Controlled Text Generation with Adversarial Learning

Federico Betti
Politecnico di Milano
federico.betti@mail.polimi.it

Giorgia Ramponi
Politecnico di Milano
giorgia.ramponi@polimi.it

Massimo Piccardi
University of Technology Sydney
massimo.piccardi@uts.edu.au

Abstract

In recent years, generative adversarial networks (GANs) have started to attain promising results also in natural language generation. However, the existing models have paid limited attention to the semantic coherence of the generated sentences. For this reason, in this paper we propose a novel network – the Controlled TExt generation Relational Memory GAN (CTERM-GAN) – that uses an external input to influence the coherence of sentence generation. The network is composed of three main components: a generator based on a Relational Memory conditioned on the external input; a syntactic discriminator which learns to discriminate between real and generated sentences; and a semantic discriminator which assesses the coherence with the external conditioning. Our experiments on six probing datasets have showed that the model has been able to achieve interesting results, retaining or improving the syntactic quality of the generated sentences while significantly improving their semantic coherence with the given input.

1 Introduction

Natural language generation (NLG) is gaining increasing attention in the NLP community thanks to its intriguing complexity and central role in many tasks and applications. Recently, generative adversarial networks (GANs) [7] have started to display promising performance also in NLG. GANs leverage a form of adversarial learning where a generator incrementally learns to generate realistic samples, while a discriminator simultaneously learns to discriminate between real and generated data. They had originally been proposed as a generative approach for continuous data, such as images, but have later found application also for discrete data, despite their well-known "non-differentiability issue". In fact, several GANs have recently been proposed for text generation [24, 16, 25] and have achieved encouraging results in comparison to comparable maximum likelihood approaches; in particular, RelGAN [16] has outperformed state-of-the-art (SOTA) results.

In general, an effective NLG model should enjoy two main properties: the syntactic correctness and the semantic coherence of the generated sentences. Although both these aspects are key for the usability of NLG models, often only the syntactic aspect is taken into account during training and evaluation. For this reason, in this paper we propose a new model – the Controlled TExt generation Relational Memory GAN (CTERM-GAN) – which explicitly takes into account both syntactic and semantic aspects. CTERM-GAN consists of the following main modules: 1) a generator based on a *relational memory with self-attention* conditioned on an external input; 2) a *syntax discriminator* which learns to discriminate between real and generated sentences based on syntactic correctness; and 3) a *semantic discriminator* trained to assess whether a sentence is coherent with the external conditioning. Like a conventional NLG GAN, this model is trained to generate syntactically-correct sentences; however, the inclusion of both a second discriminator and a generator influenced by an external input allows improving the coherence of the generated sentences. The experimental results in Section 4 show that the proposed model has been able to retain or increase syntactic accuracy while at the same time drastically improving semantic coherence.

2 Related Work

Using GANs for discrete data generation is still a developing research area. The two main research directions are along reinforcement learning (RL)-based and reparametrization-based models. The former use RL algorithms to circumvent the non-differentiability issue and include SeqGAN [24]

and several other models [8, 13, 3]. The latter, instead, leverage continuous approximations of discrete sampling [25, 4, 16]. Recently, RelGAN [16] has introduced a Gumbel-softmax relaxation of discrete sampling [11], alongside a multiple discriminator model to extract different features from the sentences. RelGAN has outperformed all other compared GAN models on a variety of challenging datasets. The idea of using external conditioning to improve or control NLG has also been widely explored [6, 22, 20, 21]. For instance, TopicRNN [6] increases the probability of words related to a control topic during sentence generation. SentiGAN [20] has proposed a model that generates sentences conditioned on a sentiment by using multiple generators, one per sentiment, and a multi-label discriminator. TCNLM [21] uses a neural topic model to first extract the latent topic, and then feeds it to a mixture of expert language models, each specialized for an individual topic. Differently from them, in our model we use a single generator that controls the text generation by means of a Relational Memory which has been exposed to the conditioning input. In turn, two distinct discriminators respectively assess the syntactic quality and coherence to the input of the generated sentences. Our model is independent of the specific nature of the conditioning input and, as such, it is the only one to date that can be used for *both* topic-conditioned and sentiment-conditioned generation, and, in principle, other flavors.

