning stops or the gradient becomes so large that the weights overflow the max length. The most efficient solution to this issue is adding a gating mechanism to the RNNs. Two gated variations of RNNs, Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU), have been developed to great tickle gradient vanishing problems.

B. *Long Short-Term Memory (LSTM)*

LSTMs were proposed by Hochrieter et al. in [9] which is developed on the basis of RNN to handle the issue relevant to gradient vanishing or exploding. LSTM mechanism is basically use varieties of gates for the ultimate goal of controlling the passing sequential data. LSTM network has similar flow as a standard RNNs can regulates the propagation of activations along the network which provides it to learn when to ignore a current input, when to remember the past hidden state or when to emit a non-zero input. These networks are efficiently remembering data for long or short period of time and hence the name Long Short-Term Memory. Mathematically, the architecture of LSTM is presented in Fig 2.

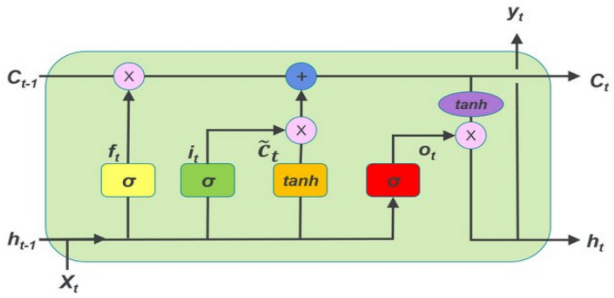

Fig. 2 The block architecture of LSTM

There three gates applied: input gate, output gates and forget gate. The input gate decide how much relevant information to add from current step, forget gate determine how much of the previous state is let through and output gate determine how much the current node affects the external network. Mathematically, the relationship between the inputs, forget and the output gates of LSTM is computed by a set of the following equations.

$$i^{(t)} = \sigma_g\,(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \quad (3)$$

$$f^{(t)} = \sigma_g\,(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \quad (4)$$

$$o^{(t)} = \sigma_g\,(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \quad (5)$$

$$\hat{c}^{(t)} = \tanh\left(W^{(\hat{c})}x_t + U^{(\hat{c})}h_{t-1} + b^{(\hat{c})}\right) \quad (6)$$

$$c^{(t)} = f^{(t)} * c^{(t-1)} + i^{(t)} * \hat{c}^{(t)} \quad (7)$$

$$h^{(t)} = o^{(t)} * \tanh(c^{(t)}) \quad (8)$$

Equations (3), (4) and (5) are used to compute the value of the input, forget and output gates respectively. Equation (6) is employed to compute the value of the proposed state $\hat{c}_t$ which is replaced to find the current state c_t in (7). Finally, the hidden state h_t is computed using (8). Each of the three gates and the proposed state has their own weights matrix U and W with a bias vector *b* that is learn through training. 'o' indicates element-wise product.

V. PROPOSED ARCHITECTURE

In this section, we describe the details of the proposed architecture, which include the input layer, embedding layer, recurrent layer, output layer and finally classification layer as softmax. Our architecture uses word embeddings as inputs and require them to a GRU learn to capture the semantics features, and are finally follow by a classifier layer.

A. *Word Embedding layer*

Word embeddings concentrates on training distributed word vector representation of words by leverage the contextual data in big dataset using deep learning approaches. Distributed representations are useful for the input of deep neural networks. Existing representations, such as one-hot representation, will lead to the curse of dimensionality [26]. Recent research [24] presents that neural networks can converge to a better local minima with a suitable unsupervised pre-training procedure. In this experiment, we set the word embedding dimension *d* and the GRU hidden dimension. Then, all words embedding from the text data were initializes by 200-dimensional GloVe word vectors pre-trained by Penington et al. [27]. This model is the state-of-the-art in several NLP tasks. In this technique, context words are applied as inputs of a neural network and effort to classify the target word. In [28], the author shown GloVe, a competing set of pre-trained embeddings, indicated that word embedding was unexpectedly between the mainstream. Consistently, each feed-forward RNN which attains words from a term as an input and embeds them as vectors into a lower dimensional space, and it then refine all through back propagation, essentially crop word embeddings as the weights of the first layer, referred as Embedding Layer. More specifically, the initial the relationship among the probabilities of the coexistence of two words is factors that contain information, and therefore relies on the encoding of this information as a vector difference.

