Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

The survey: Text generation models in deep learning

Touseef Iqbal [a,1,*], Shaima Qureshi [a]

[a] Department of Computer Science and Engineering, National Institute of Technology, Srinagar, Jammu and Kashmir India

ARTICLE INFO

Article history:
Received 9 December 2019
Revised 29 March 2020
Accepted 1 April 2020
Available online xxxx

Keywords:
Natural Language Processing (NLP)
Deep learning
Word embeddings
Recurrent Neural Networks (RNNs)
Convolutional Neural Networks (CNNs)
Variational Auto-Encoders (VAEs)
Generative Adversarial Networks (GANs)
Text generation techniques
Activation functions
Optimization techniques

ABSTRACT

Deep learning methods possess many processing layers to understand the stratified representation of data and have achieved state-of-art results in several domains. Recently, deep learning model designs and architectures have unfolded in the context of Natural Language Processing (NLP). This survey presents a brief description of the advances that have occurred in the area of Deep Generative modeling. This work considers most of the papers from 2015 onwards. In this paper, we review many deep learning models that have been used for the generation of text. We also summarize the various models and have put forward the detailed understanding of past, present, and future of text generation models in deep learning. Furthermore, DL approaches that have been explored and evaluated in different application domains in NLP are included in this survey.

Contents

* Corresponding author.
E-mail addresses: touseef_04phd18@nitsri.net (T. Iqbal), shaima@nitsri.net (S. Qureshi).

[1] Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2020.04.001
1319-1578/

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

1. Introduction

Individuals are most exceptional species on earth, the achievement as people is the ability to convey and share data. The idea of creating language comes in when we discuss human language, it is one of the attenuated and complex pieces of us. According to the (Shacklett, 2017) industry estimates just 21 percent of information is organized. Information is being created everywhere at large scale as tweets and sending messages on WhatsApp or different groups on Facebook, where the larger part of the information exists in the textual form, which is highly unstructured in nature. Now, so as to create significant bits of knowledge from this information, it is important to get to know the systems of Natural Language Processing (NLP). NLP is the area of software engineering and Artificial Intelligence (AI) that manages human languages. It is the computational technique where we represent and analyze language automatically. Learning sentence representation is fundamental to numerous natural language applications (Dair.ai, 2018). These models try to learn the fixed-length feature vector which encodes the semantic and syntactic properties of sentences. One of the popular approaches to train a sentence model is the encoder-decoder framework by using Recurrent Neural Network (RNN) (Elman et al., 1990).

Since the research on NLP has emerged from the era of punch cards and batch processing where the processing of sentiment analysis may take several minutes to the era of Google and likes of it, in which millions of web-pages can be processed in less than a second. The impressive results shown by deep learning architectures on computer vision, pattern recognition, and network traffic analysis made our attention to follow the same trend on NLP. Since the research on NLP is growing at a very high speed. As NLP tasks are hard to solve, but deep learning is required to get us the state-of-the-art on many challenging problems in this area (Machine Translation (Zhang et al., 2015), Text Summarization (Nallapati et al., 2016)). Deep learning has evolved many algorithms in the field of NLP like Recurrent Neural Networks (RNNs) (for sequence modeling), Recurrent Neural Networks with external memory (RNN-EM)(to improve memory capacity of RNN) (Peng and Yao, 2015), Gated Feedback Recurrent Neural Networks (GF-RNN) (stacking multiple recurrent layers with gating units) (Chung et al., 2015), Conditional Random Fields as Recurrent Neural Networks (CRF-RNN)(for probabilistic graphical modeling) (Zheng et al., 2015), Quasi-Recurrent Neural Networks (Q-RNN)(using parallel time-steps for sequence modeling) (Bradbury et al., 2016), Memory Networks (for Question Answering (QA)) (Weston et al., 2014), Augmented Neural Networks (Neural Turing Machines) (Olah and Carter, 2016).

One of the famous deep learning models known as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) which has revolutionized the field of computer vision particularly image generation (CycleGAN (for cross-domain transfer)). These models based on GANs are different from conventional models in deep learning as they possess an adversarial way of training the network. The structure of GANs is discussed in section-III. GANs have been progressively used for images. Many variants of GANs in the image domain have been proposed in recent years (DCGANs (Radford et al., 2015), CapsuleGANs (Jaiswal et al., 2018), FictitiousGANs (Ge et al., 2018)), but the progress in the text domain has been seen limited due to the discrete nature of the text. Many techniques in recent years have been used to fit the GAN models for the generation of text. Another popular deep learning mechanism for data generation which has been extensively used in recent years is Variational Auto-Encoder (VAE) (Kingma and Welling, 2013). This paper discusses the recent approaches made in the field of these generative models for the generation of text and the algorithms which surrounds these mechanisms. The structure of the paper is as follows: Section II introduces algorithms and technqiues used in the field of text generation like distributed representation of words, Recurrent Neural Networks (RNNs), Convolutional Neural Networks etc. Section III explains Variational Auto-Encoders (VAEs), Generative Adversarial Networks (GANs) in the field of text generation. Section IV discusses some evaluation methods. Finally Section V summarizes it all.

2. Related work

Recent approaches in artificial neural technology, particularly progressive deep learning have affected vigorously on the field of Artificial Intelligence (AI), regularly characterizing the state-of-the-art in the solutions for a majority of complex tasks in various unique areas. NLP is no special case. In numerous regions of NLP, the utilization of deep learning has delivered results that have effectively outperformed those achieved by other AI and statistical techniques utilized for a long time earlier (Iqbal et al., 2019). In this section the techniques and algorithms associated with the field of text generation are discussed.

2.1. Distributed representation

The approaches of distributed representation of input data is fundamental idea to deep generative models, particularly when applied to NLP problems. The non-distributive representation of input adds sparsity to it, which is inefficient in several ways. First, the dimensionality of data gets increased as the structure will increase, so the semantic information in which deep learning model tries to map the input data becomes difficult due to high dimensionality. Distributed representation of words in vector space boosts deep generative models to achieve better results in NLP tasks. One of the advantages of utilizing condense and low-dimensional vectors is computation: most of the deep generative systems don't perform well with extremely high-dimensional sparse vectors. The key advantage of the condensed representation is generalization control. If our dataset contains features that represent similar meanings, it is better to obtain a representation that describes these similarities (Goldberg et al., 2017). Word representation started at 1980 (Rumelhart et al., 1988). The idea was then applied to statistical language modeling (Bengio et al., 2003), followed up by many NLP tasks (Collobert and Weston, 2008; Glorot et al., (ICML-11), 2011,). Many models for obtaining distributed representation of input have been produced in recent years (Word2Vec (Mikolov et al., 2013) Node2Vec (Grover and Leskovec, 2016) Gene2Vec (Du et al., 2019)). Since NLP is a prime option for designing complex Natural Language (NL) tasks. At the beginning of any NLP tasks, when we try to learn joint probability functions of language models there is the problem of dimensionality, so it is always necessary to understand the distributed representation of text in a low dimensional space. Word embedding

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

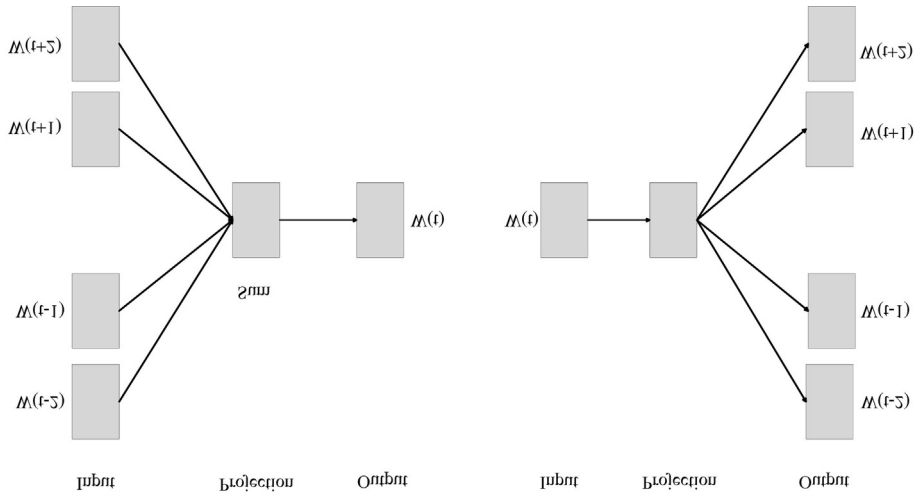

Fig. 1. (Continuous bag of words architecture on left, and skip gram on the right).

(Goldberg and Levy, 2014) is the mapping of a discrete categorical variable to a vector of continuous numbers. In the context of Artificial Neural Networks (ANNs), embeddings are low dimensional learned continuous vector representation of discrete variables. Word embeddings actually follow the distributional hypothesis where words having similar meanings occur in a similar context. Thus these vectors try to capture the characteristics of words that are closer to each other. Distributional vectors actually capture similarity (cosine similarity) between words. The significance of word embeddings in the field of deep learning turns into noticeable by considering the number of researchers in the field. One such research in the field of word embeddings led by Google prompted the advancement in the group of related techniques or algorithms commonly known to as Word2Vec.

2.1.1. Word2Vec

The Word2Vec method has feed-forward fully connected ANN architecture (NNSS, 2017). The Word2Vec predicts the target word from the given context of words (Continuous Bag of Words (CBOW)) or predicts the context from the target word (Skip-gram model) as shown in Fig. 1. The concept of Word Embedding was revolutionized by Mikolov et al. (2013) and Mikolov et al. (2013) who proposed CBOW and Skip Gram model. CBOW model determines the conditional probability of target word on giving context words, while a skip-gram model does exactly opposite which determines surrounding context words by giving central words. The assumption is taken that context words are symmetrically located to the target words at the distance equal to the size of the window in both directions. When the dimensions of the word embeddings get increased the prediction accuracy also increases. The inability of word embeddings to represent phrases is one of its limitations. For example "hot potato" is not considered as the group of two words. One among the famous results of Word2Vec is "*King − Man + Woman = Queen*".

