A Differentiable Generative Adversarial Network for Open Domain Dialogue

Asier López Zorrilla, Mikel deVelasco Vázquez and M. Inés Torres

Abstract This work presents a novel methodology to train open domain neural dialogue systems within the framework of Generative Adversarial Networks with gradient based optimization methods. We avoid the non-differentiability related to text-generating networks approximating the word vector corresponding to each generated token via a *top-k softmax*. We show that a weighted average of the word vectors of the most probable tokens computed from the probabilities resulting of the top-k softmax leads to a good approximation of the word vector of the generated token. Finally we demonstrate through a human evaluation process that training a neural dialogue system via adversarial learning with this method successfully discourages it from producing generic responses. Instead it tends to produce more informative and variate ones.

1 Introduction

Open domain dialogue systems or chatbots are systems deployed to interact with humans offering coherent responses according to the dialogue history. Unlike task-oriented dialogue systems, there is no specific goal to be achieved during the interaction by the system. The only goal is to generate appropriate, relevant, meaningful and human-like utterances.

This area of research has gained an increasing amount of interest from the community since the advent of sequence-to-sequence neural network models [22]. These

Asier López Zorrilla
University of the Basque Country UPV/EHU, e-mail: asier.lopezz@ehu.eus

Mikel deVelasco Vázquez
University of the Basque Country UPV/EHU, e-mail: mikel.develasco@ehu.eus

M. Inés Torres
University of the Basque Country UPV/EHU, e-mail: manes.torres@ehu.eus

neural networks are capable of processing and generating sequences of data of arbitrary length, which makes them very suitable for this research [24, 21]. The task of open domain dialogue generation can easily be cast as a sequence transduction problem, where the input is the sequence of words corresponding to the last user's utterance, and the output are the words of the system's response. It is also possible to condition the output of the network to a broader dialogue context or other knowledge sources in order to increase the coherence of the responses [19, 6], but in this work we will not research in that direction.

These neural models are usually learnt from corpora composed of input utterance-response pairs, via supervised learning. Movies subtitles, Twitter or online forums can be used as the source of these data. In this framework, the neural network is trained to minimize a distance between the generated response and the desired one. Even though interesting performances can be obtained with this procedure, it frequently yields models that tend to generate dull and safe responses which appear frequently in the corpus, such as *I don't know* or *I'm sorry*.

We build upon Generative Adversarial Networks (GANs) [7] to overcome this problem and to increase the overall variety in the responses of the neural dialogue model, as these have shown promising results in many data generation tasks. While in supervised learning a unique desired output is assigned to each input in the corpus, GANs allow many correct outputs, which makes much more sense in dialogue, and models better the one-to-many property of input-output pairs [23]. The learning methodology for GANs involves training two neural networks, a generator and a discriminator, in an adversarial fashion. The generator tries to learn a data distribution while the discriminator learns whether a given sample corresponds to the training data or has been generated by the generator. In the context of dialogue systems, the generator would be the sequence-to-sequence model and the discriminator would act as a Turing Test.

GANs were first successful in image generation tasks. More recently text-related problems, such as machine translation [25], text generation [27, 26] or image captioning [20] have also been tackled within this framework. GANs have also been applied in the research of dialogue systems, yet only on a few occasions. [5] and [11] experiment with training discriminators that could measure the quality of the utterances generated by chatbots. On the other hand [14] and [9] go a step further and train neural dialogue systems via adversarial learning, but with the drawback that they make use of reinforcement learning instead of gradient-based optimization methods. This is due to text being represented as a sequence of discrete tokens, which breaks the differentiability of the discriminator's output with respect to the generator's parameters, as explained in Section 3.

In this context, the contributions of our work are twofold. First, we present a novel methodology to avoid this non-differentiability: the *top-k softmax*. Since the top-k softmax allows to plug-in the output of the generator into the discriminator in a differentiable manner, our approach is simpler and easier to implement than other dialogue systems trained in the GAN framework. Second, we demonstrate that training a neural dialogue system via adversarial learning with this method successfully

discourages it from producing generic responses, and that it often leads to more informative responses too.