B. *Text classification using Recurrent Layer*

RNN is a particular kind of artificial neural networks structure especially applied for sequential modelling. GRU is other variant of RNN and relatively development introduced by Cho et al. [10]. GRU is Similar to the LSTM unit, but without a separate memory and has gating mechanism that controls the flow of information inside the unit. GRU calculates two gates called update and reset gates which handle the flow of information from previous activation when calculating new candidate state by using a reset gate. The amount of previous activation and new candidate activation into new activation are tied by an update gate. Each hidden state at time-step *t* is computed using the following equations:

Update gate:

$$z_t = \sigma_g\left(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}\right) \quad (9)$$

Reset gate:

$$r_t = \sigma_g\left(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)}\right) \quad (10)$$

Candidate state

$$\hat{h}_t = \tanh\left(W^{(\hat{h})}x_t\right) + U^{(\hat{h})}(r_t * ht_{-1}) \quad (11)$$

Final Output

$$h_t = z_t * h_{t-1} + (1 - z_t) * \hat{h}_{t-1} \quad (12)$$

where * is element-wise multiplication and σ_g is the logistic sigmoid activation function. W_z, W_r, $W_{\hat{h}}$ are the feedforward weights and U_z, U_r and $U_{\hat{h}}$ are the recurrent weights of the update gate, reset gate and output candidate state respectively. b_z, b_r and $b_{\hat{h}}$ are the biases of these gates and candidate state $\hat{h}_t$.

The update gate is calculated from the current input and the hidden state of previous time step. This gate determines to handle how much of portions of new memory and old memory should be combine in the final memory. Similarly the reset gate is computed but with various set of weights. It manages the balance among previous memory and the new input information in the new memory.

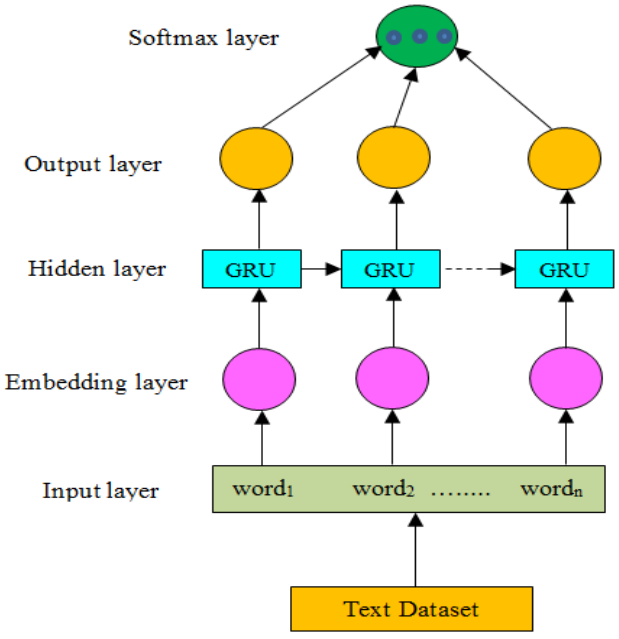

Fig. 3 The proposed GRU-Embedding base architecture for text classification

C. *THE CLASSIFICATION LAYER*

In neural networks, for text sentiment classification, softmax regression is frequently implemented as a final layer for binary and multiclass classification. Its computes fast and provide results with a probabilistic description. It provides a

fixed-dimensional input from the lower layer; the classification layer refines changes followed by a softmax activation function to calculate the predicting possibilities for all of the categories [29]. This is done by:

$$\hat{Y} = \frac{\exp(W_o^T S^* + b_o)}{\sum_{i=1}^{C} \exp(W_o^T S^* + b_o)} \qquad (13)$$

where $\hat{Y}$ is the predicted distribution of the text, Y_{pre} is the selected text label, W_o^T and b_o are the parameters of the softmax classifier to be learned.

VI. Experiment Setup

In this experiment, we conducted an experiments study to evaluate the proposed word embedding based GRU model for text classification on two benchmark datasets: Google snippets and TREC dataset to train & test our model. And we compared proposed model with other traditional RNNs approaches include, MV-RNN, LSTM, and Recursive Neural Network.

A. Datasets

To evaluates the efficiency of the proposed GRU model, we conducted experiment respectively on two benchmark datasets Google Snippets [30] and TREC [31].

1) Google Snippets

This dataset contain of 10,060 training snippets, and 2,280 for testing snippets from 8 classes as presented in table 1. The average words of each snippet has 18.05

2) TREC.