2.2. Glove & FastText

The problem of Word2Vec is that it relies on local context of sentences, which means it captures only semantic information of language. However, this can be suboptimal sometimes. Glove (Global Vectors) (Pennington et al., 2014) on the other hand, captures both the global context as well as the local context of vocabulary while changing words to vectors. However, each type of measurement has its own advantages. For example, in the case of analogy tasks, Word2Vec does very well. The Glove works on the co-occurrence of words. The intuition behind GloVe embeddings is the co-occurring matrix which is decomposed using neural networks into thoughtful and dense vectors. Since Glove vectors are faster to train than Word2Vec vectors. Both of them have failed to provide definite results in the context of distributed representation.

Another method of representing words into vectors is FastText (Bojanowski et al., 2017) which is an extension of Word2Vec. This method represents each word as an n-gram of characters rather than representing words directly. This technique is helpful in order to capture semantics of small words. This method has the advantage of representing rare words that may not have been seen in training time. FastText has provided better results than Word2Vec on different measures. The representation of words to vectors has strong impact on the output produced by the deep generative models so, it is necessary to use efficient algorithm of word representation.

2.3. Algorithms used in deep generative models

The popular algorithms which are frequently used for generation of text are listed here:

2.3.1. Recurrent Neural Networks(RNNs)

Recurrent Neural Networks (RNNs) are the most powerful algorithm for NL problems specifically when modeling the sequential data. Since RNNs contain internal memory due to which it is able to remember the previous input as well as current input that makes sequence modeling tasks lot easier (Sherstinsky, 2018). The output at any time step does not only depend on current input but also on the output generated at previous time steps, which makes it highly capable of tasks like language generation, language translation, sentiment analysis, etc. In ANNs all inputs are independent of each other, but in RNNs there is dependency among inputs. Fig. 2 shows the unrolled RNNs, the output h_t at particular time-step "t" is given by the equation:

$$h_t = \sum_{t=0}^{T} X_t W_t + U h_{t-1} + b \tag{1}$$

h_{t-1} is previous output and X_t is current input and W_t represent weight at time step t, also U represent weight associated with output h_{t-1} and b represents bias term.

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

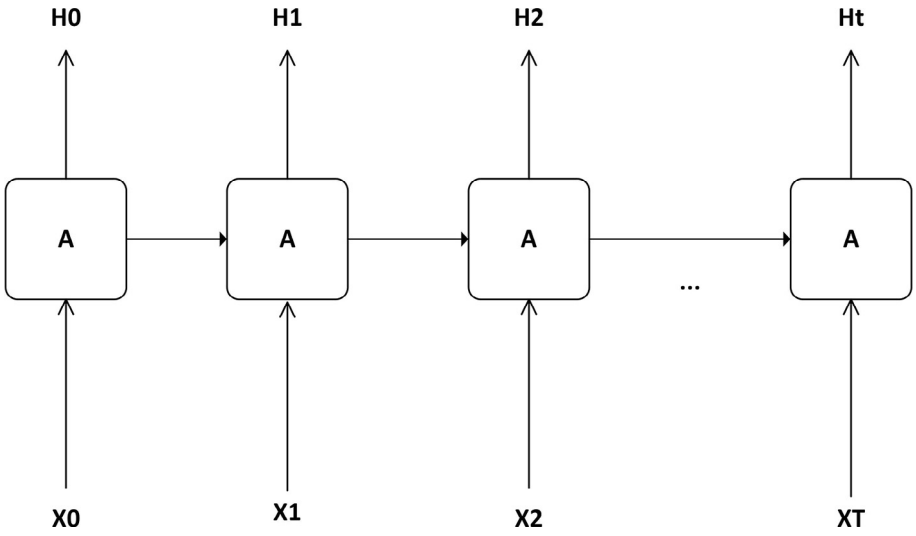

Fig. 2. Unrolled Recurrent Neural Network:.

At each iteration t the input X_t is fed to the network where the h_t is calculated based on previous output h_{t-1}. The RNNs are used for text generation due to its sequence modelling capability. The capability of RNNs to model the sequential tasks lies in its high dimensional hidden state and its non-linear dynamics. But it has been seen training of RNNs is very difficult, which hinders its use in many NLP tasks (Bengio et al., 1993) (Pascanu et al., 2013). Due to this, despite its sequential modeling capability, there is little research seen in this area for the last 20 years (Sutskever et al., 2011). The training issues related to RNNs are modeling long term dependencies: exploding/vanishing gradient, exploding gradient; that is when the algorithm assigns high results to weights which makes the model learn nothing similarly in vanishing gradient; where the values of the gradient are too small and the model stops learning. To solve such issues, the variants and advancements in RNNs were made in recent times (Salehinejad et al., 2017).

2.3.2. Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU)

LSTMs (Hochreiter and Schmidhuber, 1997) inherits the same architecture as vanilla RNNs, without hidden state. The memory units in LSTMs are called cells that take the combination of previous state and current input as input. These cells actually decide what to keep in memory and what to eliminate. Suppose, we check online reviews to determine whether we want to buy any food item or not. let's say one of the reviews is *(Amazing! This box of food items gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but definitely, be buying again!)*. After reading the review, our brain subconsciously remembers only important keywords. we remember words like "amazing" and "perfectly balanced breakfast". Words like "gave", "all" "should" we don't bother to remember. If our friend asks us the next day about the review, we probably wouldn't remember the whole review, we might remember the important points. This is exactly what LSTMs and RNNs do. It only learns to keep relevant information to make predictions and forgets all the non-relevant data. This type of mechanism overcomes the problems mentioned above. The LSTMs contains memory states (previous state, current memory, and the input) which are combined to solve problems like vanishing/exploding gradient. Many LSTM based text generation models have been proposed. (Pawade et al., 2018Chen et al., 2019Wang et al., 2019). GRU is another extension of standard RNNs (Cho et al., 2014) which modifies LSTM architecture with a gating network, which generates signals that control the present input and previous memory to update the current activation and current network state. It is simpler than LSTM in which the parameter updation is also used for gates according to the algorithm. Many deep generative architectures have used GRUs for the generation of text (Mangal et al., 2019; Hong et al., 2018).

2.3.3. BiDirectional RNNs (BRNNs)

The choice of algorithm is an important factor for designing any deep learning model. Many deep generative models have been proposed where BRNN is used to generate the sequence of outputs (Berglund et al., 2015). The idea behind BRNNs is that the output at time-step t may not only depend on previous elements but also on the future elements of the sequence. These are simply composed of two independent RNNs (Schuster and Paliwal, 1997). To examine this, the output of two RNNs must be combined, where one executes the process in the forward direction and second runs the process in the backward direction as shown in Fig. 3. The input to the first RNN is given in normal time order while as in the second one as reverse time order. This structure allows the networks to have both backward and forward information about the sequence at every time step.

The output at time step t is calculated as:

$$Y_t = g(w[h_{tf}, h_{tb}] + b) \tag{2}$$

h_{tf} and h_{tb} represents hidden state at forward and backward direction respectively, w represents weights associated with them, b represents bias, which helps the model to fit to the given data and g is the activation function used to add the non-linearity into the network. Another extension of RNNs that have extended the capability of neural networks by adding them external memory source, which can be interacted by the attentional mechanism are called Neural Turing Machines (NTM) (Graves et al., 2014). Unlike with LSTM where the memory is stored at a hidden state, and NTM has external memory that stores it.

2.4. Power of Convolutional Neural Networks (CNNs) for text

Convolutional Neural Networks (CNNs) one of the popular algorithm used for computer vision. These were the main driving force for image classification and for most computer vision systems, when talking of Facebook's automated photo tagging to self-driving cars (Britz, 2015). The research on CNNs applied for NLP

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

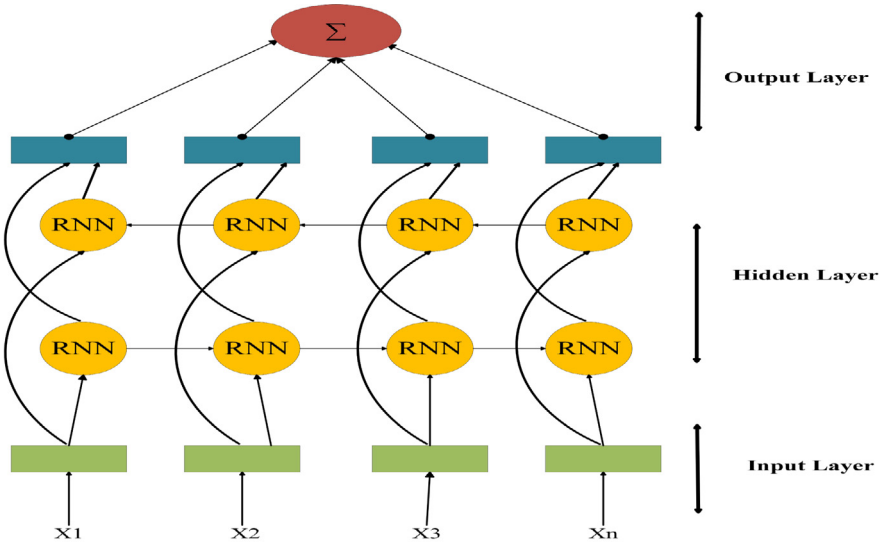

Fig. 3. BiDirectional Recurrent Neural Network:.