3 Model

This section presents the main details of CTERM-GAN (namely, the generator and the syntax and semantic discriminators). [1]

3.1 Loss function

As training loss, we have used a non-saturating GAN loss function [7], that, considering the double-discriminator model, is extended as:

$$l_D = \frac{1}{m}\sum_{i=1}^{m}\Big[\Big(log(D^S(x_r)) + log(1 - D^S(G(x_z)))\Big) + \beta\Big(log(D^T(x_r)) + log(1 - D^T(G(x_z)))\Big)\Big]$$

[1] All the training information and hyperparameters are described in Appendix D. We will release all our code publicly after the anonymity period.

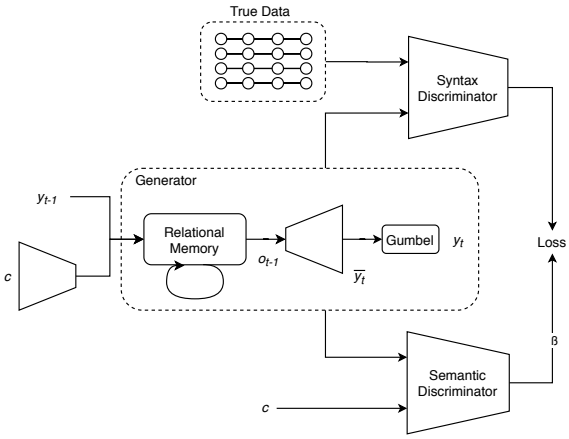

Figure 1: CTERM-GAN architecture

$$l_G = \frac{1}{m}\sum_{i=1}^{m}\Big[-log(D^S(G(x_z))) - \beta\log(D^T(G(x_z)))\Big]$$

where β is a hyperparameter that assigns a relative weight to the topic discriminator with respect to the syntax one. β plays an important role during training since, if it is too low, the model ignores the conditioning due to the limited penalty. Conversely, a too high a value would give too much importance to the conditioning, affecting the quality of the generated sentence.

3.2 Generator

The generator is based on a Relational Memory with self-attention [18, 16]. This model updates its "internal values" and produces its final output by selecting from its memory cells with a self-attention mechanism. Leveraging an idea similar to that of image-based conditional GANs [15], we introduce an external conditioning into the generator. First, given the conditioning input $c \in \mathbb{R}^d$, the model computes an embedding γ_t for c using function $f_\theta : \mathbb{R}^d \to \mathbb{R}^m$, with $m < d$. Function f_θ has been implemented using a feed-forward neural network with a self-attention layer. The conditioning vector c may originate from any type of different source as long as it remains consistent during the individual training. Depending on the required task, as shown in the experiment phase, it will change. This vector c is the only link between the conditioning and the generative model; its influence on the final output will be crucial for the conditioning of the generated sentence. f_θ has been adopted to give the model the ability to learn the best manipulation of the conditioning vector to insert into the memory.

Given the previous output y_{t-1} and the processed conditioning γ_t at the current time-step, the memory updates its state, h_t, using a double-step sequential update and computes the next value, o_t, as in Eqs. 1-2:

$$\gamma_t = f_\theta(c) \tag{1}$$

$$o_t,\ h_t = f_{RM}(h_{t-1}, \gamma_t, y_{t-1}) \tag{2}$$

where f_{RM} represents the memory cell update. The distribution over the vocabulary of the next word is evaluated using the memory output o_t as in Eq. 3 with a feed-forward layer. Then, the next soft word, $\hat{y}_t$, is sampled using the Gumbel-softmax relaxation [11] with temperature T (Eq. 4). The temperature value greatly influences the quality-diversity trade-off; more details on these parameters are provided in Appendix D.