The rest of the paper is organized as follows. In Section 2 we specify the chosen architecture for the sequence-to-sequence dialogue model and the baseline training procedure. In Section 3 we describe the proposed GAN for dialogue generation based on the top-k softmax and compare it to alternative approaches to deal with the differentiability problem. In Section 4 we give all the details about our experimental setup and hyper-parameter choice. Section 5 shows the results of two experiments to validate our proposal. We conclude with the final remarks in Section 6.

2 Sequence-to-sequence dialogue model architecture

The chosen architecture for the dialogue model is a standard sequence-to-sequence network with attention [1]. Given an input sequence of length T of discrete integer tokens $x = x_1, x_2, ..., x_T$, the corresponding sequence of vectorial word representations $\mathbf{v} = \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_T$ can be obtained via the word vector matrix $\mathbf{W}$, just by taking the corresponding row $\mathbf{v}_i = \mathbf{W}[x_i]$ per each token x_i. The size of $\mathbf{W}$ is $V \times D$, where V is the vocabulary size and D the dimension of each word vector. The encoder takes this sequence of vectors and produces another sequence of vectors of the same length $\mathbf{h} = \mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T = \text{encoder}(\mathbf{v})$. In our work the encoder is a deep bidirectional Long Short Term Memory (LSTM) Recurrent Neural Network (RNN).

To proceed with the generation of the output sequence $y = y_1, y_2, ..., y_\tau$, a global attention mechanism is applied as in [17]. At the time step t of the generation, the decoder is fed with the discrete integer token generated at previous time step, y_{t-1}. Then the corresponding word vector $\mathbf{W}[y_{t-1}]$ is input to the decoder's RNN and this outputs $\mathbf{o}_t$. Of course, due to the architecture of RNNs, $\mathbf{o}_t$ is conditioned, though implicitly, not only to y_{t-1} but also to all the previously generated tokens. In our experiments this neural network is also a deep LSTM. $\mathbf{o}_t$ is then transformed to $\tilde{\mathbf{o}}_t$ via a multilayer perceptron (MLP) that takes as input o_t and also $\mathbf{c}_t$, the context-vector produced by the attention mechanism at time step t. $\mathbf{c}_t$ is a weighted average of the encoder's output vectors:

$$\mathbf{c}_t = \sum_{j=1}^{T} a_{jt} \mathbf{h}_j \,, \tag{1}$$

where a_{jt} is the score between $\mathbf{h}_j$ and $\mathbf{o}_t$, i.e., how much attention should be put on the output of the encoder at the encoding time step j on the time step t of the decoding phase. a_{jt} is a softmax-normalized scalar output of another MLP, that takes as input $\mathbf{h}_j$ and $\mathbf{o}_t$, and outputs $\overline{a_{jt}}$. With the softmax normalization we ensure that all the scores at time step t are positive and sum one:

$$a_{jt} = \frac{\exp(\overline{a_{jt}})}{\sum_{j'=1}^{T} \exp(\overline{a_{j't}})} \tag{2}$$

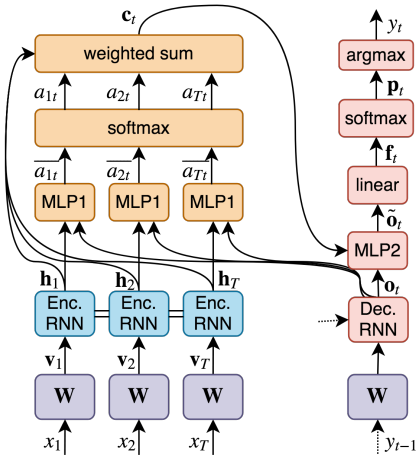

Fig. 1 A diagram of the chosen sequence-to-sequence network: blue transformations refer to the encoder, orange to the attention mechanism, purple to the word matrix (shared between the encoder and decoder), and red to the decoder. For simplicity, only the time step t of the decoding is shown.

