Evaluating the Performance of Language GANs with Small Seed Corpora

Ishadi Jayasinghe and Surangika Ranathunga,

Department of Computer Science and Engineering
University of Moratuwa, Katubedda 10400
Sri Lanka
{[1]ishadij, [2]surangika}@cse.mrt.ac.lk

Abstract. Recently many deep learning models have been proposed for language generation. Language Generative Adversarial Nets (language GANs) are one of them. Language GANs have been presented as giving good results when trained with large corpora. However, the availability of such large corpora does not hold all the time, particularly if the requirement is to generate a larger data set using a relatively smaller seed corpus. Although extensive evaluations have been carried out to compare existing language GANs, all these experiments have used large seed corpora. Therefore they do not give an indication of the usability of language GANs w.r.t. small corpora. In this paper, we present a series of experiments that we carried out to determine the viability of language GANs for language generation with small corpora. Based on our experiments, we were able to identify some models that provide acceptable results that look promising with small data sets.

Keywords: Text generation • Limited size corpora • Training dataset size • Temperature curve • Language GANs

1 Introduction

Recently, neural network models have been used to successfully generate text in multiple contexts such as image captioning [1], machine translation [2], and poem generation [3]. These models can be basically classified into two main categories: models based on MLE and Generative Adversarial Nets (GANs) for text generation. With respect to language GANs, many different models have been proposed lately. Texygen [4] is a benchmark framework for evaluating text generation models. This framework has extensively compared text generation models against large corpora such as MS COCO captions [5], and EMNLP2017 W MT News [6]. However, the requirement for text generation is not limited to these domains, and the assumption of the availability of a large seed corpus does not hold all the time. In particular, to generate large datasets to be used in other Machine Learning (ML) tasks, automatic text generation is a favorable solution when manual effort is expensive and time consuming. For example, GEOS [7] is a system for solving geometry questions; however the amount of data they are having is under 500 sentences. With the questions being complex and diverse, this low quantity inhibits any application of deep learning models for parsing these questions. Therefore, having a large seed corpus to generate text from is not always a valid assumption, since the lack of the data itself is the problem sometimes.

Although Texygen has shown how the performance of these language GANs and models based on Maximum Likelihood Estimate (MLE) vary with respect to large datasets, no research has looked into the same when the dataset is small. However, it is useful to identify models that give optimal results for small data sets, so that text generation for new domains with small seed corpora is viable. Moreover, according to Caccia et al. [8], the evaluation metrics used by Texygen makes it impossible to compare two given models. Therefore, we cannot anyways use the facts from Texygen to correctly benchmark the performance of Natural Language Generation (NLG) models. Caccia et al. [8] proposed Temperature sweep for comparing models more accurately and we have adopted this approach in our work.

In this research, we analyze the applicability of the state-of-the-art NLG models to generate large datasets from small corpora. We evaluate the vanilla MLE language model [4] and GANs (SeqGAN [9], TextGAN [10], LeakGAN [6], and GSGAN [11]) using small datasets extracted from the datasets used in Texygen. We also analyze the applicability of these models in terms of inherent properties of the reference dataset. Finally, we enhance Texygen [4] to (i) measure the performance of text generation models accurately with temperature sweep, and (ii) analyze the performance of language models over a set of different dataset sizes in a given range.

From our experiments, we could see GSGAN and TextGAN are not suitable to be applied on low-resourced domains. Interestingly, we could observe that MLE based models performing better than much complex GANs. In addition, the ranking of models based on performance was observed to be invariant over the different datasets. I.e. if the model A performed better than model B on the first dataset, A performed better than B on the second dataset too. However, the performance scores of all models decrease when the dataset became complex

2 Related Work

Text generation: There is a recent emergence in research in text generation [1, 2, 3]. With respect to text generation, we can basically classify the tasks into two categories; text generation in the supervised setting and in the unsupervised setting [12]. In the supervised setting, the goal is to to generate a text similar to a target set. From the above examples, image captioning, and machine translation fall under this approach. In the second setting, where the task is unsupervised, the aim is to generate samples which are following a probability distribution similar (or close) to the probability distribution of a given reference set. In simpler terms, the models are expected to generate samples that look like reference samples.