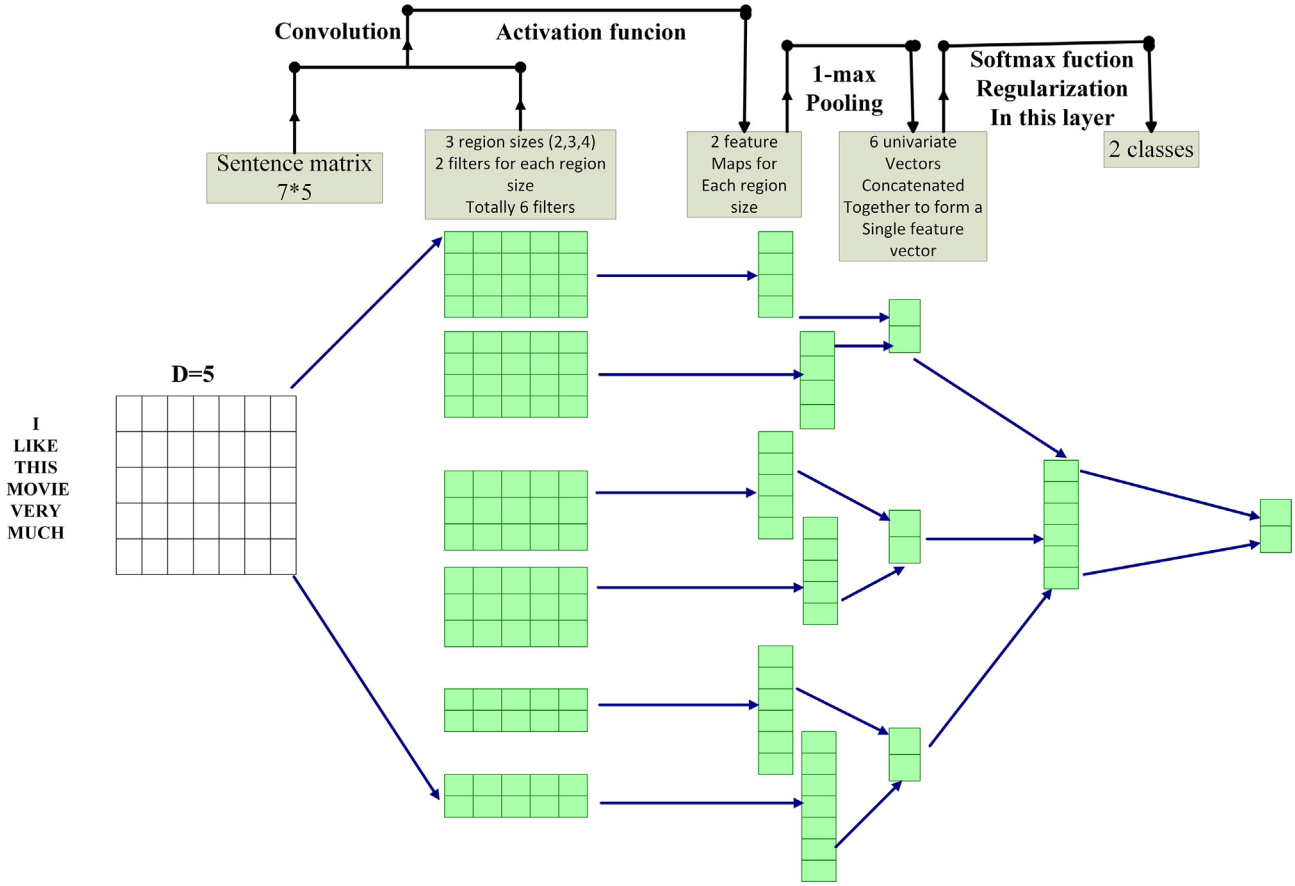

Fig. 4. Convolutional Neural Network for NLP (Britz, 2015).

tasks have been started recently and have achieved interesting results (Lai et al., 2015). Many methods have been proposed for text classification using CNNS (Jacovi et al., 2018; Xu et al., 2015). Unlike in computer vision problems where image pixels are taken as input, NLP tasks instead of image pixels use sentences, words or sometimes characters depends on problem classification. So each row is a vector that represents a word. Typically word embeddings or it could be one-hot vectors that will index the word into a vocabulary. For 10 sentences using 100- dimensional embedding will have a $10 * 100$ matrix as input. In computer vision as the filters slide over local patches of an image, here in NLP the filters slide over a full row of words (matrix). The width of the input matrix is the same as the width of the filters. The illustration of how CNNs are used for text is shown in Fig. 4, where cnns are used for sentence classification. The three filter regions are characterized as: 2,3 and 4, and every filter region contains two fil-

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

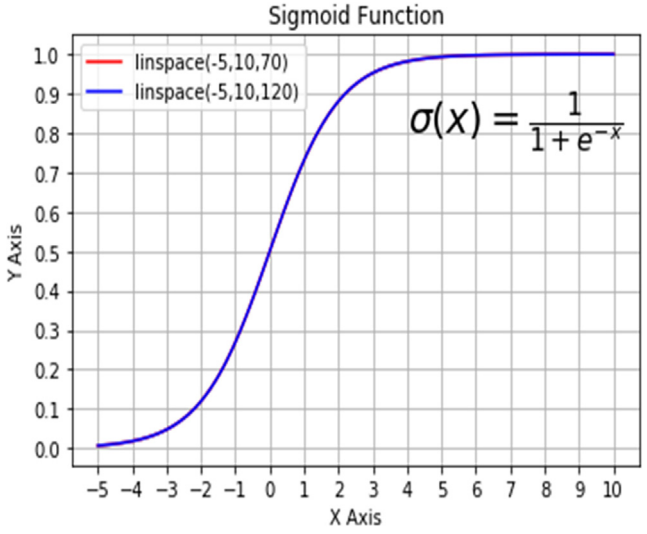

Fig. 5. Graph of Sigmiod.

ters. The convolution operation is performed on sentence matrix and variable-length feature maps are obtained, the max-pooling is applied on each map, which gives the largest number from each feature map. So, the univariate feature vectors are generated from these six maps and they are then combined to form one feature vector for the penultimate layer. Finally, the softmax layer receives this feature vector as input and then sentences are classified, here the binary classification is assumed so two possible output gets generated (Zhang and Wallace, 2015).

2.5. Activation functions used in generative models

The performance of deep generative models highly depend on the choice of activation function. They help the network to learn non-linear properties, without that the network will behave like a linear function. The implementation of the activation function makes the network learn complex problems. Since, the network is trained using backpropagation (Rumelhart et al., 6088) where there is the need to learn the gradients for parameter updation. So, the choice of activation function depends on the differntiability of it also. The commonly used activation function in deep generative models are:

Sigmiod:- For generative models to classify outputs, sigmiod activation function is used. This function ranges from 0 and 1. Despite its easy to understand and apply, it has some limitations that have made it fall out from popularity, that is its output is zero centered and shows slow convergence (Sharma, 2017). The graph of sigmoid function is shown in Fig. 5.

Relu:- Different activation functions performs well for different deep generative architectures. One amoung the many popular activations functions is Relu and Leaky Relu. It has been seen Relu converges six times faster than TanH function (Gupta, 2017). It reduces the vanishing gradient problem and is used almost in every deep generative model. Relu activation works by shortening negative values to 0. This has the impact of obstructing the slopes to move through the network. In order to prevent function for being zero. Leaky Relu enables some negative value to pass through, which helps the function to calculate the high values between the features and a little factor (Dansbecker, 2018). Limitations in Relu seen so far are that it is only used in hidden layers of neural network and some gradients become weak during training and can die, which means that Relu can result in dead neurons. So modification of Relu called Leaky Relu was introduced which uses small slopes to keep updates alive. In these cases, the gradients are completely shut to backpropagate. This is actually good for GAN models as the generator has to maintain only one method to learn by receiving the gradients from the discriminator. The choice of activation function depends upon the architeture of deep generative model. The graph of both the functions is shown in the Fig. 6.

2.6. Optimization techniques for generative modeling

The main goal of deep learning model is to find minimum which generalizes well. The optimization techniques help us to find out the minimum of an objective function (error function). Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951) has been extensively applied to deep learning problems. SGD has one of the advantages that they are very simple and converge fast for the problems of having a large training set. However, it possesses many disadvantages as well, like they require manual tuning of optimization parameters such as learning rates, convergence constraints, etc. One good strategy would be to run the learning algorithms with many parameters and pick out the best parameters that will make the algorithm to generalize well. One more difficulty in the SGD algorithm is that it is implicitly sequential, which makes it difficult to parallelize them with GPUs or distribute them with computer clusters (Le et al., 2011). The SGD algorithms use same learning rate throughout the network which if sometimes chosen very small, makes parameter updation very slow and will take very long time to achieve acceptable loss, or if the learning rate is set too large then the parameter will move all over the net-

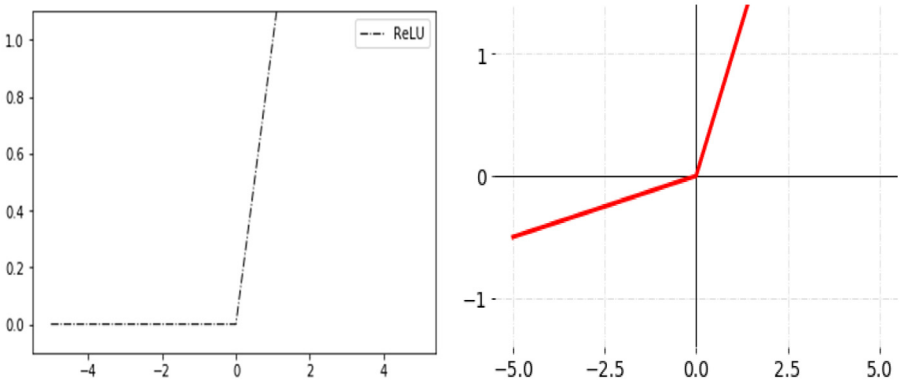

Fig. 6. Graph of Relu on left and Leaky Relu on Right.