$$\bar{y}_t = f_\alpha(o_t) \tag{3}$$

$$\hat{y}_t \sim \text{Gumbel-softmax}(\bar{y}_t, T) \tag{4}$$

3.3 Syntax discriminator

The syntax discriminator takes as input either a real sentence, $r = (r_1, \ldots, r_n)$, or a generated one, $g = (g_1, \ldots, g_n)$. Similarly to many other works (e.g., [9, 23]), the discriminator first transforms its input into an embedding matrix. This embedding allows learning a transformation that condenses the information brought in by each word optimally for any given task. The syntax discriminator is then built using two convolutional layers with ReLU activation functions, followed by a self-attention layer, again followed by two other convolutional layers with ReLU activation functions. The self-attention layer is used to attend to the output of the previous convolutional layer and select the most useful features. The final layers generate the decision.

3.4 Semantic discriminator

One of the main novelties of our approach is the explicit targeting of semantic coherence. This is achieved by augmenting the model with a semantic discriminator trained to recognize whether the input sentence is consistent with the conditioning input, c, or not. To produce its output, this discriminator receives as input both c and either a real sentence, $r = (r_1, \ldots, r_n)$, or the output of the generator, $g = (g_1, \ldots, g_n)$. The proposed architecture is composed of two networks: one for the sentence and one for input c. The first network consists of a feed-forward layer which acts as an embedding, followed by four convolutional and self-attention layers with ReLU activation functions which extract a latent vector expected to represent the main characteristics of the sentence. In the second network, input c is passed through a linear layer to suitably reduce or expand its size to that of the output vector of the first network. The two outputs are then combined and the final decision is computed with a feed-forward layer.

4 Experiments

The CTERM-GAN model has been tested over two tasks: topic conditioning and sentiment conditioning. The former consists of generation guided by exogenous text input, while the latter focuses on the generation of sentences given a sentiment.
In all experiments, we have separately trained the generator for 150 epochs and the semantic discriminator for 300 epochs before the adversarial training was started. After that, the generator has been trained for 2 batches and the discriminators for 3 batches in each adversarial epoch. For the β weight in the loss function, several values were tested and the optimal value was found to be 0.1.

4.1 Topic conditioning

In these experiments we have compared the conditioned text generation of CTERM-GAN with that of the state-of-the-art adversarial architectures – SeqGAN [24], RelGAN [16], and TGVAE [22] – and a classic auto-regressive LSTM language model with an initial conditioning, in terms of both syntactic and semantic quality. The main goal is to ensure a good quality for the generation by introducing a conditioning on the semantic of the sentence. In this task, the conditioning consists of the word distribution for a topic extracted from a sentence, either provided by the user or, as in our case, sampled from the dataset. Any type of topic model can be adopted: in our case, an LDA model [2] has been trained on a starting dataset in order to have a distribution of the topics covered within the corpus. The LDA model, both in training and in inference, given an input sentence, builds a distribution on the vocabulary. In turn, this distribution influences the model's sentence generation thanks to its inclusion in the generation process. Most likely, improving the quality of the topic extraction is likely to improve the final results of the model. Eventually, the extracted distribution is used as the condition-

Dataset	Image COCO	EMNLP	APNews	BNC
Train size	10000	270000	54000	15000
Test size	10000	10000	2000	1000
Sequence Length	37	51	1801	105550
Train Vocabulary	6612	5230	32430	41496
Topic Vocabulary	3872	4265	8564	10345

Table 1: Topic Conditioning datasets.