Text generation tasks such as the ones mentioned above have successfully been implemented using neural network models [1, 2, 3]. Basically two main variants of these models can be identified; models based on Maximum Likelihood Estimate (MLE) and language GANs. In principal, MLE models generate texts using the context of prior generated words. During the training phase, next word of the sequence is predicted based on the ground truth words, and in the inference phase, the prediction is done using already predicted words [13, 6]. Generative Adversarial Nets (GAN) [14] have been able to make significant advances in generation of synthetic data similar to real data. Two neural networks are used here; the generator is responsible for generating realistic looking data, while the discriminator's responsibility is to differentiate synthetic and real data accurately. Gradient of the training error from the discriminator is used to train the generator.

There are multiple identified issues with the MLE approach. Firstly, there is no appropriate metric to evaluate the output of these models [6]. As discussed above, MLE models use ground truth words as the context in the training phase while predicted words are used as the context during the inference phase. This discrepancy makes MLE models suffer from exposure bias [13,6]. Scheduled Sampling introduced by Bengio et al. [15] to address the latter problem was proved fundamentally inconsistent by Huszar [13].

Original setting of GANs works well when the GAN is operating on continuous data (image pixel values). However, when it comes to text generation, GANs have to deal with discrete tokens (sequence words) that are non-differentiable. Therefore the usage of GANs is rather challenging due to the difficulty of backpropagation through these random discrete variables [16].

Recent attempts to face this challenge could be classified to two categories. Reinforcement Learning (RL) based approaches model the task as a sequential decision making problem. GANs belonging to this category use policy gradient techniques for optimization. SeqGAN by Yu et al. [9] is an example for this approach, but one of the main drawbacks of this GAN is that the generator only gets the reward at the end of generating the whole sequence, thus making it difficult for the generator to sufficiently learn the distribution [6]. MaliGAN [16] uses a modified optimization algorithm to reduce high variance caused with the original form. RankGAN[17] replaces the original discriminator, which is a binary classifier with a ranking to reduce the gradient vanishing problem experienced with the binary form. This replacement is also a solution for the information given by the binary classifier being inadequate. Model collapsing refers to the problem of a model generating samples only from a limited area of the latent space i.e. the samples being less in diversity with each other. The discriminator being a binary classifier also contributes to this problem [6]. LeakGAN [6] was proposed to mitigate the instability issues faced during the training phase under the standard RL approach. Here, the generator is given the access to the feature representation learned by the discriminator. This specifically improves the generation of long text as this brings more information to the generator network compared to the single binary signal in most of the previous GANs. Despite the performance enhancements promised by above GANs that follow RL approach, still the high variance gradient they result makes the optimization challenging [12, 18, 10].

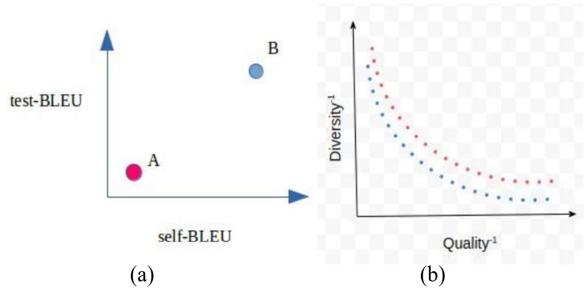
RL-free approach adheres to the original approach of GANs without incorporating ideas in RL. This approach does not yield gradients with high variance, so the GANs here are more stable and easier to train compared to the first category. TextGAN [10], GSGAN [11], and FMGAN [12] have adopted this approach and have generally reported better results compared to GANs in the previous approach.

