Simultaneous Generation-Classification Using LSTM

Daniel L. Marino, Kasun Amarasinghe, Milos Manic
Department of Computer Science
Virginia Commonwealth University
Richmond, Virginia
marinodl@vcu.edu, amarasinghek@vcu.edu, misko@ieee.org

***Abstract*— The idea that a concept is properly learned by an agent when the agent is able to generate examples and non-examples of the concept, has motivated research on generative models. Generative models are trained with the aim of improving performance of tasks such as classification. In this paper, a Long Short Term Memory (LSTM) architecture for simultaneous generation-classification is presented. The architecture is designed with the purpose of serving as a model which can generate sequence samples, while simultaneously classifying a given sequence. The presented generation-classification methodology was implemented on a sentiment analysis task. However, it can be applied to any sequence modelling or classification task. The experimental results suggest that this approach can be particularly useful as a regularization methodology which acts similarly to pre-training through Restricted Boltzmann Machines or auto-encoders.**

Keywords—Deep Learning; Deep Neural Networks; Long-Short-Term memory; LSTM, sentiment analysis.

I. INTRODUCTION

The idea that a concept is properly learned by an agent only when the agent is able to reproduce examples of it, is a mechanism that humans often use to evaluate their understanding of a particular concept. Further, it is used as a tool for improving their comprehension of the subject. As a concrete example, Frayer Models [1], [2] are often used as learning strategies in schools to evaluate the comprehension of a subject by asking the student to provide examples and non-examples of a specific concept.

Example and non-example generation is naturally an interesting task that researchers on machine learning want to replicate in machines. Therefore, it has gained increased attention. Especially, after the introduction of the contrastive-divergence algorithm [3], [4] for training Restricted Boltzmann Machines (RBM). RBMs are usually trained in an unsupervised way with the aim of providing a probabilistic model that has a high probability of generating samples similar to the ones used for training the model.

Research on Deep Learning [5] has fueled the interest on the development of efficient generative probabilistic models of data that provide accurate probabilistic inferences and fast simulation sampling of fantasy data (generated data) from the inferred model [6].

Generative models have proven to be an effective tool for improving the performance of deep neural networks architectures [7], [8], [9]. Unsupervised pre-training through RBM and auto-encoders provided the first tools for successfully training of deep architectures [7] and have been shown to act as regularizers [8] . Those methods also have been successful on for obtaining useful reusable features using unsupervised representation learning [9], which can be used to improve classification tasks.

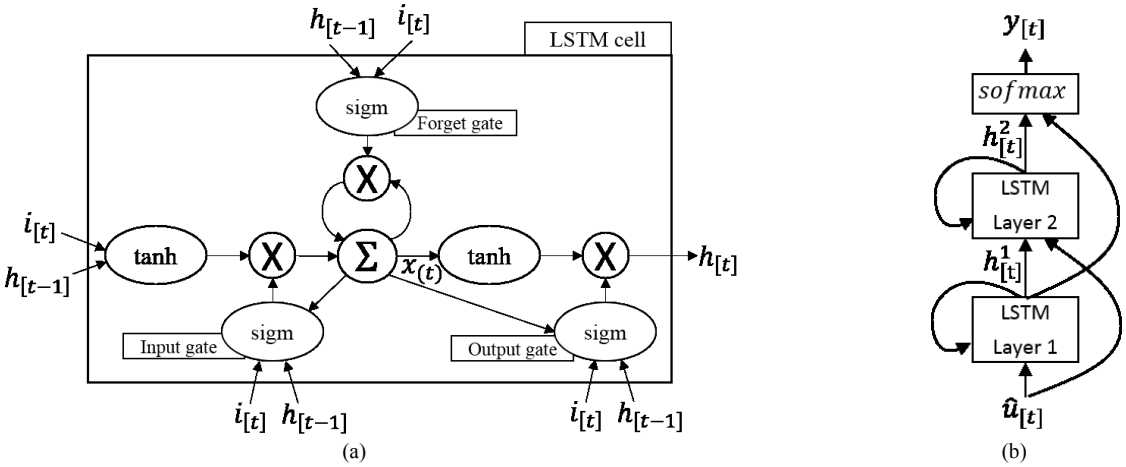
Furthermore, generating data from models provide a way of understanding how the model is behaving, and is an indication of what it has learned. Therefore, it provides a way to prove the model and visualize its internal structure [10]. It also is a tool for debugging the training process, allowing to identify why or why not is the model performing well.

In this paper, an architecture for sequence generation and classification is presented. The objectives of the developed model are; 1) generating sequences for a given class and 2) given a sequence, predicting the class to which it belongs.

Long Short Term Memory (LSTM) is a type of recurrent neural network that has become a benchmark model for sequence modeling, and have been extensively used by the Natural Language Processing community [11], [12], [13]. In addition, LSTMs have been successfully used for applications that include but not limited to energy load forecasting [14], visual attention models [15], handwriting generation [11], automatic caption generation from images [16], translation [12] and language modeling [11]. Their success is attributed to its capability of modeling long-range structures; thanks to the fact that they were designed specifically to alleviate the vanishing gradient problem [17].