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

work and makes it difficult to converge well. Therefore learning rate should be large in some dimensions and small in many dimensions. One obvious way to address this problem is to set different learning rates for each dimension, but deep learning models contain thousands or millions of dimensions, which would not be practical. So earlier methods to mitigate this problem were the AdaGrad algorithm (Duchi et al., 2011) which adaptively scaled the learning rates for each dimension. Since there is no generic right value for the learning rate, it comes by experimentation and intuition. The concept of momentum was introduced in the standard gradient descent (Ruder, 2016) which restricts the oscillation in one direction so that to make the algorithm converge faster. The RMSprop (Tieleman and Hinton, 2012) another optimization algorithm works in the same way, it restricts the oscillations in the vertical direction so that the algorithms take larger steps in horizontal directions which makes the algorithm converge faster. Adam stands for Adaptive Moment Estimation is another method that computes adaptive learning rates for each parameter. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and Natural Language Processing (NLP)(Kingma and Ba, 2014). Adam works well in practice and compares favorably to other adaptive learning methods as it converges very fast and the learning speed of the model is quite fast and efficient and also it rectifies every problem that other optimization techniques have faced, such as vanishing learning rate, slow convergence or high variance in the parameter updates which leads to fluctuating loss function.

3. Recent deep learning techniques used for text generation

Deep generative models are not only popular to study how well the model has learned, but also to learn the domain of the problem. The most popular techniques for the generation of text in deep learning era are Variational Auto-Encoders (VAEs) (Kingma and Welling, 2019) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014).

3.1. Variational Auto-Encoders (VAEs)

The power of most deep learning model depends on cleanly labelled data. Since most of the data is unlabelled or unstructured in nature. The popular deep learning models require large quantities of structured data for training. Labelling the unstructured data is very time consuming. One way to address this issue is to make use of unsupervised methods to train on data without labels. Variational Auto-Encoders (Kingma and Welling, 2013) is one of the powerful deep generative model that works on unlabelled data. It contains an encoder that encodes data into latent variables, and then the decoder decodes these latent variables to reconstruct the data. The encoding operation takes input x and produces output latent space $p_\phi(z|x), \phi$ represents the parameters of encoding operation, while the decoder exactly does the opposite. It finds the probability distribution $q_\theta(x|z)$ of data on given latent distribution, θ represents the parameters of decoding operation (Altosaar, 2016). Also ϕ and θcan be treated as the weights of encoding and decoding operation. The loss function which forces the model to learn the rich representation of latent space can be broadly defined as the sum of two terms: *Loss Function = Reconstruction loss + Regularizationterm*. Reconstruction term is simply the mean squared error between the input and output data. But the regularization term here minimizes the distance between latent distribution $p_\phi(z|x)$ and some prior distribution $p(z)$. Since VAEs learn the probability distribution of data with the help of latent space that makes it suitable for generating new data. The divergence between encoder's distribution $p(z|x)$ and posterior distribution $p(z)$ is measured by using Kullback–Leibler (D_{KL}) divergence. Mathematically the loss function is given as:

$$L_i(\theta,\phi) = -E_{z\_>q\theta}(z|x_i))[logp_\phi(x_i|z)] + D_{KL}(p_\phi(z|x_i)||p(z)) \quad (3)$$

$L_i(\theta,\phi)$ *defines the reconstruction loss at data point i,*

$-E_{z\_>q\theta}(z|x_i))$ *measures the expectation of encoder's distribution over representation*

$D_{KL}(p_\phi(z|x_i)||p(z))$ *describes the divergence, of* $(p_\phi(z|x_i)$and $p(z))$.

The block diagram of Variational Auto-Encoders is shown in Fig. 7: The input data x is sampled in latent space in the form of standard deviation σ and mean μ, then the stochastic sample of z is predicted from this distribution. Finally, the sample z is then decoded to generate the output of $x\prime$. The VAEs have developed as one of the popular way to unsupervised learning of complex distributions. The applications of VAEs for the generation of discrete data (text) is limited. The main issue of using VAEs for text generation is KL collapse (means, when the decoder becomes more powerful than the training objective, can be solved with the false strategy), that is the decoder produces the output regardless of latent space. If KL term is zero the posterior probability is independent of input data (Lucas et al., 2019). The recent attempt of text generative model on VAEs was proposed by Bowman et al. (2019) which utilizes recurrent neural networks to captures the universal features of sentences (e.g., topic, style) in continuous variables. The issue of the posterior collapse was also observed here. Some methods were proposed to alleviate this issue like Bowman et al. proposes the concept of KL-annealing (the entire sentence is incorporated into distributed latent space) and word dropout (removing some of the information during learning) (Bowman et al., 2016). This factorization helps to model some properties of sentences like style, high-level semantic features. More specifically the weights of the network get increased at the training phase and the random replacement of word token makes decoder rely on global representation z instead of the learned language model. However this technique could not fully alleviate the KL collapse problem, therefore many subsequent contributions were made to find the better techniques of alleviating it. Text generation models often use the technique to generate text based on the previously generated tokens x_t. The output generated with this approach could not fill the diversity (topic, style, semantics, etc) in generated sentences. Many different approaches have been proposed so far, the recent technique for text generation and addressing the KL collapse was proposed by Yang et al. (Yang et al., 2017) which replaces the RNN decoder with dilated CNN (Yu and Koltun, 2015) and simplifies the control of contextual capacity by changing dilation. Precisely, the previous techniques for text generation were based on modeling the joint probability $p(x)$ directly. This paper proposed to model $p(x)$ as the marginal distribution. They initially generate continuous latent space z based on the prior distribution $p(z)$ (multivariate Gaussian). Afterwards the generation of sequence x from a conditional distribution $p_\theta(\frac{x}{z})$ was parametrized by decoder. This helps to integrate the latent variable to balance the generation of whole discourse and makes it better to obtain high-level features of variation in the data. Due to the recurrent nature of RNNs, they are considered as a positive algorithm for text generation. But the core difficulty of alleviating posterior collapse needs some alternative approach. The new model was proposed by Semeniuta et al. (2017) as Hybrid VAE, where instead of LSTMs, Conv and De-Conv neural networks were used as encoder and decoder respectively. The auxiliary loss J_{aux} represented as:

$$J_{aux} = -\alpha E_{q(z|x)} logp_\phi(x|z) \quad (4)$$

J_{aux} forces the decoding process to depend on latent representation z for the optimization of ELBO (Evidence Lower Bound). α

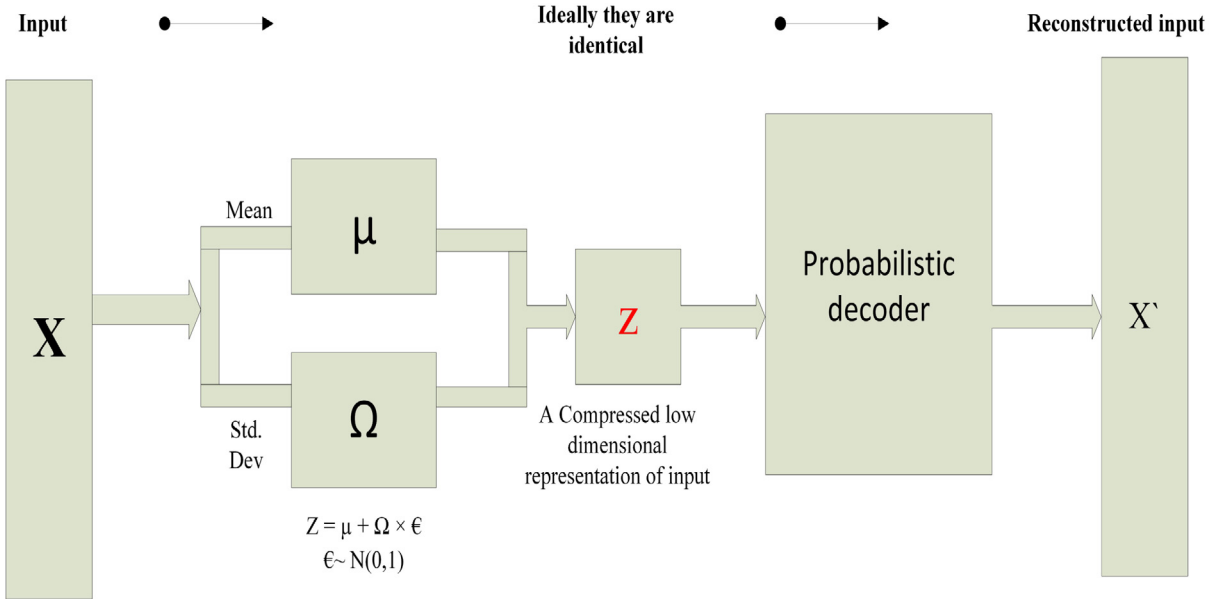

Fig. 7. Block Diagram of VAE.

controls the penalty of auxiliary loss, ϕ represents weights used in decoding process. It was demonstrated that hybrid VAE converges better and quickly than LSTM VAE. However model was able to address the problem of latent loss, but the difficulty of generating long text sequences was experienced in this approach.

The training of deep learning models is always challenging and VAEs are not different in this case. However many attempts have been made to optimize this difficulty. One such method was proposed by Kim et al. (2018). Variational Inference (VI) is the method of making the computation of network controllable. Traditional VI methods were to repeat over observed data and update the model parameters in closed form. However this method of parameter updation requires model which are conjugant. The implementation of this approach to non-conjugate models was very challenging. Stochastic Variational Inference (SVI) (Hoffman et al., 2013) and Amortized Variational Inference (AVI) (Rezende and Mohamed, 2015) are the methods used for parameter selection proposed in recent times which scale to huge training sets and non-conjugate models. This technique of predicting model parameters is helpful for VAEs when used as generative models. However, finding the local optima in the case of SVI was easy but the process of optimizing each data point is difficult. On the other hand, the inference of AVI is too fast, but the nature of input data as the parametric function makes it too strict of restriction, also the learning of AVI was in that way where the parameter updation can be done on sub-optimal variational parameters. The recent contribution in the field of text generation uses both AVI and SVI is known as *Semi-Amortized Variational Auto-encoders* (Kim et al., 2018). The initial variational parameters were chosen from the inference network, after that SVI runs to refine them. This approach, however, surpasses other autoregressive generative models but improvement for addressing the main issue in the field of text generation *KL Collapse* was not seen. The process of writing complex text from scratch in a single pass is difficult even for humans. Inspired by this process another method of generating text was proposed (Guu et al., 2018) called *Prototype then edit model*. The process begins like, It first examples a random model sentence from training corpus and afterward conjures a neural editor which draws a random edit vector and produces another sentence by taking care of the model while molding on the edit vector. However, the samples generated were competitive but there was no contribution in this method regarding the problem of "posterior collapse".