1.3 Evaluation of Text Generation Models

When a generative model is trained, it attempts to learn a probability distribution that is similar to the original probability distribution of the training dataset. Therefore, the perfect measure would be to measure the distance between the two probability distributions, which is called the estimation error. However this is not practically feasible as we cannot use the total latent space to generate samples, and the reference set itself might not be fully representing its distribution. Therefore, some other metrics have been adopted which are more feasible in a practical setting. Test-BLEU is one such score that measures the similarity between two sets of text. It scores similar n-grams and their frequency. Therefore, this reflects sample quality. Most GANs (eg. RankGAN [17], MaliGAN [16], TextGAN [10], and LeakGAN [6]) focus only on sample quality in their performance comparisons [8]. This method is severely flawed that if a GAN generates a single quality sentence repeatedly, it will be able to get a perfect score. Moreover, model collapsing is a known issue in GANs, so they would anyways be biased towards generating sentences with a less diversity.

Self-BLEU, a score to measure the diversity of a generated set of samples was proposed by Zhu et al. [19]. Here, the samples are analyzed against itself, so a low score (low similarity) means the samples are much diverse. Even though this makes the problem above solved, this makes the comparison of GANs difficult. For an instance, consider figure 1a where the two markers represent the scores of two GANs named A and B. Here, GAN A has a better diversity (low self-BLEU), but a low quality (low test-BLEU). GAN B has the opposite; a better quality, but a poor diversity. Hence this graph is not sufficient to decide on the better performing GAN. Texygen [4] is a benchmark framework for evaluating text generation models. This framework scores quality (using metrics such as test-BLEU, EmbSim) and diversity (using metrics such as self-BLEU) separately, so suffers from the same issue.

Boltzmann temperature [20] refers to the parameter that is used to change a model's entropy. I.e., high temperature values make a model's entropy high which results in generating data with a high diversity. Similarly, low temperature values make the model stick to generating quality samples, which would not be much diverse from the reference set. Caccia et al. [8] proposed a novel approach for evaluating text generation models based on this scenario. This involves moving a model across a set of temperature values so that its performance on both diversity and quality aspects could be assessed. With this approach, if a model's temperature curve is situated below the temperature curve of another model (refer figure 1b), we can say the first model is better than the second. This allows us to use the area under the curve as a performance measure [8].



(a) (b)

Fig. 13. (a) Two example NLG models named A and B plotted on a graph with the x axis as self-BLEU and the y-axis as test-BLEU. (b) Two example temperature curves of two NLG models. The axes have the inverse metrics, so the lower is better in both axes. When it comes to the model with the blue curve, in any given quality, it has a better diversity than the model with the red curve. Therefore, this graph is sufficient enough to claim that the model with blue curve is better complex.

3 Experiment

3.3 Evaluation of Text Generation Models

Our aim is to analyze the applicability of text generation models for the task of generating more data for low-resourced domains (i.e. training dataset size below 1000).

All our experiments are based on the models and data used in Texygen [4]. We experimented with two datasets: MS COCO captions [5], and EMNLP2017 WMT News [6]. Table 1 gives a summary of statistics of these two datasets. We experiment for the dataset size from 200 to 1000 in 200 step sizes as our focus is on low-resource domains. For each data size, we increased the pre-epoch count by 10 starting from 10 and adversarial epoch count by 10 starting from 20. These values were selected based on the values used by Texygen and dataset size. To deal with the large variance that is natural when extracting a small dataset from a big dataset, we experimented with three different partitions for a given large dataset size and calculate the average. First we separate three random datasets of size 1000 from the training dataset as base training sets. These three are used to get data needed for experiments. For an instance, if we are evaluating when the dataset size is 400, we take three sets of data sized 400 from the three base training sets. Same process is followed with the test sets, but with altered numbers. We take 5 sets of size 1000 as testing sets. In order to analyze the trend of performance with increasing dataset sizes, we kept test dataset size(200) constant while varying training set sizes.