Model	COCO			EMNLP News			APNews			BNC		
	B-2	B-4	KL	B-2	B-4	KL	B-2	B-4	KL	B-2	B-4	KL
LSTM Topic	0.734	0.315	0.104	0.532	0.107	**0.155**	0.689	0.196	**0.060**	0.688	0.203	**6.69e-4**
RelGAN	0.752	0.339	0.468	0.675	**0.339**	0.941	0.738	0.258	0.291	0.760	**0.282**	8.89e-4
TGVAE (T=10)	-	-	-	-	-	-	0.584	0.202	-	0.518	0.173	-
TGVAE (T=50)	-	-	-	-	-	-	0.629	0.210	-	0.535	0.188	-
CTERM GAN	**0.782**	**0.404**	**0.053**	**0.738**	0.326	0.447	**0.794**	**0.341**	0.194	**0.761**	0.242	7.62e-4

Table 2: Topic conditioning results. The values for TGVAE are reproduced from [22] where KL values are not available.

ing input, c, for the relational memory during the generation, as described in Section 3.

Datasets To evaluate the topic-conditioned text generation we have used four benchmark datasets: COCO Image Caption [5], EMNLP2017 WMT News [9], APNews [2] and BNC [1]. The COCO Image Caption dataset is composed of image captions that we have preprocessed following [26]. The EMNLP2017 WMT News dataset consists of longer sentences than COCO's that were also preprocessed according to [26]. APNews is a dataset of Associated Press' news articles from 2009 to 2016, and the BNC dataset is the written portion of the British National Corpus. These datasets are highly diverse in terms of type of texts, covering books, essays, journals and news. More specific information about these datasets are shown in Table 4.

Evaluation As evaluation measures, we have adopted corpus BLEU [17] to assess syntactic quality and the Kullback-Leibler (KL) divergence [12] between the topic used for conditioning and the topic extracted from the generated sentence to assess semantic coherence. A low KL value means that the distribution inferred from the output of the model is similar to the one extracted from the conditioning input sentence and used as conditioning vector c. This implies that the semantic conditioning has been carried out successfully.

Results Table 2 shows the results of the topic-conditioning experiments over the four datasets. The BLEU results (columns B-2, B-4) empirically demonstrate that the syntactic quality of the text generated by CTERM-GAN is often superior to that of the state-of-the-art GANs for text generation. The results for TGVAE are shown for both 10 topics, as used for CTERM-GAN , and for its best reported configuration. In turn, the KL results show that CTERM-GAN has also achieved better coherence to the conditioning topic than RelGAN for all datasets. For some datasets, the LSTM-based model has obtained a lower (i.e. better) divergence, yet at a considerable reduction in terms of BLEU scores.

[2] https://www.ap.org/en-gb/

4.2 Sentiment conditioning

In these experiments we have compared the sentiment-conditioned text generation of CTERM-GAN with that of SeqGAN [24], SentiGAN [20] and an RNNLM baseline [14]. Following the experiments carried out in [20], the conditioning has been performed based on only two sentiments, positive or negative. In this case, the conditioning vector, c, taken as input by CTERM-GAN is a one-hot binary variable representing the desired sentiment. The RNNLM and SeqGAN models have been trained separately on the two sentiments by treating positive and negative sentences as two separate datasets, while SentiGAN and the proposed model have been trained jointly. This procedure makes the results comparable although it is clear how the flexibility of SentiGAN and CTERM-GAN makes these models more general.

Model	CR			MR		
	Sentiment	Novelty	Diversity	Sentiment	Novelty	Diversity
RNNLM	0.552	0.399	0.663	0.662	0.267	0.691
SeqGAN	0.632	0.437	0.619	0.717	0.298	0.641
SentiGAN (k=1)	0.731	0.479	0.668	0.803	0.344	0.711
SentiGAN (k=2)	0.803	**0.549**	0.741	**0.885**	**0.395**	**0.741**
CTERM GAN	**0.869**	0.539	**0.746**	**0.885**	0.270	0.700

Table 3: Sentiment conditioning results. The values of models other than CTERM-GAN are reproduced from [20].