Table 1
Comparison of recent VAE models on text generation.

	Perpelexity	Negative log likelihood	KL
VAE (Bowman et al., 2019)	60.1	380	15
Improved- VAE (Yang et al., 2017)	63.9	332.1	10
Hybrid- VAE (Semeniuta et al., 2017)	*	*	12.5
Semi Amortized-VAE (Kim et al., 2018)	60.4	**327.1**	7.19
Neural Editor-VAE (Guu et al., 2018)	**26.87**	*	*
Skip-VAE (Dieng et al., 2018)	60.55	*	**22.54**

The method of alleviating the problem of KL collapse is still an active area of research. Kim et al. (2018) proposed the new way of initializing the parameters of variational auto-encoder by amortized inference and then applied stochastic inference to refine it. The skip connections between latent variable z and decoder introduced by Dieng et al. (2018) that enforces the relationship between latent variables and reconstruction loss. Since both of the methods through experiments justified there betterment than previous methods. The recent exciting advancement was proposed by Guu et al. (2018) where instead of Gaussian distribution, Von Mises-Fisher distribution (Yasutomi and Tanaka, 2014 is used. This method was further explored by Xu and Durrett (2018)). Variational Auto-Encoders with such incorporation is called Hybrid VAE and it can control KL term by hyper-parameter k which helps in addressing KL collapse. The comparison of different text generation models based on VAEs is shown in Table 1.

The table highlights some modifications in VAE for text generation tasks where many improvement have been made in its mechanism. The parameters chosen here for these models are perplexity, negative log-likelihood, and KL term, which is highlighted in this table. checking the performance parameters of these models seems hard to choose one among them. So research on VAE to use for text generation tasks is still an active area.

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

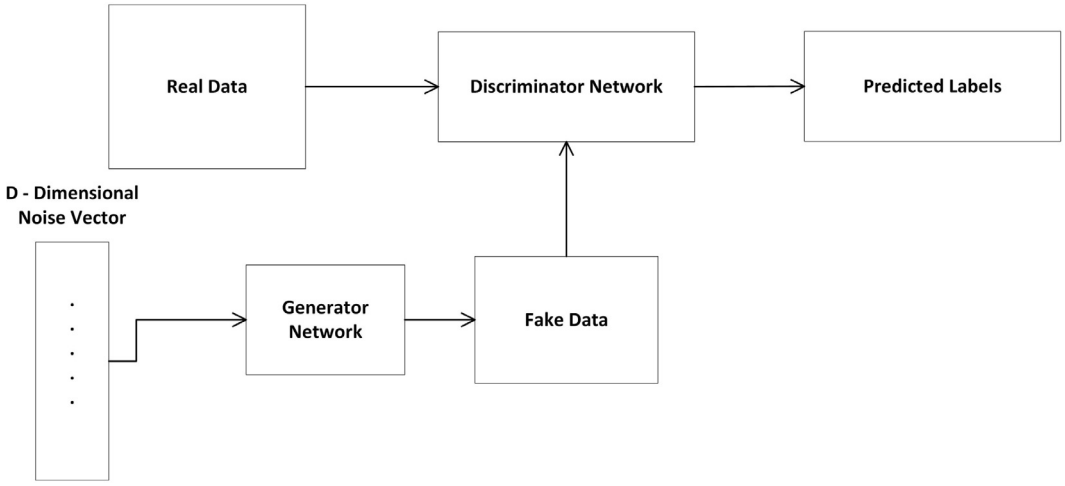

Fig. 8. Block Diagram of GANs.

3.2. Generative Adversarial Networks (GANs)

Deep learning has changed the way we work, we compute, we analyze and make our lives easier. We have taught machines to figure out things for themselves, many deep learning architectures can be credited for its creative success. In spite of that, no major success has been constituted by deep generative models which are due to their inability to approximate intractable probabilistic computations. But the solution is found that could bypass these problems faced by generative models called Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). GANs are the popular deep learning algorithm that takes up an adversarial approach, dissimilar from the conventional neural network. GANs contains two models that are trained in an adversarial way. First, the generator generates the data samples and discriminator which classifies these data samples as real (training data) or fake (generated by generator) as shown in Fig. 8. The goal of the generator is to generate samples that are very close to the true data so that it can fool the discriminator and the goal of the discriminator is to classify accurately these two types of data samples. Since it's like a game-theoretic approach, where the objective function is represented as a part of the minimax function. The discriminator D tries to maximize the objective function, while the generator G tries to minimize the objective function. In other words, D and G try to play the minimax game having value function $V(G,D)$:

$$min_G max_D V(G,D) = E_{x->p_{data}(x)}[logD(x)] + E_{z->p_z(z)}[log(1-D(G(z)))] \quad (5)$$

Since GANs have shown remarkable results for generation of images (Pix2Pix GANs, Conditional GANs, Westerrrian GANs, etc.). The training of GANs for text is more challenging because of the non-differentiable nature of discrete symbols. The earliest attempts for generating meaningful sentences (Li et al., 2017) used the concept of Maximum Likelihood Estimation (MLE) (a technique to find the parameter values that maximizes the probability of process defined by the model). Even though it is successful, but this training objective has some issues like responses that are shallow, repetitive, short-sighted. So to solve these issues, some clarification on many areas is needed like which ideas define standard communication and techniques and how they can be merged with deep learning.

Many methods have been proposed (Li et al., 2016; Li et al., 2017). Physically characterization of few aspects (acknowledgment, informativeness, and consistency) and incorporation of reinforcement learning structure to train and generate highly rewarded sentences was discussed. However, manually defined reward functions cannot cover all correct information and may lead to low-quality statements. A good generative model should generate sentences that are indistinguishable from human-generated. Using GANs for NLP tasks has not achieved comparable success. The reason is that the text generation process is discrete, which makes output error hard to back-propagate to the generator. *For example: While using backpropagation it is convenient to make parameter updation in image pixel as* $1.011 + 0.01$ *but for text, it is meaningless to make updation like parrot* $+ 0.01$. How GANs can be used for better text generative models and how the issues related to this can be addressed is an active area of research. One improvement in recent times was proposed called *Professor Forcing* (Lamb et al., 2016). the distribution of two sequences (Training Sequence and Generated Sequence) was compared on variable-length inputs. The demonstration was made, that the discriminator not only looks for single-step prediction but also at the statistics of the behavior. Providing the discriminator central hidden values of the generator results in a differentiable model and accomplishes good results in various NLP problems like sequence generation and acoustic generation.

The well-known technique for solving NLP tasks is maximizing the probability of every word in labeled data conditioned on previously calculated output. The problem with this approach is *exposure bias* (where the system gets more exposure to ground truth data and fails to generate meaningful sequences at test time). The solution to this problem proposed by Bengio et al. 2015 used the concept of scheduling sampling (a training strategy, where the model gets explored more during training stage which makes it more robust to deal with the mistakes made during inference) was used (Bengio et al., 2015). The idea is to partially feed the generative model with synthetic data while predicting the next word in the learning stage. This technique was regarded as an inconsistent training strategy with errors not backpropagated through sampling decisions and also there was no improvement seen to address the problem of exposure bias.

Reinforcement learning To address the problems discuused above, the concept of reinforcement learning was used in deep generative models (Li et al., 2018; Shi et al., 2018; Dethlefs and Cuayáhuitl, 2010). The idea of reinforcement learning used in deep generative models has shown promising results. Reinforcement learning is the area of deep learning where the model learns by interacting with the surroundings and rewarded produced to perform actions (Skymind, 2018). The reinforcement learning

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

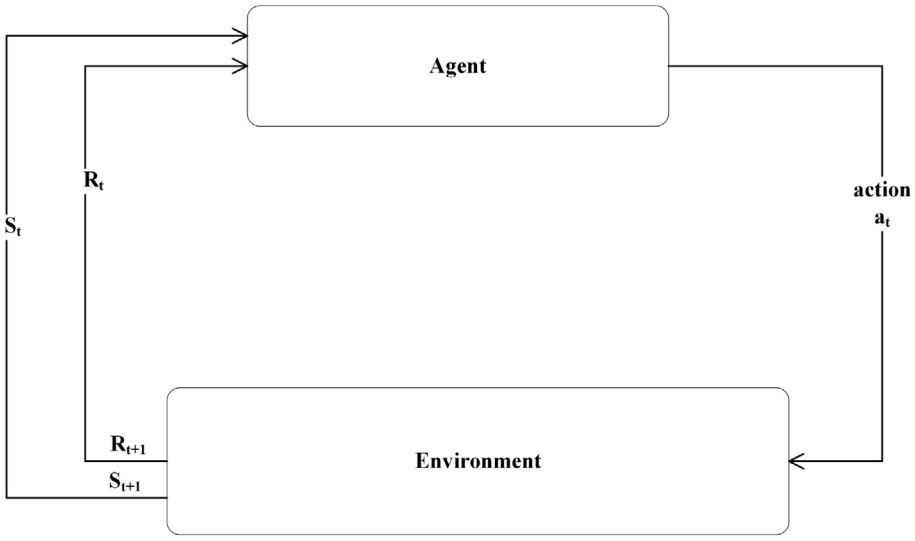

Fig. 9. Block diagram of Reinforcement Learning.

algorithms can be understood by using the concept of *agents*, *environments*, *states*, *actions*, and *rewards* :-.