As illustrated in table 2, TREC consists 6 various question types classes such as LOC, NUM, ENTY and so on. The training dataset contains of 5,452 labeled questions, and the test dataset contains of 500 questions.

TABLE I
STATISTICS SUMMARY OF GOOGLE SNIPPETS DATASETS

Labels	Training	validation
Business	1200	300
Computers	1200	300
Culture-arts-entertainment	1870	330
Education science	2330	320
Engineering	200	130
Health	890	300
Politics-Society	1230	300
Sports	1140	300
Total	**10,060**	**2,280**

TABLE II
STATISTICS SUMMARY OF TREC DATASETS

Labels	Training	Validation
DESC.	1157	134
ENTY.	1255	98
ABBR.	89	11
HUM.	1220	63
NUM.	890	110
LOC.	841	84
Total	**5,452**	**500**

B. Implementation setting

In order to improve the performance of the proposed model that first step is the quality of the dataset, we enhance the quality of text dataset by preprocessing technique such as eliminating stop words from the input sequence (e.g. “and”, “are” “of”, “the”, “to”) and also used streaming to reduce all similar presentation words in one common word and punctuations. In this experiment, we set the word embedding dimension *d* and the GRU hidden dimension. Then, all words embedding from the text data were initializes by 200-dimensional GloVe word vectors pre-trained by Penington et al. [27]. Some researchers adopted the fine-tuned training strategies for word vectors to enhance the performance for sentence sentiment classification tasks [32].

In variation, with the aim of well reflected generalization capability of the model, we prefer to apply the common embeddings for all datasets. What is more, we processed all the context words as classification resource words to implementing the self-attention mechanism as if there was no classification resource word in the texts. For the deep learning networks, the hidden states of the GRU unit in each layer were set to 200. Through the training process, we optimized the proposed model with the AdaDelta algorithm [33] by following the learning rate of 0.001 and the mini-batch size of 64. To alleviate the overfitting problem, we applied the dropout strategy [34], with a dropout rate of 0.5 or the GRU layer and 10^{-5} for the coefficient λr of L2 regularization. To evaluate state-of-the-art performance, we have used accuracy and error rate as a metrics for the text classification task. Completely simulations were implemented on Intel Core i7-3770XPU @3.40 GHz, and 4GB of RAM machine.

VII. Empirical Results and Analysis

In our framework, the out-of-vocabulary words in texts are simply discarded, since they are often low-frequency tokens. The overall experimental performance and results are summarized as follows.

A. Overall performance

In this experiment, we executed the implementation of our proposed GRU model on two Google snippets, and TREC benchmark datasets with particular parameters. We employed to evaluate performance of the proposed GRU model with three well-known traditional recurrent neural network included Recursive Neural Network (RNN), Recursive Matrix-Vector Neural Network (MV-RNN) and Long Short Term Memory (LSTM) on both datasets. Model training was completed over the stochastic gradient descent through shuffled mini-batches. We also observed that maintaining the embedding dimensions equivalent to the number of gated units performed well than networks containing units much more than words embedding dimensions. In our work, we employed the pre-trained GloVe method in the word-level embedding layer in sequence to transfer each word in the texts for calculating a real value vector representation of a word.

In this experiment, we evaluated the classification results, based on standard evaluation metrics of accuracy and means square errors (MSE) were used to compared state-of-the-art three existing RNNs models, namely RNN, MV-RNN and LSTM. For the first dataset of Google snippets, the performance comparisons between three models are presented in Fig. 4

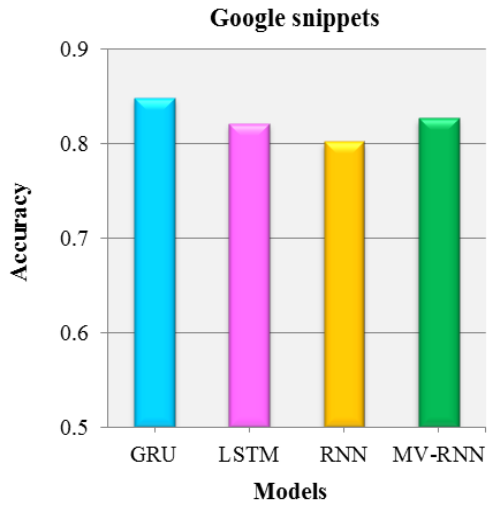

Fig. 4 The performance comparison of three classifiers for Google snippets

As shown in Fig.3, the best performance can be achieved for GRU with an accuracy of 0.848. We can see the consistently better performance for our proposed model than RNN, MV-RNN and LSTM. Recursive recurrent network showed the worst performance due to its high false positive rates. Next, the performance for more casual TREC dataset is presented in Fig. 5