After training a model with a training set, we generate testsets of size 200 for a set of temperatures ranging from 0.001 to 2.0 with 0.5 step sizes. Based on work by Caccia et al. [8] the temperature value could be varied starting from zero to positive integers, and normally, the syntax breaks down losing the coherence of a sentence when the value is increased above 1.0. Therefore, we took 0.001 as the value equivalent to zero (to reduce overflowing/under-flowing issues due to dealing with zero) and 2.0 as the temperature on the extreme end. Compared to each test set, we calculate the test-BLEU and self-BLEU scores with the n-gram count 3. These scores are averaged over the test sets. Then this average scores for each temperature is again averaged over the scores from the other two training sets. This is done for sizes ranging from 200. At the end of this process, we get a test-BLEU score and a self-BLEU score per temperature per size. Then, for each size, we can draw a graph taking (1−testBLEU) as the x-axis and the selfBLEU as the y-axis. As the performance is directly proportional to the test-BLEU and inversely proportional to the self-BLEU, the area under this temperature curve is an accurate indicator [8]. Therefore, for each size, we need to the area under its temperature curve. We divide this task into three steps; (i) compute the area under the curve w.r.t. x-axis (this area depends on the selfBLEU score and independent on testBLEU), (ii) compute the area under the curve w.r.t. y-axis (this area depends on the 1−testBLEU score and independent on selfBLEU), and (iii) get the total area as the sum of the areas calculated in the two steps above. The area calculated under step (i) relates to the quality of the generated sample. The lesser this value, the more quality the sample. Similarly, the area calculated under step (ii) relates to the diversity of the generated sample. The lesser this value, the more diverse the sample. The sum of these two values could be taken as the performance indicator for a given training set size for a given GAN. Algorithm 1 explains this. We take these area values to plot the final result graph.

Table 1. EMNLP News Dataset

	Train	Test	Vocabulary size	Average length
MS COCO captions	120,000	10,000	27,842	11
EMNLP2017 WMT News	278,686	10,000	5728	28

3.2 Experiments with MS COCO Captions

COCO captions [5] dataset contains set of image captions that are smaller in length compared to EMNLP News dataset (refer Table 1).

Figure 4a indicates the area of the temperature curve w.r.t (1-testBLEU) axis. Here, we can see GSGAN and TextGAN to have the largest values for all data sizes compared to other models. This implies that they are the worst performing GANs with respect to quality. We can see other models in an improving trend towards decreasing in area values (increasing quality). Figure 4b indicates the area of the temperature curve w.r.t selfBLEU axis. This graph relates to the diversity of generated data and lower the area, the better the model at producing diverse data. We can see GSGAN going almost to zero when the dataset size is 1000. At this size, the generated samples from

GSGAN have just spaces except for a couple of words for the whole sample i.e. the generated set here has no more than 20 words. Analyzing the reason behind this scenario is out of the scope for this paper. However, this low count of words results in very low selfBLEU scores that the area has gone to zero or near zero. It can be seen that LeakGAN and seqGAN are on an increasing trend while MLE is seen to be diminishing on the increasing trend. TextGAN seems to follow the trend of MLE. Figure 2 is the sum of above discussed two areas; so the lower is better. Due to the scenario discussed above, the negative slope of GSGAN is misleading here. Interestingly, we can see how MLE is in the bottom compared to the rest of the models. This implies it performs the best in quality and diversity combined. LeakGAN and SeqGAN could be seen performing alike as the size grows. Even though TextGAN seems performing, we saw it has a low quality from Figure 4a. Figure 2 is also misleading due the fact it shows as if TextGAN gets close to LeakGAN over the end, but TextGAN had the worst quality along with GSGAN as per the figure 4a.