- Agent:- An agent is someone who takes actions; for example, Sophia playing guitar
- Action:- All possible moves, an agent can make is considered as an action, like in a video game action can be moving right, moving left, standing still.
- Environment:- Space where an agent learns, here agent takes the current state as input and outputs the reward and its next state.
- State:- explicit and quick circumstance agent operator gets itself, it might be solid spot and point, an unconstrained design which places the agent in connection to other critical things, for example, apparatuses, hindrances, foes or prizes, it might be the present circumstance returned by environment or any future circumstance.
- Reward:- A reward is an input with which we measure the achievement or disappointment of the agent's activities, for instance in a computer game when an agent makes contact with the coin he wins points. The agent from some random state sends output as activities to the environment which gives the agent a new state just as rewards. The way of determining the next actions based on the current state. It maps states to actions, actions that promise the highest reward, as shown in Fig. 9.

This is the key idea used when modeling text generation as the reinforcement learning problem. This concept was first studied by Bachman et al. (2015) which determines that sequence generation problems may be formulated as the sequential decision-making problem. Inspired by this and the concept of reinforcement learning an alternative way of training generative model was proposed (Bahdanau et al., 2016). The additional network "*Critic*" is trained to output the value of each token is known as expected task score. These predicted outputs are used to train the main sequence prediction model "*Actor*". The core idea of this approach is that, under the presumption that the critic calculates the exact output values, the explanation used to train the actor is a neutral measure of the gradient of the expected task-specific score. But using the concept of reinforcement learning in GANs for text generation needs to answer any questions. How the rewards can be generated to guide the generator. Developing methods to answer this question is an active area of research. One naive approach is to use the task-specific score as our reward (Bahdanau et al., 2016). However such task-specific score is hard to find sometimes. Yu et al. (2017) propose sequence GANs (SeqGAN) which uses the prediction score of the discriminator to guide the generator. The generator G_θ generates sequence $s\text{-}p_G$ which discriminator predicts as real (high reward) or fake (low reward).

Presently, the objective of the generator model (policy) is to produce a sequence from the initial state s_0 so that the expected end reward gets maximized. R_T is the end reward that is controlled by D. Precisely, as given in the equation:

$$J_\theta = E[R_T|s_0,\theta] = \sum_{y_1 \epsilon Y} G(y_1|s_0).Q_D^G(S_0, Y_1) \quad (6)$$

Y represents vocabulary. Now, the expectation of obtaining the end reward R_T conditioned on initial state s_0 and θ (generator parameter) is the product of all possible values of reward (2nd term) and the probability of the value occurring (Ist term). The 2nd term represents the action-value function, which returns the reward value for action y_1 from the initial state s_0 having policy G. This action-value function predicted by the discriminator D gives reward for only the completed sequence. But it is necessary that the fitness of both previous words and the generated words should be considered. For that purpose, the Monte-Carlo (A technique used in game theory where starting from random actions and iterating repeatedly till the terminal state is reached) (Smith et al., 2013) search gets applied to roll-out the current policy in order to measure the reward. At the end of the sequence, the discriminator D predicts the score which is accumulated over each node of the network. The generator uses the current learned policy network to roll-out many times until sentences get completed to obtain the estimated reward. The performance of GANs is highly affected by the training strategies of generator and discriminator. The training strategy of SeqGAN and that of standard GANs is different because SeqGAN requires pre-training of the generator on target corpus before adversarial training.

As mentioned earlier, when output is discrete it's hard to back-propagate the gradient in the network. To address this issue the generative model is treated as a stochastic parametrized strategy where Monte Carlo is used to surmise the state value. The idea used to train the policy gradient automatically reduces the problem of differentiation in regular GANS for discrete data. Indeed, even with rigorous pre-preparing, it was seen that policy has prob-

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

lems to get positive and stable reward signals from the discriminator. To address these issues, MaliGAN (Maximum-Likelihood Augmented Discrete Generative Adversarial Networks) was proposed (Che et al., 2017). Inspired by Norouzi et al. (2016) the normalized maximum likelihood optimization is used to overcome the difficulty of back-propagating rewards. The utilization of many reduction techniques to optimize this objective. This new objective however provided better training signals even if discriminator shows low optimality, but was prone to overfitting. Since the output of the discriminator network is binary (0,1). For the sequence generation problem, the binary prediction becomes too restrictive. Since the diversity and the content richness of the generated sequence lacks behind due to binary classification. The new adversarial training is known as RankGans (Lin et al., 2017) was proposed to generate high-quality text descriptions. Instead of classifying the output between 0 & 1 its ranked form 0 & 1 according to the diversity of generated output. The model learns from this relative ranked information between human-generated and machine-generated. Precisely the adversarial framework consists of two networks *Generator* and *Ranker*. The ranker is trained to rank the machine-generated sequences lower than the human-generated, while the generated is trained to generate sentences that fool the ranker. To overcome the problem of the non-differentiability policy gradient method is used. Despite its increased BLEU (Papineni et al., 2002) score, there are still a few challenges that limit its use in practice.

1 The use of a scaler as a score may not be descriptive enough to teach the generator because it cannot represent well the transitional structure of text during its generation.
2 To estimate the reward for intermediate sentences tends to be noisy and scanty, mainly in long text generation when the generator gets reward only after the entire sentence gets finished.

The main challenge of generating long text sequences is the sparsity of binary guided signal and it only gets provided when the whole sample is generated. Zhang et al. 2017 proposed that rather maximize the rewards from discriminator the generator *G* should be trained to learn the feature representation of the real text and generated text to be matched.

Inspired by this, LeakGAN (Guo et al., 2018) were proposed which combines feature matching and hierarchical reinforcement learning (Vezhnevets et al., 2017) to alleviate these problems. The hierarchical generator *G* consists of *Manager* (LSTM which is used as a mediator receives the feature representation from discriminator *D*) & *Worker* (which uses these features received from discriminator as the guiding signal for generator) This information from *D* is maintained internally and is named as *leaked* information. One important result of LeakGAN is that we can analyze that if the generator accomplishes the leaked information from the discriminator to generate data. All the methods discussed above tries to clout more information from the discriminator to have better quality text generation. Research on generating realistic-looking samples with adversarial training is growing rapidly. For instance, the idea of reinforcement learning by Yu et al. 2017; Li et al. 2017 (Yu et al., 2017) assumes the text generation as a sequential decision-making process. Despite the success of these methods, the two fundamental problems of GANs framework limit their use in practice.

1. *Mode – Collapsing* :- Mode collapse is a similar and related way of Generative Adversarial Networks to fail. In mode collapsing, the generator only learns one mode in a multi-modal distribution and chooses to always use that method to exploit the discriminator, e.g, if our training set contains dogs and cats, the generator try to generate wired cats but no dogs to fool the discriminator. This type of problem is categorized as mode collapse (Metz et al., 2016)
2. *Vanishing – Gradient* :- The problem arises when training the two models in an adversarial way, the discriminator converges quickly and the generator ends up learning nothing, this type of behavior is termed as vanishing gradient (Arjovsky and Bottou, 2017).

Inspired by the technique of hierarchical feature representation (Salimans et al., 2016) the new model was proposed by Zhang et al. (2017) known as TextGAN which uses LSTM as generator and CNN as the discriminator. The kernel-based moment-matching (a technique used to match the moments of two distributions) is used, which forces the generated and real sentences to have the same moments. The generator was trained to synthesize data that matches the empirical distributions of real data in the feature space by minimizing Maximum Mean Discrepancy (MMD), which can be understood to minimize the moments of two distributions. This approach boosts the model to learn features that are both instructive of the first sentences (utilizing the autoencoder) and discriminative w.r.t. generated sentences (through the discriminator). The initialization techniques (Weights of LSTM generator were initialized from a pre-trained CNN-LSTM Auto-Encoder) were also discussed to ease the training of generative adversarial networks. This approach alleviates the mode-collapsing issue associated with conventional GAN training. Since as mentioned earlier RNNs are the most popular generative model for text as well as for many NLP tasks. The problem of GANs to use them for discrete data is a highly active research area. To reduce the impact of the problems seen in conventional GANs when used for NLP tasks another variant of GAN called Mask-GANs (Fedus et al., 2018) was proposed, where the model is trained on a sequence infilling task. The objective of the model is to put the missing words of the sequence when a few words of the sequence were erased or amended. The objective of this model is to infill the missing words of the sequence with the goal that it would be discernable from the original sequence. While infilling this missing segment of sequence, the model works auto regressively over the words it has so far filled in, as in standard language modeling, conditioned by true known context. If somehow the entire sequence gets amended, at that point it decreases to language modeling.

Since deep learning models get trained on large datasets. so the data generated by generative models is the primary for enhancing data size to avoid this issue. The text generated is the promising mechanism for data augmentation like (Zhang et al., 2016; Semeniuta et al., 2017) which uses GANs for the generation of text and achieved state of the art results, The Generated data needs to be categorical (labeled) to overcome the problem of small datasets and to train better models. For the generation of labeled sentences a new technique was proposed called CS-GAN (Li et al., 2018) where RNNs used as a generator, also generator behaves like an agent which predicts the next character based on the current characters as in the process of reinforcement learning. So the ensemble of RNNS and reinforcement learning particularly tackled two challenges.

1. They generate realistic sentences with GANs given the discrete nature of the text.
2. Incorporation of category information in GANs to generate labeled synthetic data

Recently, to address the main two challenges (Mode Collapse and Reward Sparsity) in GANs for text generation the new method was proposed (Shi et al., 2018) where concept of Inverse Reinforce-

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

Table 2
Comparison of recent GAN models on text generation.