Given the success of LSTM networks as sequence models, the presented generation-classification architecture is composed of a set of LSTM models. Each model is trained for generating sequences belonging to a specific class. For classifying a given sequence, losses that correspond to the error committed by each network are used to predict the class of the sequence

The presented LSTM based generation-classification architecture was implemented and tested on a sentiment analysis task. Previous LSTM models for sentiment analysis have also used a joint training between the language model and the classifier [13]. However, the main difference of the presented architecture is that it aims to obtain a separate model for each concept (each class) found in the dataset. The models are jointly trained to generate samples corresponding to its respective class and to identify to which class a given sample belongs. Furthermore, the study also explores the benefits that simultaneous generation-classification training methodology has on the classification performance.



(a) (b)

Fig. 1. (a) LSTM cell, (b) multilayer LSTM architecture

The presented architecture was tested on two datasets for sentiment analysis. The most interesting result found in the experiments was that, when the architecture is trained to jointly minimize classification error and generation loss, it achieves better performance on testing dataset rather than when the same architecture is trained only to minimize classification error. This suggests that simultaneous generation-classification training acts as a regularizer, providing models with better generalization.

The rest of the paper is organized as follows. Section II provides a brief introduction of LSTM networks. Section III presents the proposed generation-classification architecture. Section IV shows the loss and specifics for the training of the architecture. Section V presents the results on the two datasets used for sentiment analysis. Section VI presents a discussion on the results and the pros/con of the architecture is presented. Section VII concludes the paper.

II. LSTM Model

This section provides an overview of the used LSTM model. The same LSTM model used in [11] is employed in this paper. The model for the cells is shown in Fig. 1.a. Eq. (1.a) through (1.f) express a single LSTM cell's operation

$$i_g = \sigma\big(i_{[t]}W_{xi} + h_{[t-1]}W_{hi} + x_{[t-1]}W_{ci} + b_i\big) \quad (1.a)$$

$$f_g = \sigma\big(i_{[t]}W_{xf} + h_{[t-1]}W_{hf} + x_{[t-1]}W_{cf} + b_f\big) \quad (1.b)$$

$$z = \tanh\big(i_{[t]}W_{xc} + h_{[t-1]}W_{hc} + b_c\big) \quad (1.c)$$

$$x_{[t]} = f_g \circ x_{[t-1]} + i_g \circ z \quad (1.d)$$

$$o_g = \sigma\big(i_{[t]}W_{xo} + h_{[t-1]}W_{ho} + x_{[t]}W_{co} + b_o\big) \quad (1.e)$$

$$h_{[t]} = o_g \circ \tanh(x) \quad (1.f)$$

where $\circ$ denotes the element-wise product. i_g corresponds to the input gate, f_g to the forget gate and o_g to the output gate. $x_{[t]}$ is the value of the memory state at time step t, $h_{[t]}$ the output of the cell and z is the update signal. σ is the sigmoid function. W_{ci}, W_{cf}, and W_{co} are diagonal matrices. For simplicity, all vectors are represented as row vectors.

Fig. 1.b shows the LSTM multilayer architecture used in [11], with $L = 2$ layers. This architecture is usually used in sequence modeling, where given a sequence of T consecutive samples $u_{[1:T]} = \{u_{[1]}, u_{[2]}, \dots, u_{[T]}\}$, we would like to predict the value of u in the next time step ($u_{[T+1]}$). For this purpose, the architecture on Fig .1.b uses a softmax function to map the activation value of the hidden layers $\{h^l_{[t]} \mid l = 1, \dots, L\}$, to a discrete probability distribution $y_{[t]}$:

$$y_{[t]} = \Psi\big(\big[h^1_{[t]} \quad \dots \quad h^L_{[t]}\big]W_y + b_y\big) \quad (2)$$

where Ψ is the softmax function. $y_{[T]}$ is therefore a model of the probability distribution of $u_{[T+1]}$ given $u_{[1:T]}$:

$$y_{[T]} = P\big(u_{[T+1]} \mid u_{[1:T]}\big) \quad (3)$$

For discrete inputs, usually $u_{[t]}$ is encoded as a one-hot encoding, which is represented by $\hat{u}_{[t]}$.

