Text Generation: A comparison and review of approaches

Amrit Sreekumar
amritsre@buffalo.edu
University at Buffalo
New York, USA

Abstract

Text Generation has been in research for more than two decades now. From LSTMs to the latest Transformer based approaches, it has progressed to a great extent. Recurrent Neural Networks, Long-short term memory, and gated neural networks were considered the state of the art approaches in sequence modeling and transduction problems before Transformers came into the picture (Vaswani et al. 2017) [1]. Transformers also came with a lot of computational overheads that make it almost impossible to train in a normal setting. At present most new approaches are Transformer based and come out from the biggest names in the industry. This paper reviews the performance of some of the popular approaches and the economical feasibility factor that pertains to each approach.

1. Introduction

The size of language models is increasing at a very fast pace, with Transformers now reaching a trillion trainable parameters. Multiple successful approaches have constituted the development of systems that generate text based on previous inputs.

Recurrent Neural Networks were one of the first approaches to Text Generation, but in its original form, it was extremely difficult to train due to the vanishing or exploding gradients issues (Sutskever et al. 2011) [6]. But, this was overcome using Hessian-Free optimizers by applying them to character-level language modeling tasks.

Deep LSTMs was shown to improve significantly over its initial implementations, but even this approach led to trainable parameters being shot up to 380 million (Sutskever et al. 2014) [5]. Even though GANs were mostly used for image-related tasks, it found its way to NLP as well and SeqGan showed excellent performance when generating sequences that are based on real-world scenarios (Yu et al. 2017) [2].

The breakthrough was however when attention-based models were introduced, it significantly outperformed all other previous implementations and was found to be versatile enough to be used over different domains. The training of Transformers is highly parallelizable, but they require humongous computing power for days. However, Transformers remain the benchmark to date and has state-of-the-art performance scores in a multitude of tasks. This paper will analyze how these

approaches compare with each other and how the trade-off between the computation power and the final performance scores decide the feasibility of each model.

2. Approaches

This literature review focusses and compares 6 different approaches that came out in about the last decade:

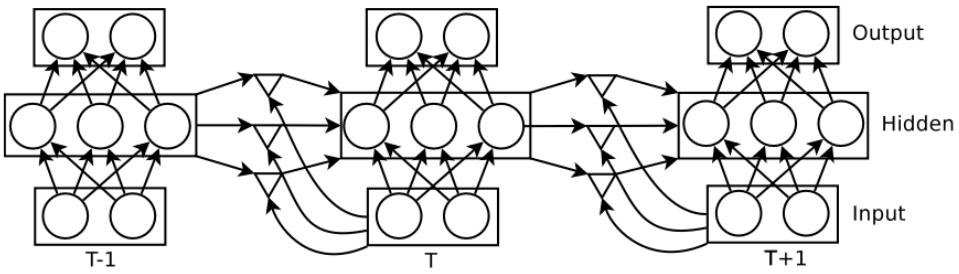
- Attention is all you need [1].
- SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient [2].
- Diversity Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation [3].
- Convolutional Sequence to Sequence Learning [4].
- Sequence to Sequence Learning with Neural Networks [5].
- Generating Texts using Recurrent Neural Networks [6].

The above papers in arbitrary order had a novel or state-of-the-art approaches at the times they were published.

2.1. Recurrent Neural Networks

The paper Generating Texts using Recurrent Neural Networks (Sutskever et al. 2011) [6] demonstrated that RNNs trained with Hessian-Free optimizers (HF) on character-level language models solved the problem of vanishing/exploding gradients and gave good results. They termed this type of RNNs as MRNNs or multiplicative RNNs as they use gated connections which allow the current input character to determine the transition matrix from one hidden state vector to the next. They are temporal architectures and the paper argues that they are better suited for language modeling tasks.

Below is a representation of an MRNN. “The Multiplicative RNN gates the recurrent weight matrix with the triangles represent a factor that applies a learned filter at each of its two input vertices and the product of the outputs is sent via weighted connections to all the units connected to the third vertex of the triangle” (Sutskever et al. 2011) [6].