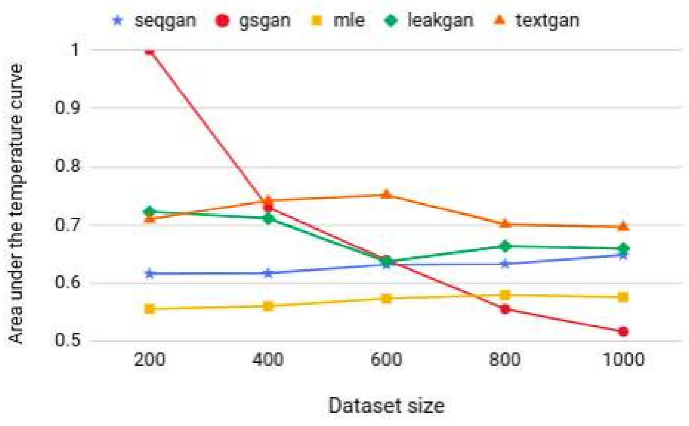


Fig. 2: Experiment results on MS COCO dataset

3.2 Experiments with EMNLP2017WMT News Dataset

Sentences of this dataset is much complex and have a longer average length compared to the previous COCO dataset. Similar to the graphs for COCO Dataset, figure 4c indicates the area of the temperature curve w.r.t (1-testBLEU) axis. This is quite similar to figure 4a in that GSGAN and TextGAN shows the same worse behavior. Here, LeakGAN produces samples with the best quality in all the sizes while MLE and SeqGAN perform quite similarly. Figure 4d indicates the area of the temperature curve w.r.t selfBLEU axis. As explained above, the values in this graph relates to the diversity; a lower value means a higher diversity. Interestingly, MLE seems to be winning here as well in that it is located at the bottom. GSGAN behaves quite different to the way it behaved during the experiments with COCO dataset. With COCO dataset, GSGAN generated a few words (around 20 words for the whole sample) with a lot of spaces. Here, it generates a small set of words repetitively. This makes its diversity score low. This happens when the dataset size is low. As the dataset size grows, GSGAN turns back to its normal behaviour with COCO dataset generating only a few words with a lot of spaces. This could be seen from 4d. LeakGAN seems to be producing more diverse data when the datasize increases. SeqGAN seems to be performing quite acceptable compared to TextGAN and LeakGAN. Figure 3 sums up these two areas. It can be observed that GSGAN making way towards lower values when the size is 1000. Similar to figure 2, representation for TextGAN is a bit misleading given its very low quality. Apart from those facts, we can see MLE as the best performing model along with SeqGAN and LeakGAN.

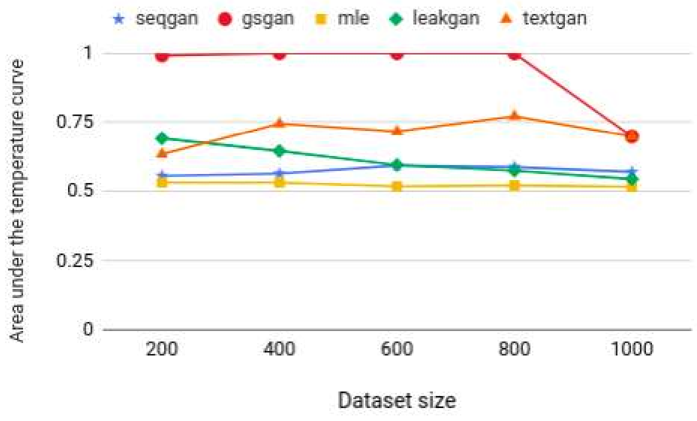


Fig. 3: Experiment results on EMNLP NEWS dataset

4 Conclusions and Future works

In this work, we studied the applicability of state of the art text generation models for the task of increasing dataset size in low resource domains. When it comes to dataset sizes below 1000, we observed GSGAN, and TextGAN to perform worse than other models in both datasets. Hence, we can come to a solid conclusion that GSGAN and TextGAN are not suitable to be applied on low resource domains no matter what qualities the given dataset is having. On the other hand, SeqGAN and LeakGAN showed a fairly similar behavior in both datasets. It was interesting that MLE turned to be performing the best in both domains in both quality and diversity aspects despite more complex GANs it was compared with. This proves the claim made by Caccia et al. [8] that MLE models are underestimated remains true when it comes to small datasets as well.