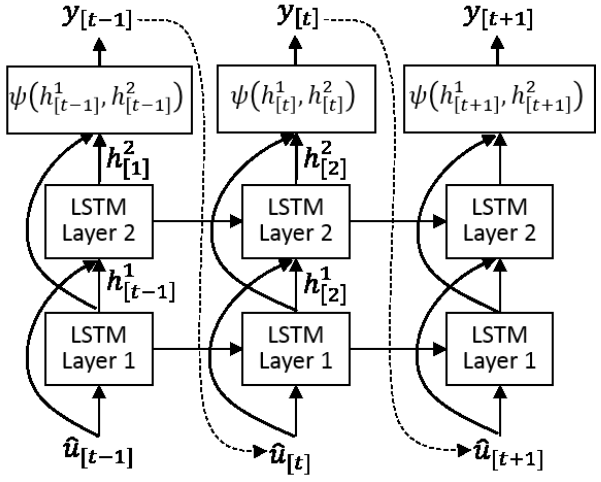


Fig. 2. Sequence generation using LSTM network

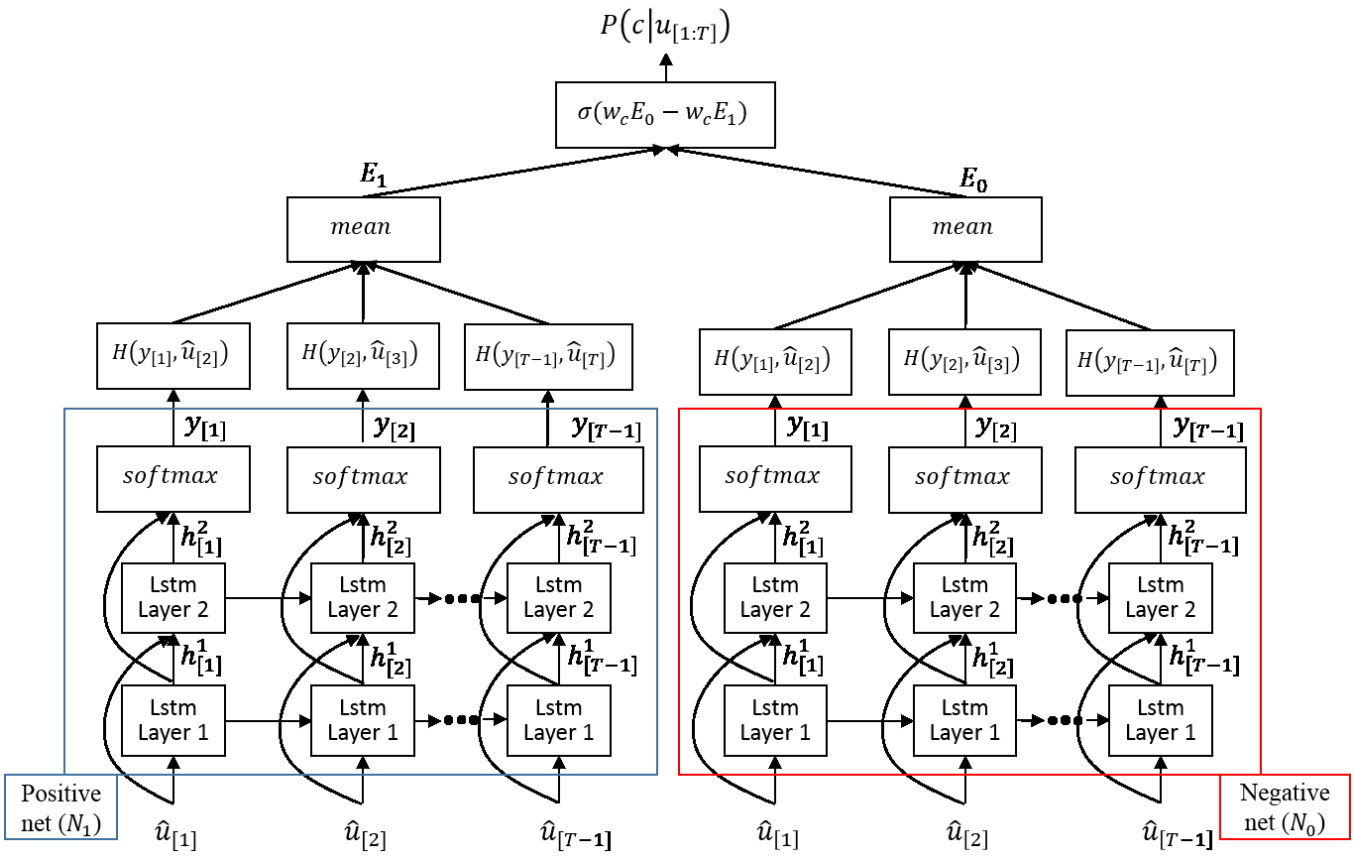


Fig. 3. Simultaneous generation-classification architecture

III. LSTM BASED SIMULTANEOUS GENERATION-CLASSIFICATION MODEL

This section elaborates the presented LSTM based simultaneous generation-classification model.

Assume we have a set of sequences u, each belonging to a class $c \in C$, where C is the set of possible classes (E.g. $C = \{\text{positive}, \text{negative}\}$). An architecture that aims to achieve simultaneous generation-classification of those sequences, should provide a model from where sequences for each class can be generated. I.e. it should provide a model for $P(u_{[T+1]}|\, u_{[1:T]}, c)$. At the same time, given a sequence, $u_{[1:T]}$, the model should be able to predict the class which the sequence belongs to. I.e. it should estimate $P(C|u_{[1:T]})$.