Dataset We have used two datasets, Movie Reviews (MR) [19] and Customer Reviews (CR) [10], where individual sentences are annotated as either positive or negative. The Movie Reviews dataset consists of user reviews of movies, with $2,133$ positive and $2,370$ negative sentences. The Customer Reviews dataset consists of $1,500$ reviews of products sold online, with positive/negative annotation at sentence level. For this task, only sentences of length shorter than 15 words have been retained, to be able to use the same preprocessing as [20].

Evaluation For this task, we have classified the generated sentences in terms of their sentiment using a Bidirectional-LSTM as classifier. In addition, we have evaluated two quality metrics: 1) the *novelty* of each generated sentence (Eq. 5) using the definition from [20], where JS is the Jaccard similarity and C_j are the training set sentences. The novelty measures the diversity between the generated data and the training corpus; and 2) the *diversity* metric (Eq. 6), a measure of the model's ability to generate diverse sentences and avoid mode collapse.

$$Novelty(S_i) = 1 - \max\{JS(S_i, C_j)\}_{j=1}^{j=|C|} \quad (5)$$

$$Diversity(S_i) = 1 - \max\{JS(S_i, S_j)\}_{j=1}^{j=|S|,\, j\neq i} \quad (6)$$

Results Table 3 shows the results from the sentiment-conditioned experiments. CTERM-GAN has been able to achieve a remarkable performance trade-off, with the best sentiment classification accuracy and diversity over the CR dataset, and the same sentiment classification accuracy as SentiGAN (k=2) on the MR dataset, yet with significantly decreased novelty and diversity. While the performance of CTERM-GAN and SentiGAN may be regarded as comparable overall, we emphasize once again that the proposed model is not specialized for sentiment conditioning or any specific types of conditioning.

5 Conclusion

In this short paper we have proposed the Controlled TExt generation Relational Memory GAN (CTERM-GAN), a model aiming to generate sentences that are both syntactically correct and semantically coherent. The proposed model leverages a Relational Memory that is influenced by a conditioning input and is used to generate sentences, alongside two discriminators that respectively assess the sentences' syntactic quality and semantic coherence. The experimental results over topic-conditioned and sentiment-conditioned tasks have shown that the proposed model has performed at the same level or above that of SOTA GANs and relevant baselines. In the near future, we will explore text generation with other type of conditioning inputs such as writer's style and images to further probe the generality of the proposed model.

6 Acknowledgment

We thank Professor Stefano Ceri for the support, and the valuable comments and ideas.

References

[1] The British National Corpus, version 3 (bnc xml edition). In *Distributed by Bodleian Libraries, University of Oxford, on behalf of the BNC Consortium.*, 2007.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[3] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv:1702.07983*, 2017.

[4] Liqun Chen, Shuyang Dai, Chenyang Tao, Dinghan Shen, Zhe Gan, Haichao Zhang, Yizhe Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. *arXiv:1809.06297*, 2018.

[5] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár,

and C Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv:1504.00325*, 2015.

[6] Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. TopicRNN: a recurrent neural network with long-range semantic dependency. *arXiv: 1611.01702*, 2016.

[7] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv: 1406.2661*, 2014.

[8] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long Text Generation via Adversarial Training with Leaked Information. *arXiv:1709.08624*, 2017.

[9] 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.

[10] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[11] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv:1611.01144*, 2016.

[12] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[13] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. *CoRR*, abs/1705.11001, 2017.

[14] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. *INTERSPEECH 2010*, 2010.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

[16] Weili Nie, Nina Narodytska, and Ankit B. Patel. RelGAN: Relational Generative Adversarial Networks for Text Generation. *ICLR 2019*, pages 1–20, 2019.

[17] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318. Association for Computational Linguistics, 2002.

[18] Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Théophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. 2018.

[19] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*, pages 1631–1642, 2013.

[20] Ke Wang and Xiaojun Wan. SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4446–4452, 2018.

[21] Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. Topic Compositional Neural Language Model. *arXiv:1712.09783*, 2018.

[22] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. Topic-guided variational autoencoders for text generation. *CoRR*, abs/1903.07137, 2019.

[23] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv:1609.05473*, 2016.