Furthermore, we extended Texygen [4] to (i) be capable of evaluating text generation models more accurately using Temperature Sweep [8], and (ii) to be capable of evaluating the trend of the performance of text generation models with dataset size increasing. Our work can be accessed from this repository1

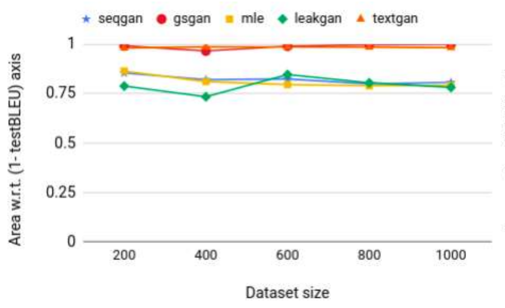
There exists much newer language GANs that claim to be performing better than some GANs we have evaluated (eg. FM-GAN [12]), which are not evaluated in Texygen. We need to evaluate those as well, but to make it a fair comparison, we should carry out the evaluations in a similar context. TexyGen provides the infrastructure for this. Therefore, by integrating recent GANs to Texygen, it would be much easier and accurate to make evaluations and comparisons. Moreover, we only experimented for sizes under 1000 in 200 steps. There is space to fine tune this considering more training dataset sizes. We kept the size of the testing size (200) throughout the experiments. It would be interesting if we find a way to determine these values based on the qualities of the reference dataset. Intuitively, if a dataset is more complex (higher vocabulary, longer average length, low self-BLEU, etc.) than another dataset, we need more data to start generating data for the first dataset than the second dataset. This was apparent from our experiments too in that EMNLP NEWS dataset needed more data than 1000 to become equal to the unscaled area scores of the models which were trained with the COCO dataset. We believe it would be very much useful if we could come up with a mechanism to decide on a minimum size of data needed for a given reference set in order to generate more data. One approach to this would be fixating on an unscaled area score and increasing the dataset size till models reach that score. Experimenting this with a set of datasets with distinguishable features such as vocabulary size, average sequence length, self-BLEU would make it possible to get some idea on the effect of each feature on the target minimum value. Our research provides infrastructure for this.

Algorithm 1: Area generate area coordinates for different training dataset sizes for a given GAN

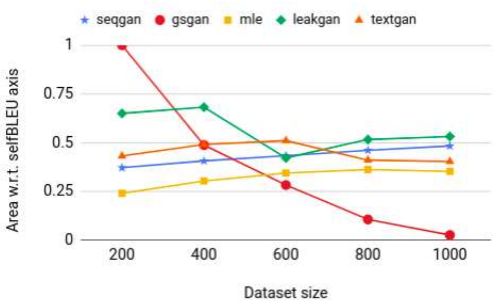
Input: Two finite sentence sets Tr and Ts for training and test set respectively, initSize, stepSize, and the finalSize of the dataset sizes evaluated