The RNN is designed to predict the next character in the sequence. We can sample stochastically from the MRNN as well, even though the hidden states are deterministic. The resulting model was claimed to be of high quality and generated a somewhat contextual continuation to the sequence. The following were the generated texts when a Wikipedia trained MRNN model was given the inputs: “England, Spain, France, Germany,”

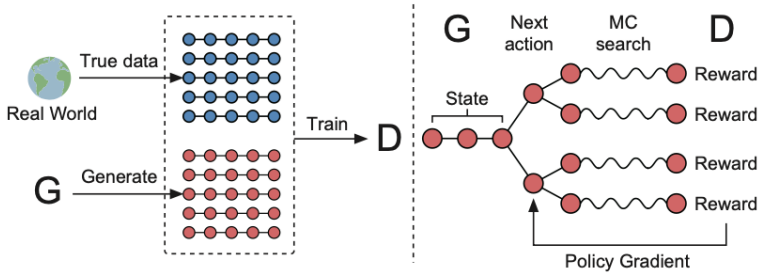
England, Spain, France, Germany, and Massachusetts.
England, Spain, France, Germany, cars, and direct schools
England, Spain, France, Germany, , or New Orleans and Uganda.
England, Spain, France, Germany, , Westchester,
Jet State, Springfield, Athleaves and Sorvinhee

The MRNN correctly interpreted that the lists that it was given as inputs were geographic locations and the generated texts that match the contexts of the inputs.

2.2. Generative Adversarial Networks

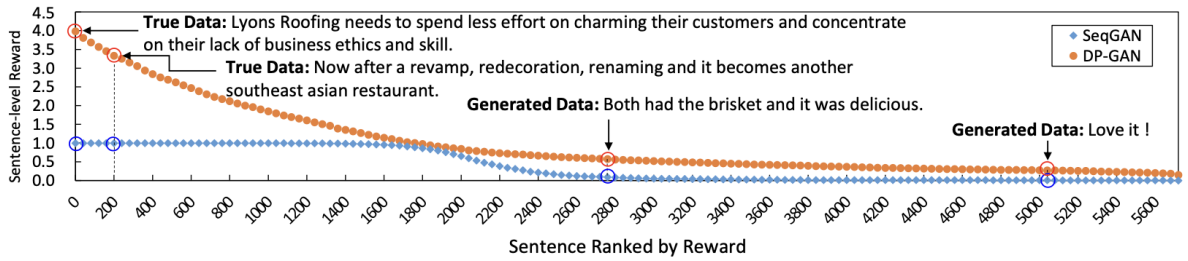
GANs contain a discriminative neural network and a generative neural network. The first is used to identify whether the data is computer generated or from the original pool of sentences. The latter is used to generate the sequences from the given distribution. GANs are very popular in computer vision-based tasks. GAN is designed to generate real-valued, continuous data, hence it is not easy to generate sequences of discrete tokens using GANs as the generator randomly samples first followed by a deterministic transformation. Moreover, GAN can give loss for an entire sequence as it is generated, but for a partially generated sequence, it is not easy to find the loss without a knowledge of the future words (Yu et al. 2017) [2].

The paper SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient (Yu et al. 2017) [2] proposes an architecture shown below:



The left half of the SeqGAN takes the generated data from the generator and true data from the samples. “The right side has G trained by policy gradient where the final reward signal is provided by D and passed back to the intermediate action value via Monte Carlo search.” (Yu et al. 2017) [2]