[24] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 2852–2858, 2017.

[25] 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.

[26] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *CoRR*, abs/1802.01886, 2018.

Appendix

A Datasets Information

Additional information about the datasets used for conditioned text generation are presented in Table 4. The data fed into the topic model were preprocessed by removing stop-words and punctuation so as to retain only those words that provided semantic information. Moreover, only for the biggest corpora, the top 0.1% frequent words and the words that appeared in less than 100 documents have been removed. The sequence length has been capped to 60 in the APNews and BNC datasets for computational reasons.

Dataset	Image COCO	EMNLP	APNews	BNC
Training Size	10000	270000	54000	15000
Test Size	10000	10000	2000	1000
Sequence Length	37	51	1801	105550
Training Vocabulary	6612	5230	32430	41496
Topic Vocabulary	3872	4265	8564	10345

Table 4: Text generation dataset information.

Dataset	MR	CR
Positive Sent.	2,133	1,024
Negative Sent.	2,370	501

Table 5: Sentiment generation dataset information.

B Generated Sentences

Tables 6 and 7 show examples of generated sentences from the COCO and CR datasets.

COCO Dataset	
Generated	**Conditioning**
handicapped bathroom features a tub and sink in a bathroom mirror .	entryway to a bathroom , toilet and sink prominent
a large passenger jet flying through the sky .	contrails can be seen from a descending jet .
a white kitchen with white walls and counter tops .	a kitchen with tiled floors and counter tops

Table 6: Examples from the COCO dataset: generated and conditioning sentence.

C Model Details

The self-attention layer (Eq. 1) is used to adjust the content extracted from the conditioning vector $\tilde{\gamma}$ at word level. Three matrices are created and updated during training: Q queries, K keys and V values. The final value y is computed as in Eq. 7:

$$y = \sigma\Big(f(Qx \cdot Kx)\Big) \cdot Vx \tag{7}$$

considering f as a normalization function and σ the softmax function.

D Training Details

In every experiment, we separately train the generator for 150 epochs and the semantic discriminator for 300 epochs before the adversarial training is started. After that, the generator is trained for 2 batches and the discriminators for 3 batches in each adversarial epoch.

	CR Dataset
Positive	as said before this works perfectly .
	the nokia 6600 is a decent extension of the smartphone line .
Negative	overall , the player is crap .
	this is a terrible company with a bad product .

Table 7: Examples of generated sentences for the CR dataset.

We have obtained the best results with the non-saturating loss [7], adding a component derived from the sentiment discriminator with a weight of β. Several values were tested for β, and the optimal value was found to be 0.1, as shown in Fig. 2. Other details are in Appendix E.

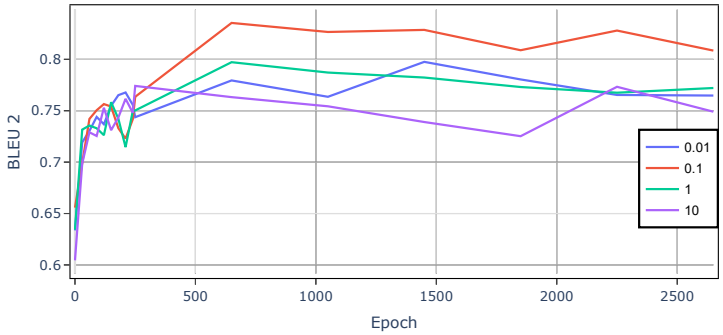

Figure 2: BLEU 2 score for different values of β.

The temperature, T, of the Gumbel-softmax operator allows us to trade off quality and diversity of the generated sentences. A high temperature produces a one-hot-like vector, while a lower temperature yields an output that takes into account several different values. A lower temperature, thus, allows for better diversity in generation; often a rising temperature is preferred during training in order to have more initial exploration and better exploitation afterwards. Considering that the value comes into direct contact with the weight gradient, having a lower temperature allows a smoother procedure that makes it easier for the model to converge.