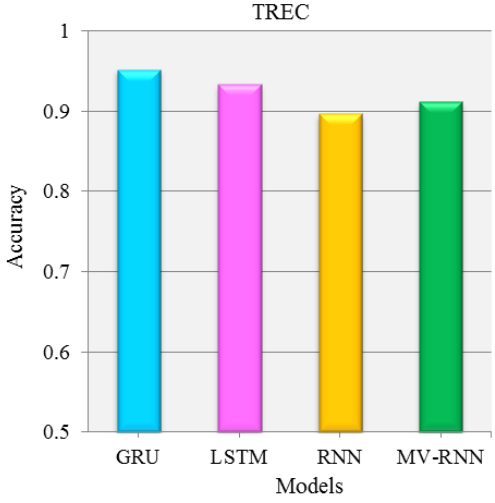

Fig. 4 The performance comparison of classification for TREC

As shown in Fig. 5, the performance of all models is comparable with slight differences. The best performance has achieved GRU in the term of accuracy 0.952, while LSTM obtained a comparable F-measure of 0.934.

B. Comparison an Error rate with traditional RNNs

In this section, we perform to analysis an error rate of our proposed GRU model with three state-of-the-art deep learning RNNs models LSTM, MV-RNN and RNN. Execution setup showed that with the continuous increase of epochs, the mean square error is continuously decreasing and the final MSE is 1.182 on Google snippets dataset and 0.629 on TREC.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(P_i - P_i^*\right)^2$$

Where n is the number of the data points and P_i, P_i^* are refers to observed values and predicted values. We fixed both the word embedding dimensions and number of units to 64 and execute the model on various parameters are mention in implementation setting. We found that proposed model converged faster as compared other RNNs models to achieved lower error rate even after many epochs. To make these models comparable, we implement these models with the identical structural design. We evaluate our GRU model with state-of-the-art existing RNNs models on two Google snippets, and TREC datasets. Table 1 demonstrates the results that proposed GRU model achieves much better performance in the term of the error rate than RNN, MV-RNN and LSTM.

TABLE III
COMPARISON ERROR RETE (%) WITH EXISTING RNNS MODELS.

Models	Google snippets	TREC
RNN	2.063	0.977
MV-RNN	1.652	0.914
LSTM	1.757	0.783
GRU (Proposed)	**1.182**	**0.629**

VIII. CONCLUSION

In this paper, we have addressed the issue of efficiently analyzing the document-level text classification approach based on GRU. In this way, we explored traditional Recurrent Neural Networks (RNNs) architectures on two benchmark text classification datasets. We found that Gated Recurrent Units to be the better choice in terms of accuracy and error rates. The pre-trained words embeddings such as GloVe word vector model are used to extract the textual semantics features between words in texts. We conclude that GRU is a suitable model for sequential data of text classification especially when there are more amounts of learning data. Being a recurrent network it can effectively capture long sequence data required for natural language understanding. Empirical results on two benchmark datasets included (Google snippets and TREC) show that our proposed model achieved state-of-the-art superior performance compared to other traditional RNNs models.

Furthermore, it is remarkable to observe future work on investigating proposed model for further tasks of natural language processing.

REFERENCES

[1] J. Protasiewicz, "A recent overview of the state-of-the-art elements of text classification," *Expert Syst. Appl.*, vol. 106, pp. 36–54, 2018.

[2] W. Sharif, N. A. Samsudin, M. M. Deris, and M. Aamir, "Improved relative discriminative criterion feature ranking technique for text classification," *Int. J. Artif. Intell.*, vol. 15, no. 2, pp. 61–78, 2017.

[3] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," pp. 1–23, 2017.

[4] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks," *IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1717–1724, 2014.

[5] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment Embeddings with Applications to Sentiment Analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. October, pp. 496–509, 2016.

[6] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by

multi-context deep learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07–12–June, pp. 1265–1274, 2015.