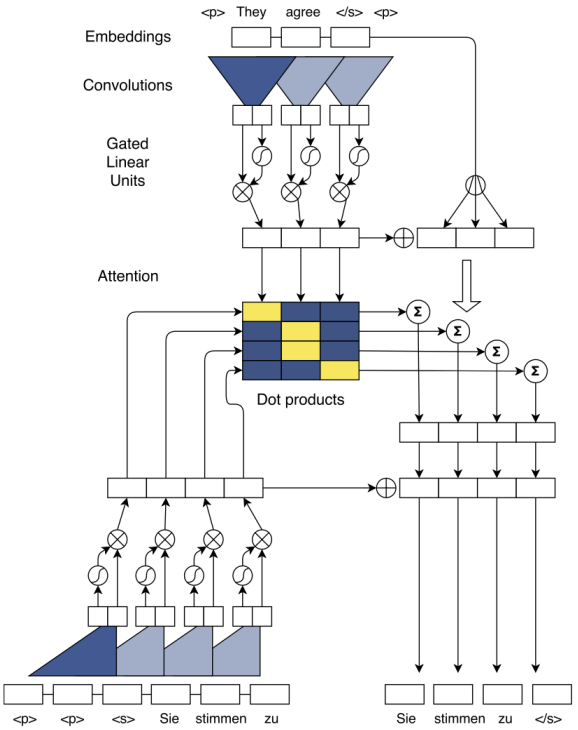
Diversity Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation (Xu et al. 2018) [3], is an improvement over SeqGAN in terms of Relevance, fluency, and Diversity of words generated by the GAN. The paper mentions that the existing methods of text generation produce repetitive and “boring” expressions, hence they claim DP-GAN produces more novel words than the others by rewarding the system based on it. In comparison to SeqGAN which uses a binary classifier as the discriminator, DP-GAN proposes a language-model-based discriminator that can better distinguish novel text from repeated text without saturation problems. It is mentioned that “DP-GAN has the strong ability to resist reward saturation and can give more precise rewards for text in terms of novelty. Below is a comparison of SeqGAN and DP-GAN based on the distribution of awards, where the upper two sentences are sampled from real-world data and the lower two from the generated data. It is evident that the reward distribution for SeqGAN saturates and does not distinguish the novelty of text” (Xu et al. 2018) [3].



2.3. Neural Networks

Deep Neural Networks are a benchmark for many tasks across domains. Even though DNNs work well with large labeled training sets, they cannot be used in sequence to sequence mapping. The paper Sequence to Sequence Learning with Neural Networks (Sutskever et al. 2014) [5] presents an approach using sequence to sequence learning that makes minimal assumptions on the sequence structure. This implementation uses two different LSTMs for the purpose. One of which is multilayered and maps the input to a vector representation with a certain dimension. The second LSTM is deeper and it decodes the target from its vector representation. The core of the experiments done in the above research involved training large Deep LSTM on many pair sentences. The research also discovered a hack to train LSTM much better by reversing the source sentences. (Sutskever et al. 2014) [5]

Facebook AI Research's Convolutional Sequence to Sequence Learning (Gehring et al. 2017) [4] uses an architecture based entirely on a Convolutional neural network. The paper introduces a convolutional block structure in which both the encoder and decoder share the block structure that computes intermediate states based on a fixed number of inputs. Several blocks that contain a 1D convolution below a non-linear layer are stacked to build the architecture.

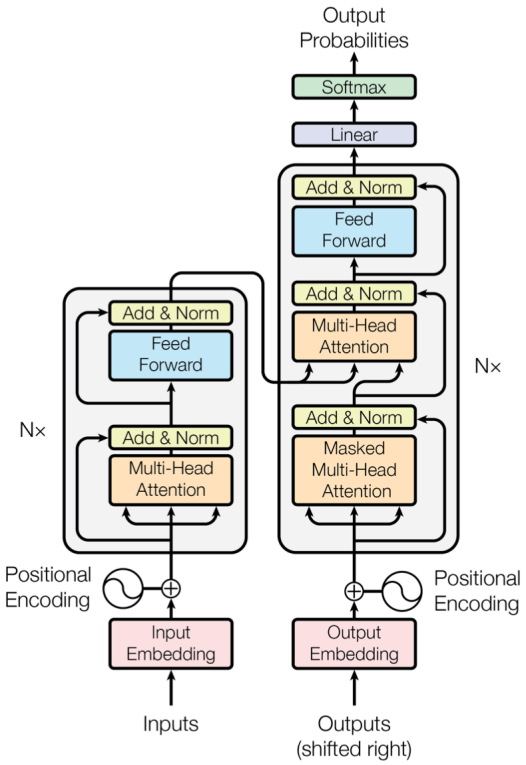


The above architecture of the system is defined for a machine translation task from English to German. The source sentence is encoded (top) and the attention values are computed for the four German target words (center) simultaneously. "The conditional inputs are computed by the attention (center right) to

the decoder states which then predict the target words (bottom right). The sigmoid and multiplicative boxes illustrate Gated Linear Units." (Sutskever et al. 2014) [5]