The architecture depicted in Fig .1.b gives us a model for generating sequences [11]. One simple approach for generating a sequence $u_{[1:T]}$ is to initialize the state $x_{[0]}$ and outputs of the hidden layers to zero (or a random value). The first value of the sequence $u_{[1]}$ is randomly initialized. Then, the calculation of $y_{[1]}$ is performed by following Eq.1 . $u_{[2]}$ is sampled (generated) from $y_{[1]}$ and used for obtaining $y_{[2]}$ from where $u_{[3]}$ can be sampled. Recursively using the outputs of the network as inputs for the next time steps, a sequence of arbitrary length can be generated. This procedure is illustrated in Fig. 2, where the dashed lines are used to represent the sampling of $u_{[t]}$ from $y_{[t-1]}$.

Given that the architecture of Fig .1.b can be used to generate sequences, we can have a set of networks, $Net = \{N_c |\, c \in C\}$, where each one of the network N_c serves as a sequence model for each class c.

Having the set of networks Net, the class of a given sequence $u_{[1:T]}$ can be predicted by evaluating which N_c better predicts the given sequence. This is the idea behind the presented generation-classification architecture.

Eq. 4 gives the metric used in this paper to evaluate how well a network N_c predicts the sequence $u_{[1:T]}$.

$$E_c(u_{[1:T]}) = \frac{1}{T}\sum_{t=1}^{T} H(N_c.\, y_{[t]}, \hat{u}_{[t+1]}) \qquad (4)$$

where $N_c.\, y_{[t]}$ represents the distribution $P(u_{[t+1]}|u_{[1:T]})$ obtained using the network N_c. $\hat{u}_{[t+1]}$ is the one-hot encoding of $u_{[t+1]}$, and $H(N_c.\, y_{[t]}, \hat{u}_{[t+1]})$ is the cross-entropy between $P(u_{[t+1]}|u_{[1:T]})$ and $\hat{u}_{[t+1]}$.

A model for $P(C|u_{[1:T]})$ can be constructed by evaluating $E_c(u_{[1:T]})$ for all classes $c \in C$, using Eq. 4, and then introducing the values of E_c into a softmax. This operation is expressed in Eq. 5

$$P(C|u_{[1:T]}) = \Psi([E_1 \quad \dots \quad E_{|C|}]W_c + b_c) \qquad (5)$$

where E_k represents Eq. 4 evaluated for network N_k, which corresponds to class k and $|C|$ is the number of classes.

Fig. 3. illustrates the proposed generation-classification architecture for a two class setting $C = \{0,1\}$, where 0 represents the "negative" class and 1 the "positive" class. Basically we have a network N_1 that is trained to generate positive sequences and a network N_0 which generates negative samples. It is important to note that, for the two-class setting, the softmax of Eq. 5 is replaced by a sigmoid function.

IV. Training of the Simultaneous Generation-Classification Model

This section details the training methodology used for the simultaneous generation-classification model.

For training the generation-classification architecture, a multi-objective optimization problem which aims to jointly minimize the classification error together with the perplexity of the model, is formulated.

The architecture is trained on a set of pairs $(u_{[1:T]}, c)$, where $u_{[1:T]}$ is a given sequence and c is the class to which $u_{[1:T]}$ belongs. The classification loss is defined as:

$$L_c(u_{[1:T]}, c) = H(P(C|u_{[1:T]}), \hat{c}) \quad (6)$$

where $\hat{c}$ is a one-hot encoding of class c.

For sequence generation, each one of the networks $N_c \in Net$ is desired to have a low prediction error (according to Eq. 4) over sequences $u_{[1:T]}$ that belong to their corresponding class c. Therefore, the following is the loss to reduce perplexity:

$$L_p(u_{1:T}, c) = E_c(u_{[1:T]}) \quad (7)$$

We also would like to have high prediction errors when a network N_c is evaluated under a sequence $u_{[1:T]}$ that does not belong to its corresponding class. For this purpose, Eq. 4 is evaluated on a decreasing positive function that asymptotically converges to zero. In this case, we chose $\exp\left(-E_k(u_{[1:T]})\right)$. The following is the loss that aims to maximize the prediction error for sequences that do not belong to the respective class of the network $N_c \in Net$:

$$L_{cp}(u_{[1:T]}, c) = \sum_{k \in C} \begin{cases} 0, & if\ k = c \\ \exp\left(-E_k(u_{[1:T]})\right), & o/w \end{cases} \quad (8)$$