	BLEU2	BLEU3	BLEU4	BLEU5
SeqGAN	0.724	0.416	0.178	0.086
MaliGAN	0.755	0.436	0.168	0.077
RankGAN	0.686	0.387	0.178	0.086
MaskGAN	0.265	0.165	0.094	0.057
TextGAN	0.205	0.173	0.153	0.133
MLE	0.205	0.173	0.153	0.133
LeakGAN	**0.835**	**0.648**	**0.437**	**0.271**

ment Learning (IRL) (Ziebart et al., 2008) was used, which treats text generation as IRL problem, where the reward function learns to explain the expert behavior and then generated policy is learned to maximize the expected total rewards. The reward function aims to increase the rewards of the real texts in the training set and decrease the rewards of the generated texts. Intuitively, the reward function plays a similar role as the discriminator in SeqGAN. Unlike SeqGAN, the reward function is an instant reward of each step and action, thereby providing more dense reward signals. The generation policy generates a text sequence by sampling one word at a time. The optimized policy be learned by the "entropy regularized" policy gradient (Levine et al., 2016), which intrinsically leads to a more diversified text generator. To check the performance of generative models the similarity between two summaries (system summary and reference summary) is calculated. Many evaluation metrics were proposed to check the performance of deep generative models (Rouge, Bleu, etc). The problem with the evaluation metric BLEU is that generating the single sentence over and over again will give us a perfect BLEU score. The new evaluation procedure was proposed (Caccia et al., 2018) which proposes to use the sweep of temperatures for each model in order to compute the temperature curves in quality-diversity space, which provides open-air to the researchers, that is which model should be used for generating high quality-diversity samples. The temperature sweep concept can be treated as the cross-validation tool (which means early stop once the best curve has been achieved). However, the evaluation metrics discussed above outputs some errors like the one mentioned above. This also an inactive area for NLP practitioners. Table 2 shows the BLEU score of the above discussed GAN based generative models. All the models suffer from mode collapsing except MaskGAN. Both of the generative models show remarkable results, but till now it has been demonstrated that VAEs are well suited for text while GANs for images. However, we expect the better performance of both the models for "text" as well as for "images".

4. Text evaluation methods

Since for now the GANs were seen promising approach for the text generation. But the problem was to have a clear evaluation metric in order to check the potential of GANs for text generation. Some of the text evaluation methods are:-.

1. Recall Oriented Understudy for Gisting Evaluation (Rouge):- The similarity between the reference summary (Golden Summary) and the system-generated summary is checked on the basis of the Rouge Score. The Rouge is the set of metrics for evaluating summaries. (Lin, 2004) It actually works by comparing the golden summary with the system summary. Let us take an example, suppose we have a system summary as: *The boy was found playing cricket* and reference summary (Gold Summary) as: *The boy was playing cricket*. The number of overlapping words between the system and the golden summary is 5. This, however, does not give much information about the similarity of summaries. To calculate good results we actually calculate Rouge Precision and Rouge Recall,
Rouge Recall:- It means how much of the golden summary this system summary captures. It is calculated as

$$Recall = \frac{Similar - Tokens}{total - Tokens - in - Training - Summary} \quad (7)$$

Rouge Precision: It actually gives us the system summary which is relevant or needed and is calculated as:

$$Precision = \frac{Similar - Tokens}{total - Tokens - in - Generated - Summary} \quad (8)$$

2. Bilingual Evaluation Under-study (BLEU):- is the metric used to evaluate the generated sentence with a reference sentence. BLEU (Papineni et al., 2002) was originally used for machine translation systems and works by calculating the similarity between machine-translated text and reference text, where unigram or 1-gram is treated as one word while as Bi gram is treated as word pair. It actually works by comparing the N-grams of machine translation with reference data and count the number of matches. Therefore more matches results in better machine translation.

In-addition to above discussed text evaluation technqiues many methods (Banerjee and Lavie, 2005; Vedantam et al., 2015; Anderson et al., 2016) have been proposed for evaluation. However recent studies identified that the existing methods are poorly correlated with human judgment. The evaluation of text generation methods is still an open research problem. Since there was not an evaluation metric for GAN based text generation, and the metrics discussed above are based on N-gram overlapping are seen to have low correlation and low robustness (Semeniuta et al., 2018). The method of evaluating Generative Adversarial Networks with standard probability-based evaluation metrics was proposed (Tevet et al., 2018) where the prediction of the generative model can be seen as language modeling (LM) and a simple Monte-Carlo method is used for approximating it. The approximated probability distribution is then evaluated as the LM metrics such as Perplexity (the measurement of how well model predicts a sample) or Bits Per Character (BPC) (average number of bits needed to encode on a character). A particular concern for the text generation techniques mentioned in this paper is whether, in addition to the text quality, the model suffers from mode-collapse (lack diversity). Counting unique n-gram is a recent approach to calculate the diversity of text. Improving such an n-gram metric does not automatically improve the diversity as discussed in MaskGAN. The diversity of text generation is less dealt with the topic in the recent literature, although some very recent work is beginning to explore it.

5. Conclusion

Deep learning provides us a way to strap huge amount of computation and data with small efforts done by hand. With the help of word embeddings and popular models of deep learning like RNNs, CNNs, VAEs, GANs have made NLP problems a lot easier. The enormous increase in available data and computational power, also the developments in deep learning, have provided many new possibilities for researchers to analyze new applications for text generation. We expect such a trend to continue with more and better model designs. We expect to see more NLP applications to employ reinforcement learning methods to explore the research on text generation.

Generating natural language is a chalice of text generation research. The current deep learning models have not yet fully captured the nuances, technicalities, and interpretation of natural

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

language, which aggregates when generating longer text. The evaluation progress of text generation models requires a better metric carefully designed by the human study. This paper only summarizes two lines of text generation research. There may be other interesting methods in this category that could bypass the currently active models. The paper concludes with this statement that the area of continious data (images) is dominated by GANs while as the discrete data (text) is dominated by Variational Auto-Encoders

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Altosaar, J., 2016. What are Variational Autoencoders. URL: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/ [Online; accessed 19-January-2019].

Anderson, P., Fernando, B., Johnson, M., Gould, S., 2016. Spice: Semantic propositional image caption evaluation. In European Conference on Computer Vision. Springer, pp. 382–398.

Arjovsky, M., Bottou, L., 2017. Towards principled methods for training generative adversarial networks. arxiv.

Bachman, P., Precup, D., 2015. Data generation as sequential decision making. Adv. Neural Inf. Process. Syst., 3249–3257

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., Bengio, Y., 2016. An actor-critic algorithm for sequence prediction. arXiv preprint arXiv:1607.07086.

Banerjee, S., Lavie, A., 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pp. 65–72.

Bengio, Y., Frasconi, P., Simard, P., 1993. The problem of learning long-term dependencies in recurrent networks. In: IEEE international conference on neural networks. IEEE, pp. 1183–1188.

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., 2003. A neural probabilistic language model. J. Mach. Learn. Res. 3 (Feb), 1137–1155.

Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N., 2015. Scheduled sampling for sequence prediction with recurrent neural networks. CoRR, vol. abs/1506.03099, 2015. [Online]. Available: URL: http://arxiv.org/abs/1506.03099.

Berglund, M., Raiko, T., Honkala, M., Kärkkäinen, L., Vetek, A., Karhunen, J.T., 2015. Bidirectional recurrent neural networks as generative models. Adv. Neural Inf. Process. Syst., 856–864

Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., 2017. Enriching word vectors with subword information. Trans. Assoc. Computat. Linguistics 5, 135–146.

Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., Bengio, S., 2016. Generating sentences from a continuous space. In: Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 10–21. Available: URL: https://www.aclweb.org/anthology/K16-1002.

Bowman, S.R., Vilinis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S., 2019. Generating sentences from a continuous space," arXiv preprint arXiv:1511.06349.

Bradbury, J., Merity, S., Xiong, C., Socher, R., 2016. Quasi-recurrent neural networks. arXiv preprint arXiv:1611.01576.

Britz, D., 2015. Understanding Convolutional Neural Networks for NLP. URL: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/, [Online; accessed 16-January-2018].

Britz, D., 2015. Convolutional Neural Network for NLP. URL: https://www.kdnuggets.com/2015/11/understanding-convolutional-neural-networks-nlp.html/2 [Online; accessed 18-january-2019.

Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., Charlin, L., 2018. Language gans falling short, arXiv preprint arXiv:1811.02549.

Che, T., Li, Y., Zhang, R., Hjelm, R.D., Li, W., Song, Y., Bengio, Y., 2017. Maximum-likelihood augmented discrete generative adversarial networks. arXiv preprint arXiv:1702.07983.

Chen, X., Li, Y., Jin, P., Zhang, J., Dai, X., Chen, J., Song, G., 2019. Adversarial subsequence for text generation. arXiv preprint arXiv:1905.12835.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2015. Gated feedback recurrent neural networks. In: International Conference on Machine Learning, pp. 2067–2075.

Collobert, R., Weston, J., 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning ACM, pp. 160–167.

Dair.ai, E., 2018. Deep Learning for NLP: An Overview of Recent Trends. URL: http://medium.com/dair-ai/deep-learning-for-nlp-an-overview-ofrecent-trends-d0d8f40a776d/2018 (Online; accessed 14-january-2018).

Dansbecker, 2018. Rectified linear units in Deep learning. URL: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning [Online; accessed 22-februray-2019].

Dethlefs, N., Cuayáhuitl, H., 2010. Hierarchical reinforcement learning for adaptive text generation. In: Proceedings of the 6th International Natural Language Generation Conference. Association for Computational Linguistics, pp. 37–45.

Dieng, A.B., Kim, Y., Rush, A.M., Blei, D.M., 2018. Avoiding latent variable collapse with generative skip models. arXiv preprint arXiv:1807.04863.

Duchi, J., Hazan, E., Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. 12 (Jul), 2121–2159.

Du, J., Jia, P., Dai, Y., Tao, C., Zhao, Z., Zhi, D., 2019. Gene2vec: distributed representation of genes based on co-expression. BMC Genom. 20 (1), 82.

Elman, J.L., 1990. Finding structure in time. Cognitive Sci. 14 (2), 179–211.

Fedus, W., Goodfellow, I., Dai, A.M., 2018. Maskgan: Better text generation via filling in the_, arXiv preprint arXiv:1801.07736.

Ge, H., Xia, Y., Chen, X., Berry, R., Wu, Y., 2018. Fictitious gan: Training gans with historical models. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 119–134.

Glorot, X., Bordes, A., Bengio, Y., 2011,. Domain adaptation for large-scale sentiment classification: a deep learning approach. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp. 513–520.