2.4. Transformers

Follows an overall encoder-decoder type of architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. The underlying mechanism for the state-of-the-art model that is Transformer based is self-attention. Self-attention is used successfully in various domains. The inputs to the attention layer are used as the queries. Transformers use multi-head attention layers and are based entirely on attention. The self-attention layer takes a set of vectors and produces outputs. A residual connection and layer norm are added to get the final output from the block. Following is the architecture of the first transformer that the paper Attention is all you need (Vaswani et al. 2017) [1] uses and it consists of 12 blocks with 6 attention heads.



Transformers had outperformed the best previously reported models including ensembles. To date, various transformer-based architectures are the benchmark across domains and the scores keep increasing.

3. Performance and Compute comparison

A critical factor that affects the feasibility and usability of a model is its performance scores on various datasets and the amount of computing required to train and use them. The researches reviewed in this paper span over a time period of about a decade. Even though it is not possible for a direct comparison on all aspects having the same setting, getting an idea of how the architectures fare on these fronts is possible by looking at the experiments conducted as a part of these researches, their scores, and the setting they used for training the network.

The MRNN introduced by the 2011 paper: Generating Texts using Recurrent Neural Networks (Sutskever et al. 2011) [6], performs at the state of the art for pure character-level models (including sequence memoizers). However, its compression performance is not as good as some models which have explicit knowledge of words like the PAQ model. "PAQ is a mixture model of a large number of

well-chosen context models whose mixing proportions are computed by a neural network whose weights are a function of the current context, and whose predictions are further combined with a neural-network like model." (Sutskever et al. 2011) [6]. The below table shows the test bits per character for each experiment, with the training bits in brackets (where available).

DATA SET	MEMOIZER	PAQ	MRNN	MRNN (FULL SET)
WIKI	1.66	1.51	1.60 (1.53)	1.55 (1.54)
NYT	1.49	1.38	1.48 (1.44)	1.47 (1.46)
ML	1.33	1.22	1.31 (1.27)	

The setting in which the MRNN was trained, consists of 8 high-end GPUs with 4GB of RAM each, and the gradient was computed on 160·300=48000 sequences of length 250, of which 8·300=2400 sequences were used to compute the curvature-matrix vector products that are needed for the HF optimizer. It was fairly trained with 100MB of data.

The SeqGAN (Yu et al. 2017) [2] compares four generative models: Random Token Generation, MLE trained LSTM, scheduled sampling (Bengio et al. 2015), and Policy Gradient with BLEU (PG-BLEU). SeqGAN significantly outperforms other baselines as shown in the table below which contains Negative log-likelihood (NLL) and the p-value which is between SeqGAN and the baseline from T-tests.

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	

The DP-GAN model (Xu et al. 2018) [3] conducts two forms of evaluations, an automatic one, and another human evaluation. The following two tables give the scores (automatic evaluation: left, human evaluation: right).

Yelp	Token	Dist-1	Dist-2	Dist-3	Dist-S
MLE	151.2K	1.2K	3.9K	6.6K	3.9K
PG-BLEU	131.1K	1.1K	3.3K	5.5K	3.1K
SeqGAN	140.5K	1.1K	3.5K	6.1K	3.6K
DP-GAN(S)	**438.6K**	1.7K	7.5K	15.7K	10.6K
DP-GAN(W)	271.9K	2.8K	14.8K	29.0K	12.6K
DP-GAN(SW)	406.8K	**3.4K**	**22.3K**	**49.6K**	**17.3K**
Amazon	Token	Dist-1	Dist-2	Dist-3	Dist-S
MLE	176.1K	0.6K	2.1K	3.5K	2.6K
PG-BLEU	124.5K	0.6K	1.9K	3.5K	2.3K
SeqGAN	217.3K	0.7K	2.6K	4.6K	3.2K
DP-GAN(S)	**467.6K**	0.8K	3.6K	7.6K	7.0K
DP-GAN(W)	279.4K	1.6K	8.9K	18.4K	9.6K
DP-GAN(SW)	383.6K	**1.9K**	**11.7K**	**26.3K**	**13.6K**
Dialogue	Token	Dist-1	Dist-2	Dist-3	Dist-S
MLE	81.1K	1.4K	4.4K	6.3K	4.1K
PG-BLEU	97.9K	1.2K	3.9K	5.5K	3.3K
SeqGAN	83.4K	1.4K	4.5K	6.5K	4.5K
DP-GAN(S)	**112.2K**	1.5K	5.2K	8.5K	5.6K
DP-GAN(W)	79.4K	1.9K	7.7K	11.4K	6.0K
DP-GAN(SW)	97.3K	**2.1K**	**10.8K**	**19.1K**	**8.0K**