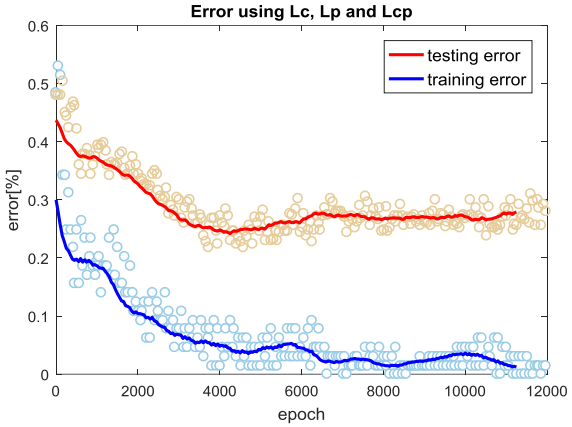
Therefore, the objective function to be minimized is:

$$L = \sum_{(u_{[1:T]}, c)} L(u_{1:T}, c) \quad (9.a)$$

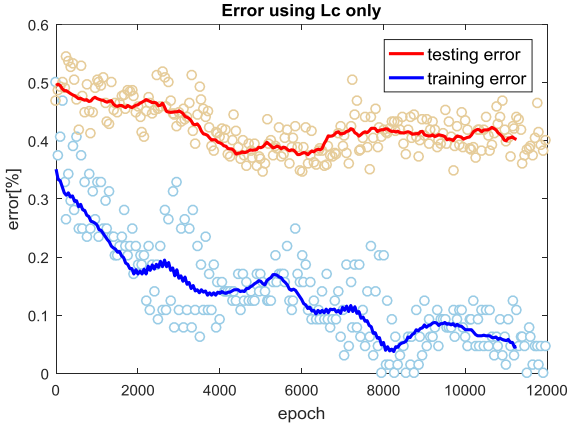
$$L(u_{1:T}, c) = L_c(u_{[1:T]}, c) + \alpha L_p(u_{[1:T]}, c) + \beta L_{cp}(u_{[1:T]}, c) \quad (9.b)$$

where α and β are hyperparameters that weight the contribution of each of the corresponding losses.

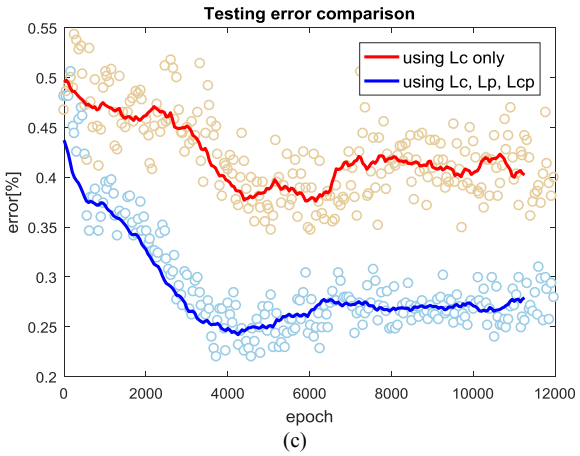
Training was performed using Backpropagation through time [18] by unrolling T times each one of the LSTM networks and minimizing Eq. 9. ADAM [19] algorithm was used as the gradient based optimizer. During the minimization process, gradient norm clipping [20] was used to alleviate the exploding gradient problem. To reduce overfitting problems, dropout [21] was introduced as a regularization scheme. The architecture was implemented using TensorFlow [22]



(a)



(b)



(c)

Fig 4: Comparison of training and testing errors on a network trained to achieve: (a) low generation/classification errors (b) only low classification errors. (c) shows the testing error comparison between the two schemes. The error being shown is the percentage of misclassified sequences. These results correspond to the Twitter dataset

V. EXPERIMENTS

The generation-classification architecture was implemented and tested on a sentiment analysis task [1]. Given a set of positive and negative reviews, the proposed architecture was trained to generate positive and negative reviews and classify a given review into positive or negative class.

The architecture was tested on a dataset of tweets [24], and on the imdb movie review dataset [23]. The architecture shown in Fig.3 was used for both datasets given that both datasets were treated as binary classification problems.

A. *Twitter dataset results*

The Twitter dataset consisted of 5,137 tweets, divided on 3,094 positive and 2,043 negative tweets. The dataset contained a total of 16939 characters [24].

For this dataset, the architecture was trained at character-level, therefore $\hat{u}_{[t]}$ is a one-hot encoding over a set of graphemes. Only lowercase characters in the English alphabet were considered, uppercase characters were converted to their corresponding lowercase grapheme.

$P(C|u_{[1:T]})$ was evaluated according to Eq. 10 instead of Eq. 5, given that the tweets are classified only as positive and negative:

$$P(C|u_{[1:T]}) = \sigma(E_0 w_c - E_1 w_c) \qquad (10)$$

where w_c is a positive scalar. Eq. 10 is basically just a soft threshold between the prediction errors of N_0 and N_1

The dataset was divided 90% for training and 10% for testing, the training was performed using an unrolling of 100 characters. Using a network of two layers, with 50 neurons in each layer, the presented architecture achieved classification errors of 18% on testing and 6.2% on training, using early stop to prevent overfitting. Table I shows some of the sequences generated by this model.