[7] O. I. and C. Cardie, "Deep Recursive Neural Networks for Compositionality in Language," *Adv. neural Inf. Process. Syst.*, pp. 2096–2104, 2014.

[8] A. Dahou, M. A. Elaziz, J. Zhou, and S. Xiong, "Arabic Sentiment Classification Using Convolutional Neural Network and Differential Evolution Algorithm," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–16, 2019.

[9] S. Hochreiter, "Long Short Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1–32, 1997.

[10] K. Cho, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," *arXiv*, vol. 5, pp. 1–9, 2014.

[11] R. Collobert and J. Weston, "A unified architecture for natural language processing," *Proc. 25th Int. Conf. Mach. Learn. - ICML '08*, pp. 160–167, 2008.

[12] R. Socher, A. Perelygin, and J. Wu, "Recursive deep models for semantic compositionality over a sentiment treebank," *Proc. …*, no. October, pp. 1631–1642, 2013.

[13] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III, "A Neural Network for Factoid Question Answering over Paragraphs," *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process.*, pp. 633–644, 2014.

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

[15] A. Kumar *et al.*, "Ask Me Anything: Dynamic Memory Networks for Natural Language Processing," vol. 48, 2015.

[16] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light Gated Recurrent Units for Speech Recognition," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 2, pp. 92–102, 2018.

[17] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, and J. Cernocky, "Neural network based language models for highly inflective languages," *Icassp-2009*, pp. 4725–4728, 2009.

[18] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation ofWord Representations in Vector Space," *arXiv Prepr. arXiv1301.3781*, pp. 1–12, 2013.

[19] M. J. Berger, "Large Scale Multi-label Text Classification with Semantic Word Vectors," *Tech. Rep.*, pp. 1–8, 2014.

[20] P. Liu, X. Qiu, and X. Huang, "Recurrent Neural Network for Text Classification with Multi-Task Learning," *Proc. 25th Int. Jt. Conf. Artif. Intell. IJCAI-16*, p. to appear, 2016.

[21] Y. Xiao and K. Cho, "Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers," *arXiv*, vol. 1602, no. 00367, 2016.

[22] A. Karpathy, "Deep Visual-Semantic Alignments for Generating Image Descriptions."

[23] M. Sundermeyer, H. Ney, and R. Schlüter, "From Feedforward to Recurrent LSTM Neural Networks for Language Modeling," *IEEE/ACM Trans. Audio, Speech, Lang. Process*, vol. 23, no. 3, pp. 517–529, 2015.

[24] Z. H. I. Li, F. A. N. Yang, and Y. Luo, "Context Embedding Based on Bi-LSTM in Semi-Supervised Biomedical Word Sense Disambiguation," *IEEE Access*, vol. 7, pp. 72928–72935, 2019.

[25] V. Srividhya and R. Anitha, "Evaluating Preprocessing Techniques in Text Categor ization," *Int. J. Comput. Sci. Appl.*, vol. 47, no. April, pp. 49–51, 2010.

[26] R. Johnson and T. Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks," no. 2011, 2014.

[27] J. Pennington, R. Socher, and C. D. Manning, "GloVe : Global Vectors for Word Representation," *Proc. Conf. Empir. Methods Nat. Lang. Process.*, no. October, pp. 1532–1543, 2014.

[28] A. Cochez *et al.*, "This is an electronic reprint of the original article . This reprint may differ from the original in pagination and typographic detail . Global RDF Vector Space Embeddings," *Int. Semant. Web Conf.*, pp. 190–207, 2017.

[29] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions," *Proc. Conf. Empir. methods Nat. Lang. Process.*, no. ii, pp. 151–161, 2011.

[30] X. Phan, "Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections," *Proc. 17th Int. Conf. World Wide Web*, pp. 91–100, 2008.

[31] D. Roth, "Learning Question Classifiers £," *Proc. 19th Int. Conf. Comput. Linguist.*, vol. 1, no. August, pp. 1–7, 2002.

[32] H. Lee, "for Modeling Sentences and Documents," *Proc. 15th Annu. Conf. North Am. Chapter Assoc. Comput.*, no. June, pp. 1512–1521, 2015.

[33] D. P. Kingma and J. L. Ba, "A method for stochastic optimization," *arXiv*, no. March, pp. 1–15, 2015.

[34] G. Hinton, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res. 2014, 15*, vol. 15, pp. 1929–1958, 2014.