Function f_α, shown in Eq. 3, maps the memory output at time t to values in $\mathbb{R}^V$. We have modeled f_α as a feed-forward neural network. The output of the memory depends on the size of the memory itself; in the experiments we have directly used the memory cells reshaped into one dimension. The number of memory cells is a hyperparameter of the model. We have tested with 512 and 1024 cells; the 1024 cells version showed slightly better results at the cost of a model with almost twice the number of parameters.

The Gumbel-softmax operation is shown in Eq. 8. It is fundamental for the input $\bar{y}_t$ not to be a distribution in $\mathbb{R}^V$, otherwise the uniform noise would be the dominating component of the final output. This would lead to a completely random model.

$$\begin{aligned} U_t &\sim Uniform(0,1) \\ g_t &= -log(-log\ U_t) \\ y_t &= \sigma((\bar{y}_t + g_t)/T) \end{aligned} \tag{8}$$

with σ the softmax function.

E Loss

The loss function used is crucial for the final result. The real advantage of adversarial models is that the discriminators can create new architectures with personalized loss functions that they learn during training.

Tests were performed using a non-saturating GAN loss function [7], shown in Eq. 9

$$
\begin{aligned}
l_D &= \frac{1}{m}\sum_{i=1}^{m}\Big[log(D(x_r)) + log(1 - D(G(x_z)))\Big] \\
l_G &= \frac{1}{m}\sum_{i=1}^{m}\Big[-log(D(G(x_z)))\Big]
\end{aligned}
\tag{9}
$$

but considering the double-discriminator model, it is transformed in:

$$
\begin{aligned}
l_D = \frac{1}{m}\sum_{i=1}^{m}\Big[&\Big(log(D^S(x_r)) + log(1 - D^S(G(x_z)))\Big) \\
&+ \beta\Big(log(D^T(x_r)) + log(1 - D^T(G(x_z)))\Big)\Big] \\
l_G = \frac{1}{m}\sum_{i=1}^{m}\Big[&\Big(-log(D^S(G(x_z)))\Big) + \beta\Big(-log(D^T(G(x_z)))\Big)\Big]
\end{aligned}
\tag{10}
$$

where β is a hyperparameter that assigns a relative weight to the topic discriminator with respect to the syntax one. This hyperparameter plays an important role during training since, if it is too low, the model ignores the conditioning because of the limited penalty. Conversely, a too high a value for β would give too much importance to the conditioning, affecting the quality of the generated sentence.

F Topic Model

The topic model chosen to extract the topic from the control sentence is LDA. It has been trained separately over each dataset before the GAN training procedure. The number of topics used has been decided measuring the coherence over the dataset and taking the maximum value. Examples of coherence values for the COCO and EMNLP datasets are shown in Fig. 3.

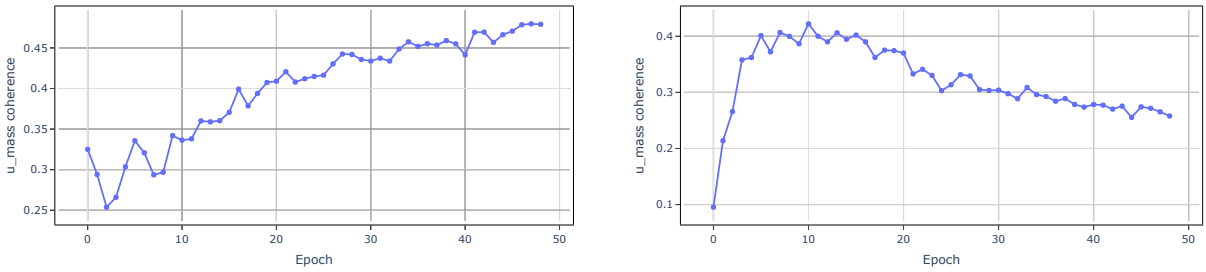

Figure 3: C_v coherence for the COCO and EMNLP datasets.