To evaluate the effects that the joint generation-classification training has on the classification performance of the architecture, the performance of a network trained using Eq. 9 as the objective function was compared with the performance of the same network trained using only L_c on Eq. 9 as the objective function. The idea behind the experiment was to evaluate how the performance of the network varied when trained for simultaneous generation-classification (Eq.9 which includes L_c, L_p, L_{cp}) and when it is trained only for classification (i.e. using only L_c as the objective). The architecture was kept the same for both scenarios to ensure that the capacity of the network is not changed. The network used for the experiment

TABLE I. SAMPLES OF SENTENCES GENERATED BY THE NETWORK TRAINED USING THE TWITTER DATASET

Positive	c latet ac dcwit ing we th witerad lot in wizkis yawe ist ytamazy ou ; gorc dc al azvol baok buck cowtist to bor ok tes ifl cletos mt yo ; francicoftos tterid ac fot tinfrredring on tunten freakinn to nlai ; qle s anvewe valsere a dyiwiby be counee ay gay sef alats gocket aur ; qbk ate mizonsa fadd nc tocerti me ding to amachizs in dlved bopco cack
Negative	spusoy fo man toing i t ivi merme nore carkcoft hale wharl jsow be ; zy the mustat mazon ride ed xit gredange onglive time it dh craspfard ; hof frame danosen sntimidoaus soveveor marnit a wure andiligut groe ; aa berther thimak of re hay bud you dig bel maye is progoe dry adrela ; core soft as pucleturans buft aalazln vreene on hate laii a day vou

[1]Code available at: github.com/MHRG-VCU

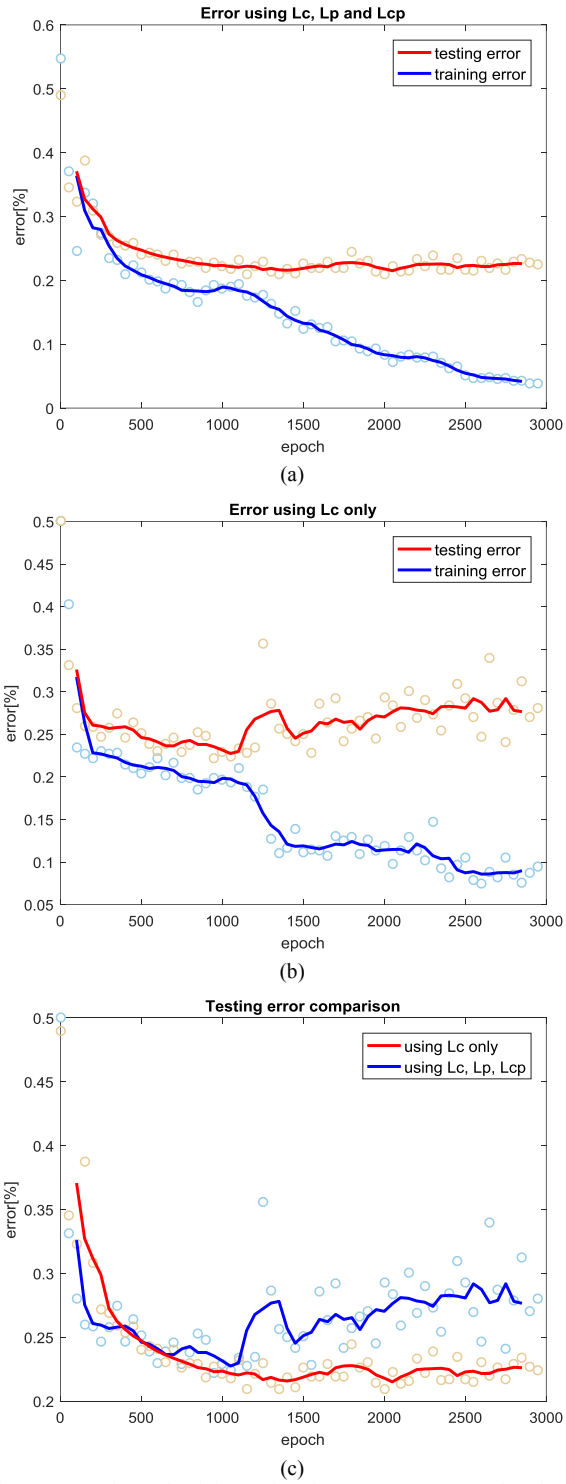


Fig 5: Comparison of training and testing errors on a network trained to achieve: (a) low generation-classification errors (b) only low classification errors. (c) shows the testing error comparison between the two schemes. The error being shown is the percentage of misclassified sequences. These results correspond to the imdb dataset

comprised of two layers, with 10 units in the first layer and 15 units in the second layer. The results of the experiment are shown in Fig. 4, where the error (percentage of misclassified sequences) trough epochs is shown for both scenarios.

The estimation of the classification error for training and testing set is noisy because of the mini-batch training procedure, therefore in Fig. 4 the error through epochs is shown together with an averaged version to improve analysis.

Fig. 4 shows that for both objective functions, the architecture is able to achieve low training error. On the other hand, as seen in Fig. 4.c., using L_c, L_p and L_{cp} as the objective function gives better classification performance on testing data than using only L_c. These experiments suggest that jointly training the network with the aim of achieving simultaneous generation-classification serves as a powerful regularization scheme, allowing the architecture to improve generalization.