Yelp	Relevance	Diversity	Fluency	All
MLE	1.49	1.73	1.78	1.89
PG-BLEU	1.47	2.59	**1.38**	2.22
SeqGAN	1.48	2.40	1.54	2.12
DP-GAN	**1.32**	**1.23**	1.66	**1.51**
Amazon	Relevance	Diversity	Fluency	All
MLE	1.52	1.81	1.72	1.93
PG-BLEU	1.62	2.48	1.63	2.24
SeqGAN	1.56	2.37	**1.40**	1.97
DP-GAN	**1.31**	**1.25**	1.52	**1.50**
Dialogue	Relevance	Diversity	Fluency	All
MLE	1.19	1.84	1.37	1.87
PG-BLEU	**1.13**	1.85	1.21	1.75
SeqGAN	**1.13**	1.71	**1.20**	1.64
DP-GAN	**1.13**	**1.50**	1.30	**1.55**

The top-left table indicates: "performance of DP-GAN and three baselines on review generation and dialogue generation tasks. Higher is better. DP-GAN(S), DP-GAN(W), and DP-GAN(SW) represent DP-GAN with only sentence-level reward, only word-level reward, and combined reward, respectively. Token represents the number of generated words. Dist-1, Dist-2, Dist-3, and Dist-S are

respectively the number of distinct unigrams, bigrams, trigrams, and sentences in the generated text. (Xu et al. 2018) [3]. The top right table has the score that represents the averaged rankings of each model and the lower is better.

The 2014 paper: Sequence to Sequence Learning using Neural Networks (Sutskever et al. 2014) [5] used deep LSTMs with 4 layers, with 1000 cells at each layer and 1000 dimensional word embeddings, with an input vocabulary of 160,000 and an output vocabulary of 80,000. The final model has 380M parameters with 64M recurrent connections. The highest scores were obtained when the initializations and order of the minibatches were randomised. The paper mentions that while the decoded translations of the LSTM ensemble do not beat the state of the art, it is the first time that a pure neural translation system outperforms a phrase-based SMT baseline on a large MT task by a big margin even though it could not handle out-of-vocabulary words. The following table shows the methods that use Neural Network along with an SMT system on the WMT'14 English-French dataset.

Method	**test BLEU score (ntst14)**
Baseline System [29]	33.30
Cho et al. [5]	34.54
State of the art [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	∼45

The paper Convolutional Sequence to Sequence Learning (Gehring et al. 2017) [4], on three translational tasks with Recurrent Neural Network-based approaches. The following table summarises the results based on BLEU scores and the ConvS2S outperforms the others on all three tasks.

WMT'16 English-Romanian	**BLEU**
Sennrich et al. (2016b) GRU (BPE 90K)	28.1
ConvS2S (Word 80K)	29.45
ConvS2S (BPE 40K)	30.02

WMT'14 English-German	**BLEU**
Luong et al. (2015) LSTM (Word 50K)	20.9
Kalchbrenner et al. (2016) ByteNet (Char)	23.75
Wu et al. (2016) GNMT (Word 80K)	23.12
Wu et al. (2016) GNMT (Word pieces)	24.61
ConvS2S (BPE 40K)	25.16

WMT'14 English-French	**BLEU**
Wu et al. (2016) GNMT (Word 80K)	37.90
Wu et al. (2016) GNMT (Word pieces)	38.95
Wu et al. (2016) GNMT (Word pieces) + RL	39.92
ConvS2S (BPE 40K)	40.51

The largest of the three datasets WMT'14 English-French took 8 GPUs 37 days to train with a batch size of 32 on each worker. These GPUs were however not the most powerful or modern at the time of the release of this paper as claimed by its authors.