Goldberg, Y., 2017. Neural network methods for natural language processing. Synthesis Lectures on Human Language Technologies 10 (1), 1–309.

Goldberg, Y., Levy, O., 2014. word2vec explained: deriving mikolov et al'.s negative-sampling word-embedding method, , vol. 2. arXiv preprint arXiv:1402.3722.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680.

Graves, A., Wayne, G., Danihelka, I., 2014. Neural turing machines. arXiv preprint arXiv:1410.5401.

Grover, A., Leskovec, J., 2016. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining ACM, pp. 855–864.

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J., 2018. Long text generation via adversarial training with leaked information. In: Thirty-Second AAAI Conference on Artificial Intelligence.

Gupta, D., 2017. Fundamentals of deep learning: Activation functions and when to use them. URL: https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/ [Online; accessed 25-february-2019].

Guu, K., Hashimoto, T.B., Oren, Y., Liang, P., 2018. Generating sentences by editing prototypes. Trans. Assoc. Comput. Linguistics 6, 437–450.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9 (8), 1735–1780.

Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J., 2013. Stochastic variational inference. J. Mach. Learn. Res. 14 (1), 1303–1347.

Hong, M., Wang, M., Luo, L., Tan, X., Zhang, D., Lao, Y., 2018. Combining gated recurrent unit and attention pooling for sentimental classification. In: Proceedings of the 2018 2Nd International Conference on Computer Science and Artificial Intelligence, ser. CSAI '18. New York, NY, USA: ACM, pp. 99–104. Available: URL: http://doi.acm.org/10.1145/3297156.3297267.

Iqbal, T., Sambyal, A.S., 2019. A review of text summarization using gated neural networks. Int. J. Comput. Sci. Eng. 06 (03), 51–55.

Jacovi, A., Shalom, O.S., Goldberg, Y., 2018. Understanding convolutional neural networks for text classification. arXiv preprint arXiv:1809.08037.

Jaiswal, A., AbdAlmageed, W., Wu, Y., Natarajan, P., 2018. Capsulegan: Generative adversarial capsule network. In: Proceedings of the European Conference on Computer Vision (ECCV).

Kim, Y., Wiseman, S., Miller, A.C., Sontag, D., Rush, A.M., 2018. Semi-amortized variational autoencoders. arXiv preprint arXiv:1802.02550.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D.P., Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kingma, D.P., Welling, M., 2019. An introduction to variational autoencoders. CoRR, vol. abs/1906.02691. [Online]. Available: URL: http://arxiv.org/abs/1906.02691.

Lai, S., Xu, L., Liu, K., Zhao, J., 2015. Recurrent convolutional neural networks for text classification. In: Twenty-ninth AAAI Conference on Artificial Intelligence.

Lamb, A.M., Goyal, A.G.A.P., Zhang, Y., Zhang, S., Courville, A.C., Bengio, Y., 2016. Professor forcing: A new algorithm for training recurrent networks. Advances In Neural Information Processing Systems, 4601–4609.

Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A.Y., 2011. On optimization methods for deep learning. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. Omnipress, pp. 265–272.

Levine, S., Finn, C., Darrell, T., Abbeel, P., 2016. End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. 17 (1), 1334–1373.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., Gao, J., 2016. Deep reinforcement learning for dialogue generation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Austin, Texas, pp. 1192–1202. Available: URL: https://www.aclweb.org/anthology/D16-1127.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., Jurafsky, D., 2017. Adversarial learning for neural dialogue generation. arXiv preprint arXiv:1701.06547.

Li, Y., Pan, Q., Wang, S., Yang, T., Cambria, E., 2018. A generative model for category text generation. Inf. Sci. 450, 301–315.

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001

Li, Z., Jiang, X., Shang, L., Li, H., 2018. Paraphrase generation with deep reinforcement learning. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3865–3878. Available: URL: https://www.aclweb.org/anthology/D18-1421.

Lin, C.-Y., 2004. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out.

Lin, K., Li, D., He, X., Zhang, Z., Sun, M.-T., 2017. Adversarial ranking for language generation. Advances in Neural Information Processing Systems, 3155–3165.

Lucas, J., Tucker, G., Grosse, R., Norouzi, M., 2019. 'Understanding posterior collapse in generative latent variable models.

Mangal, S., Joshi, P., Modak, R., 2019. Lstm vs. gru vs. bidirectional rnn for script generation. arXiv preprint arXiv:1908.04332.

Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J., 2016. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Adv. Neural Inf. Process. Syst., 3111–3119

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al., 2016. Abstractive text summarization using sequence-to-sequence RNNS and beyond. arXiv preprint arXiv:1602.06023.

NSS, 2017. An intuitive Understanding of word embeddings: from count vectors to Word2Vec. URL: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/ (Online; accessed 18-january-2018).

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al., 2016. Reward augmented maximum likelihood for neural structured prediction. Advances In Neural Information Processing Systems, 1723–1731.

Olah, C., Carter, S., 2016. Attention and augmented recurrent neural networks. Distill.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, pp. 311–318.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: International conference on machine learning, pp. 1310–1318.

Pawade, D., Sakhapara, A., Jain, M., Jain, N., Gada, K., 2018. Story scrambler-automatic text generation using word level RNN-LSTM. Int. J. Inf. Technol. Comput. Sci. (IJITCS) 10 (6), 44–53.

Peng, B., Yao, K., 2015. Recurrent neural networks with external memory for language understanding. arXiv preprint arXiv:1506.00195.

Pennington, J., Socher, R., Manning, C.D., 2014. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543.

Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Rezende, D.J., Mohamed, S., 2015. Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770.

Robbins, H., Monro, S., 1951. A stochastic approximation method. Ann. Math. Stat., 400–407

Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., et al., 1988. Learning representations by back-propagating errors. Cognitive Modeling 5 (3), 1.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 6088. Learning representations by back-propagating errors. Nature 323 (6088), 533–536.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., Valaee, S., 2017. Recent advances in recurrent neural networks. arXiv preprint arXiv:1801.01078.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved techniques for training gans. Advances in Neural Information Processing Systems, 2234–2242.

Schuster, M., Paliwal, K.K., 1997. Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45 (11), 2673–2681.

Semeniuta, S., Severyn, A., Barth, E., 2017. A hybrid convolutional variational autoencoder for text generation. arXiv preprint arXiv:1702.02390.

Semeniuta, S., Severyn, A., Gelly, S., 2018. On accurate evaluation of gans for language generation. arXiv preprint arXiv:1806.04936.

Shacklett, M., 2017. Unstructured data: A cheat sheet. URL: https://www.https://www.techrepublic.com/article/unstructured-data-the-smart-persons-guide// (Online; accessed 12-November-2019).

Sharma, S., 2017. Activation Functions in Neural Networks. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 [Online; accessed 24-february-2019].

Sherstinsky, A., 2018. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. CoRR, vol. abs/1808.03314, 2018. [Online]. Available: URL: http://arxiv.org/abs/1808.03314.

Shi, Z., Chen, X., Qiu, X., Huang, X., 2018. Towards diverse text generation with inverse reinforcement learning. arXiv preprint arXiv:1804.11258.

Shi, Z., Chen, X., Qiu, X., Huang, X., 2018. Toward diverse text generation with inverse reinforcement learning. arXiv preprint arXiv:1804.11258.

Skymind, 2018. A beginners guide to deep reinforcement learning. URL: https://skymind.ai/wiki/deep-reinforcement-learning, [Online; accessed 15-March-2019].

Smith, A., 2013. Sequential Monte Carlo methods in practice. Springer Science & Business Media.

Sutskever, I., Martens, J., Hinton, G.E., 2011. Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 1017–1024.

Tevet, G., Habib, G., Shwartz, V., Berant, J., 2018. Evaluating text gans as language models. arXiv preprint arXiv:1810.12686.

Tieleman, T., Hinton, G., 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: neural networks for machine learning 4 (2), 26–31.

Vedantam, R., Lawrence Zitnick, C., Parikh, D., 2015. Cider: Consensus-based image description evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4566–4575.

Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K., 2017. Feudal networks for hierarchical reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, pp. 3540–3549.

Wang, W., Gan, Z., Xu, H., Zhang, R., Wang, G., Shen, D., Chen, C., Carin, L., 2019. Topic-guided variational autoencoders for text generation. CoRR, vol. abs/1903.07137, 2019. [Online]. Available: URL: http://arxiv.org/abs/1903.07137.

Weston, J., Chopra, S., Bordes, A., 2014. Memory networks. arXiv preprint arXiv:1410.3916.

Xu, J., Durrett, G., 2018. Spherical latent spaces for stable variational autoencoders. arXiv preprint arXiv:1808.10805.

Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., Hao, H., 2015. Convolutional neural networks for text hashing. In: Twenty-Fourth International Joint Conference on Artificial Intelligence.

Yang, Z., Hu, Z., Salakhutdinov, R., Berg-Kirkpatrick, T., 2017. Improved variational autoencoders for text modeling using dilated convolutions. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, pp. 3881–3890.

Yasutomi, S., Tanaka, T., 2014. Parameter estimation for von mises-fisher mixture model via gaussian distribution. In: Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific. IEEE, 2014, pp. 1–5.

Yu, F., Koltun, V., 2015. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122.

Yu, L., Zhang, W., Wang, J., Yu, Y., 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI Conference on Artificial Intelligence.

Zhang, Y. Wallace, B., 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

Zhang, J., Zong, C., et al., "Deep neural networks in machine translation: An overview," 2015.

Zhang, Y., Gan, Z., Carin, L., 2016. Generating text via adversarial training. NIPS workshop on Adversarial Training vol. 21.

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., Carin, L., 2017. Adversarial feature matching for text generation. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, pp. 4006–4015.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P. H., 2015. Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE international conference on computer vision, pp. 1529–1537.

Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K., 2008. Maximum entropy inverse reinforcement learning. In: Aaai, vol. 8. Chicago, IL, USA, pp. 1433–1438.

Please cite this article as: T. Iqbal and S. Qureshi, The survey: Text generation models in deep learning, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.04.001