B. Imdb dataset

The Imdb dataset is a movie review dataset, with reviews classified as positive or negative. It consists of 50,000 movie reviews, 25,000 positive and 25,000 negative [23]. Further, the dataset is divided 50%- for training and 50% for testing.

This dataset is composed by 32,376,725 characters. Due to the size of this dataset, the architecture was trained in this case at word-level, therefore $\hat{u}_{[t]}$ is a one-hot encoding over a set of words in a dictionary.

The dictionary is composed by the 3000 most frequent words that were found in the dataset. The word frequency was counted after a stemming process. When a word in a sentence is not found in the dictionary, it is ignored (eliminated) before being processed by the generation-classification architecture.

The training dataset was divided 90% for training and 10% for validation, the training was performed using an unrolling of 64 words. Using a network of two layers, with 300 neurons in each layer, the presented architecture achieved classification errors of 18.5% on validation, 2% on training, and 20% on testing, using early stop to prevent overfitting. Table II shows some of the sequences generated by this model. Given that the training of the network was done using stem words, the generated sentences are composed by stem words. This is the reason of not having well-formed words for the generation.

It is important to mention that given a sufficiently large network, the architecture can easily overfit the dataset achieving perfect classification on training dataset, although the performance in testing is significantly decreased. An overfitted model also generates sequences that make more sense, as shown in Table III. The overfitted network had perfect classification performance on training set, and an error of 26% on testing.

The obtained results are slightly worse than reported in [13]. However, it has to be noted that a much simpler word encoding method was used in the presented work. Furthermore, the emphasis of the paper was to explore the design and benefits of generation-classification architectures, not on improving sentiment analysis classification. Sentiment analysis was intended to serve as a benchmark for testing the architecture.

The same tests performed on the Twitter dataset were performed on the imdb dataset to compare the classification accuracies with and without simultaneous generation-classification training. The results are shown in Fig. 5. The model used for the experiments consisted of two LSTM layers, each with 100 units. The imdb dataset proved to be a more challenging task than the Twitter dataset. A plateau in the training process was found, as seen in Fig. 5.a.

It was seen that the model could get almost perfect classification accuracy on training data using the simultaneous generation-classification training loss (L_c, L_p and L_{cp}). However, even when the error in training data was dramatically reduced, the error on testing dataset was not considerably reduced from the testing error that was achieved during the plateau. (Fig. 5.a).

During the experiments, it was seen that the model trained without including the generation loss (only L_c), often plateaued. As a measure to improve the classification performance, the architecture of Fig. 3 was changed by bypassing the output $y_{[1:T]}$ from the LSTM networks straight to the mean calculation, i.e. ignoring the cross-entropy term H. The results shown on Fig. 5 used this modification. , By bypassing the cross entropy operation, the architecture was able to further reduce the training error, but as can be seen on Fig. 5. b., the generalization of the model is significantly reduced after the training error exits the plateau..

Comparing the testing error performance between the two settings, joint training for generation-classification still provides a better error on testing dataset, although the difference is not as substantial as in the Twitter dataset. Fig. 5.c. shows the comparison between the two approaches. Through the experiments, we noticed that using simultaneous generation-

TABLE II. SAMPLES OF SENTENCES GENERATED BY THE NETWORK TRAINED USING THE IMDB DATASET

Positive	beyond what thi is wonder and comedi a sort of unexpect littl stori wa an almost in anim bourn genr will treasur be alcohol goe on an oscar matthau new film had record it mostli too late and knife back they dont realli enjoy it the charm wonder everi bug bond cartoon and act especi albert is offer their stage as tri to boot and high school they pass a book spectacular the director is portray as american and man and the caus so last thi thing in the way secret civil is a everi step of and accept that onli have the problem of be a consist in the book the of life the murder by the joy are now allow to and observ by the killer who entir lie to the dark system not just and the answer to the
Negative	pure action structur came loos i wonder what the hell would i agre with the stone read a lot of favor read the direct and stori itself for the most part stephen king are spoiler in mani way that the director doesnt rememb us common much from wa just an interest storylin by wa it dull as the whose name the knowledg and go past for a real world and he cinematographi and thi just rate a three rate the last number of minut of the movi thi movi is noth short of a cult classic i think thi is a wast of my time even can you say well the reason suck thing thi movi wa an and that what im surpris that thi movi wa unfortun the rate of or the act had the me to give a tast of

TABLE III. SAMPLES OF SENTENCES GENERATED BY THE OVERFITTED NETWORK TRAINED USING THE IMDB DATASET

Positive	by thi and in my opinion the best film ive seen in a long time after watch movi a sequel are combin with give a great movi that impress you by can past word how to love when your have christma yet polic to the of a who through all of the peopl it is an intellig and documentari well give a as the killer the film director creat an atmospher that we cannot imagin an extrem well made into an theme by all these element to make a veri good movi thi film is for real valu and is certainli not a polit well i am to see how it is made for a high product and highli somewhat origin stori also obvious in end like the movi success it easili for the charact and is interest peter is amaz as the and hi girlfriend
Negative	attract cant tell it a plot myself how did thi movi wa is still possibl the worst movi i have ever put out time like some peopl have a of consid thi is a movi about you will see howev expect for relat to the charact but at the veri least he doesnt even de a black guy who know woman would want an and her film beauti cant explain how he pull sometim she could least be with all get in troubl an attempt at social on a to make stick with it one unless your watch a film like thi it where plenti of peopl for thi movi is about minut to sing the thing on is done in the cast are worth wrong use it subject in a veri differ stori but just fail to into when it wa into the stori had

classification training also lead to a more stable training. Fig. 5 shows evidence for this observation. As can be seen, training without generation leads to a more "noisy" training (see Fig. 5.b). Moreover, finding hyper-parameter values that result in stable training was more challenging than using the simultaneous generation-classification architecture.