Output: (size, area) coordinates for the considered GAN

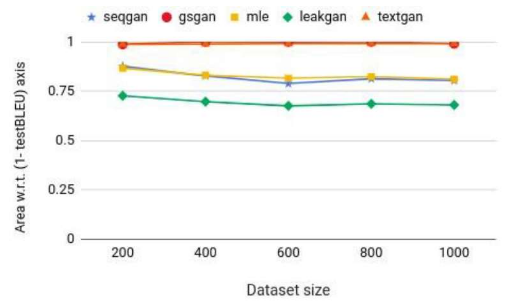
```
1  $Tr_1, Tr_2, Tr_3 \leftarrow$ 3 distinct sets of 1000 random rows from Tr without replacement
2  $Te_1, Te_2, Te_3, Te_4, Te_5 \leftarrow$ 5 distinct sets of 200 random rows from Te without replacement
3  $Tr \leftarrow \{ Tr_1, Tr_2, Tr_3\}$
4  $Te \leftarrow \{ Te_1, Te_2, Te_3, Te_4, Te_5\}$
5  Temps $\leftarrow \{0.0001, 0.5, 0.75, 1.0, 1.5, 1.75, 2.0\}$
6  Areas $\leftarrow \emptyset$
7  for size $\leftarrow$ initSize to finalSize do
8      EvalTr $\leftarrow \emptyset$
9      EvalTe $\leftarrow \emptyset$
10     AvgTempTestBLEUs $\leftarrow \emptyset$
11     AvgTempSelfBLEUs $\leftarrow \emptyset$
12     TempTestBLEUs $\leftarrow \emptyset$
13     TempSelfBLEUs $\leftarrow \emptyset$
14     for tr in Tr do
15         tr1 $\leftarrow$ tr[: size]
16         EvalTr $\leftarrow$ EvalTr $\cup$ {tr1}
17     for itr $\leftarrow$ 0 to length(EvalTr) do
18         Train the GAN with EvalTr[itr]
19         TestBLEUs $\leftarrow \emptyset$
20         SelfBLEUs $\leftarrow \emptyset$
21         for temp in Temps do
22             te1 $\leftarrow$ generate sample of initSize under temp temperature
               /* compare this with the 5 testing sets, average the
                   scores over the 5 testing sets */
23             TestBLEUs $\leftarrow$ TestBLEUs$\cup$ {Average of the test-BLEU scores}
24             SelfBLEUs $\leftarrow$ SelfBLEUs$\cup$ {Average of the self-BLEU scores}
           /* list of average scores for a single training set */
25         TempTestBLEUs $\leftarrow$ TempTestBLEUs $\cup$ {TestBLEUs}
26         TempSelfBLEUs $\leftarrow$ TempSelfBLEUs $\cup$ {SelfBLEUs}

    /* average test-BLEU and self-BLEU scores over 3 training sets */
27     AvgTempSelfBLEUs $\leftarrow$ {Average(TempSelfBLEUs)}
    /* 1 testBLEU */
28     AvgTempTestBLEUs $\leftarrow 1 -$ {Average(TempTestBLEUs)}
29     tbArea $\leftarrow$ area under the curve w.r.t. axis with AvgTempTestBLEUs
30     sbArea $\leftarrow$ area under the curve w.r.t. axis with AvgTempSelfBLEUs
31     Areas $\leftarrow$ Area $\cup$ (tbArea + sbArea)
32     size $\leftarrow$ size + stepSize
33 Areas $\leftarrow$ scaleAgainstMax(Areas)
```



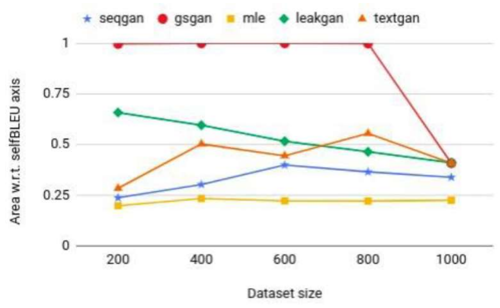
(a)



(b)



(c)



(d)

Fig. 4: Area under the temperature curve of COCO dataset plotted with respect to (a) (1− testBLEU axis) (b) selfBLEU axis. Area under the temperature curve of EMNLP NEWS dataset plotted with respect to (c) (1− testBLEU) axis (d) selfBLEU axis

References

1. Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3156–3164, 2015.
2. Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving neural machine translation with conditional sequence generative adversarial nets. arXiv preprint arXiv:1703.04887, 2017.
3. Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 670–680, 2014.
4. Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. arXiv preprint arXiv:1802.01886, 2018.
5. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014.
6. Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
7. Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Pro cessing, pages 1466–1476, 2015.
8. Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. arXiv preprint arXiv:1811.02549, 2018.
9. Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In AAAI, pages 2852–2858, 2017.
10. Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 4006–4015. JMLR. org, 2017.
11. Matt J Kusner and Jos´e Miguel Hernandez-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. arXiv preprint arXiv:1611.04051, 2016.
12. Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In Advances in Neural Information Processing Systems, pages 4671–4682, 2018.
13. Ferenc Huszar. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101, 2015.

14. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.
15. Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sam pling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems, pages 1171–1179, 2015.
16. Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversar ial networks. arXiv preprint arXiv:1702.07983, 2017.
17. Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Ad- versarial ranking for language generation. In Advances in Neural Information Processing Systems, pages 3155–3165, 2017.
18. Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distri bution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712, 2016.
19. Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. Neural text generation: past, present and beyond. arXiv preprint arXiv:1803.07133, 2018.
20. David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. Connectionist models and their implications: Readings from cognitive science. chapter a learning algorithm for boltzmann machines, 1988.