The attention/transformer-based models come with a lot of computational overhead. These overheads are significantly bigger than the previous approaches measured in this paper. Nevertheless, the

Transformer based models achieve state-of-the-art performance scores on the WMT'14 dataset, beating all other previous attempts at it. One thing to note is, even though the computational overhead is huge, it uses it more efficiently and performs better than the other approaches even with a fraction of their training cost. "On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in the following table) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4" (Vaswani et al. 2017) [1]. The comparisons are shown in the table following.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

The training for the big model took 3.5 days on 8 state-of-the-art P100 GPUs. However, this is just about the base model, and today there are many more much bigger models trained on 100s of GPUs for days. They give the best results and cannot be beaten by any other approach in an array of domains, but it comes at a cost that is not anywhere affordable by any common person. The following table gives an idea of a few of the scaled-up transformers used in the present day.

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	12	768	?	117M	40 GB	
GPT-2	24	1024	?	345M	40 GB	
GPT-2	36	1280	?	762M	40 GB	
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	40	1536	16	1.2B	174 GB	64x V100 GPU
Megatron-LM	54	1920	20	2.5B	174 GB	128x V100 GPU
Megatron-LM	64	2304	24	4.2B	174 GB	256x V100 GPU (10 days)
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)

We can see that it should cost millions of dollars to train one such model to achieve the performance figures we have today. The best things come at a cost, but the big question is to ask whether the improvement figures are big enough to pump in millions of dollars into training these models.

4. Conclusion

Language modeling has been around for quite some time now, the models keep getting bigger and better by the day. Text Generation has progressed so much that it has been put to commercial use in different scenarios. The latest models are so good that as humans we may find it difficult to distinguish between what is machine-generated and what is written by a human. All these advantages make our

lives a lot easier and they are the results of years of research and funding. But what we don't think many times is that these come with a lot of tradeoffs. The environmental factor is one big thing we have to keep in mind as we progress. Most of the big models with billions of parameters come from the big names in the industry who can afford to run 100s of GPUs for days, any individual entity stands no match for this type of setting, and here comes the issue of accessibility.

Starting with just a fraction of a GB of data to train RNNs, we have progressed to use 100s of GBs of data to train transformers. These have led to some groundbreaking performance scores but they are from perfect and biases in the data. The ethicality and the carbon footprint these types of settings leave are one huge factor that should be placed into consideration when we evaluate what we gain out of these models. However, open-source forums like HuggingFace [8] make Transformers easily accessible to everyone.

Acknowledgment

Writing this literature review was a big learning and an eye-opener at the same time. It showed me how our technology has drastically changed in the last decade and how it is changing by the day. I thank Dr. Cassandra Jacobs for this opportunity and Liz Soper for the support throughout the semester.

References

[1] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

[2] Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu. "Seqgan: Sequence generative adversarial nets with policy gradient." In *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1. 2017.

[3] Xu, Jingjing, Xuancheng Ren, Junyang Lin, and Xu Sun. "DP-GAN: diversity-promoting generative adversarial network for generating informative and diversified text." *arXiv preprint arXiv:1802.01345* (2018).

[4] Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. "Convolutional sequence to sequence learning." In *International Conference on Machine Learning*, pp. 1243-1252. PMLR, 2017.

[5] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." In *Advances in neural information processing systems*, pp. 3104-3112. 2014.

[6] Sutskever, Ilya, James Martens, and Geoffrey E. Hinton. "Generating text with recurrent neural networks." In *ICML*. 2011.

[7] Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. "Scheduled sampling for sequence prediction with recurrent neural networks." *arXiv preprint arXiv:1506.03099* (2015).

[8] https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture13.pdf

[9] https://huggingface.co/docs/transformers/index