VI. DISCUSSION

The conducted experiments showed that the presented simultaneous generation-classification architecture has several advantages. One major advantage of the presented architecture is that it provides a direct way for proving the model, which allows to understand and evaluate the performance of the network and how it is behaving. Another major advantage is that, as shown by the experimental results, the presented architecture enables using the generation procedure as a regularization technique for the optimization/training procedure. Furthermore, the presented architecture enables creating models for each concept (each class) in the form of networks, $N_c \in Net$.

In addition to the advantages mentioned above, several drawbacks of the presented architecture were identified. One such drawback is that training of the "generative" model needing supervised data. Although a pre-training phase using unsupervised data could be introduced, the learning algorithm relies heavily on the premise that labeled data are available. Another drawback of the presented architecture is that since optimization problem is multi-objective, minimization of the objective function does not necessarily produce a reduction on the classification error. Furthermore, a major drawback of the system is that the number of hyper-parameters is increased. Therefore, the amount of parameters that needs to be tweaked increases, introducing variability in results depending on those values.

VII. CONCLUSIONS

This paper presented an initial version of an architecture for simultaneous generation and classification of sequences based on LSTMs. The presented architecture was implemented and tested on a sentiment analysis task with two different datasets. The experiments carried out on the two datasets revealed that the presented architecture had several advantages including the fact that the generation procedure can be used as a regularization technique for the classification. However, it was also noticed that the presented architecture had some drawbacks, which included the need of labeled data and the increased number of hyper-parameters. Several fronts of future work were identified. Since the presented architecture is an initial version, future work will be conducted to improve the classification capability and the generation capability of the architecture. In particular, different versions of loss functions will be experimented with. Further, the multi-objective optimization process needs to be enhanced to guarantee a stable reduction of generation and classification losses during optimization.

REFERENCES

[1] S. C. Greenwood, "Making Words Matter: Vocabulary Study in the Content Areas," *The Clearing House: A Journal of Educational Strategies,* pp. 258-263, 2002.

[2] C. W. Peters, "A Comparison between the Frayer Model of Concept Attainment and the Textbook Aproach to Concept Attainment," *Reading Research Quartely,* vol. 10, no. 2, 1974.

[3] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation,* vol. 14, no. 8, pp. 1771-1800, 2002.

[4] G. E. Hinton, "A fast learning algorithm for deep belief nets," *Neural Computation,* vol. 18, no. 7, pp. 1527-1554, 2006.

[5] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[6] D. J. Rezende, S. Mohamed and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31 st International Conference on Machine Learning*, Beijing, 2014.

[7] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007.

[8] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio and P. Vincent, "The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, 2009.

[9] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *arXiv preprint arXiv:1511.06434*, 2015.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, 2014.

[11] A. Graves, "Generating Sequences with Recurrent Neural Networks," 5 Jun 2014. [Online].

[12] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *NIPS 2014*, 2014.

[13] K. S. Tai, R. Socher and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, Beijing, 2015.

[14] D. L. Marino, K. Amarasinghe and M. Manic, "Building Energy Load Forecasting using Deep Neural Networks," in *Press IECON*, 2016.

[15] L. Bazzani, H. Larochelle and L. Torresani, "Recurrent Mixture Density Network for Spatiotemporal Visual Attention," in *arXiv preprint arXiv:1603.08199* , 2016.

[16] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[18] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of IEEE,* vol. 78, no. 10, pp. 1550-1560, 1990.

[19] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," in *ICLR*, San Diego, 2015.

[20] R. Pascanu, T. Mikolov and Y. Bengio, "On the Difficulty of Training Recurrent Neural Networks," in *30th International Conference on Machine Learning*, Atlanta, 2013.

[21] V. Pham, T. Bluche, C. Kermorvant and . J. Louradour, "Dropout Improves Recurrent Neural Networks for Handwriting Recognition," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Heraklion, 2014.

[22] M. Abadi, A. Agarwal, P. Barhman, E. Brevdo, Z. Chen, C. Citro and et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," in *arXiv preprint arXiv:1603.04467*, 2016.

[23] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[24] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani and V. Stoyanov, "SemEval-2016 task 4: Sentiment analysis in Twitter," in *10th international workshop on semantic evaluation (SemEval 2016)*, San Diego, US, 2016.