Finally, $\tilde{\mathbf{o}}_t$ is linearly projected to a vector of dimension V: $\mathbf{f}_t = \text{linear}(\tilde{\mathbf{o}}_t)$. This vector represents an unnormalized probability distribution over all the possible words in the vocabulary. A softmax normalization is then applied to $\mathbf{f}_t$ to get $\mathbf{p}_t = \text{softmax}(\mathbf{f}_t)$, the normalized version of $\mathbf{f}_t$. The output token at time step t, y_t, can be sampled from $\mathbf{p}_t$ taking the argument of the maxima:

$$y_t = \arg\max_i \left(\mathbf{p}_t[i]\right) \tag{3}$$

Generation stops at time τ, when y_τ corresponds to the end-of-sequence token. The architecture of the network is summarized in Figure 1.

Maximum Likelihood Estimation via Supervised Learning

As aforementioned, this neural network can be trained from a corpus composed of input-output sequence pairs via supervised learning. A maximum likelihood estimation (MLE) of the parameters of the network can be carried out by minimizing the word level cross entropy loss L_{MLE}:

$$L_{MLE} = \frac{1}{|\mathscr{C}|} \sum_{x,s \in \mathscr{C}} \frac{1}{|s|} \sum_{t=1}^{|s|} -\log \mathbf{p}_t[s_t] \,, \tag{4}$$

where $\mathscr{C}$ is a corpus composed of pairs of inputs x and desired outputs s, s_t each of the words in s, and $\mathbf{p}_t[s_t]$ the output of the network in the t-th time step corresponding to the token s_t. We omit the output's dependence on x to keep the notation simple.

During training we employ the teacher forcing strategy, i.e., in the t-th step of the decoding we feed the ground true token s_{t-1} to the decoder's RNN instead of the prediction y_{t-1}. We experimented with other sampling techniques such as scheduled sampling [2], but we found no improvement.

3 Sequence Generative Adversarial Network training

In the context of dialogue systems, the generator network in the GAN is the sequence-to-sequence dialogue model, which produces a response y to the input utterance x. The discriminator is another network that acts like a Turing Test: it takes an input utterance x and a response r as inputs, and outputs a scalar between 0 and 1 representing the network's confidence level on r being produced by a chatbot. Namely, the lower the output of the discriminator is, the more human-like r is according to the discriminator's criteria.

The procedure to train the dialogue system in this framework involves iteratively updating the generator and the discriminator. The generator is trained to fool the discriminator and make it think that its responses are human-like, and in contrast the discriminator is trained to distinguish between human and bot responses.

Let us now define the losses to be minimized in this two optimization procedures. Given a batch of input utterances, responses and labels indicating whether each response has been generated by a bot or a human, the discriminator's parameters will be updated to minimize the next cross-entropy loss:

$$L_D = \frac{1}{|\mathscr{B}_D|} \sum_{x,r,l \in \mathscr{B}_D} -\left[l \cdot \log a + (1-l) \cdot \log (1-a)\right] , \tag{5}$$

where $\mathscr{B}_D$ is a batch composed of tuples of input utterances x, responses r and boolean labels l, and a the output of the network given x and r.

The objective for the generator is just to minimize the output of the discriminator when the latter is fed with a batch of input utterances and the responses of the generator to those same input utterances:

$$L_G = \frac{1}{|\mathscr{B}_G|} \sum_{x \in \mathscr{B}_G} a , \tag{6}$$

where $\mathscr{B}_G$ is a batch composed of input utterances x. a is the output of the discriminator given x and y, where y is the output of the generator given x.

The differentiability problem

We have already described the architecture of the generator in Section 2. On the other hand, the discriminator is a composed of two deep bidirectional LSTM-RNNs, for x and r respectively, followed by some fully-connected layers. Before being processed by the RNNs, both x and r integer sequences are converted to word vector sequences via the same word vector matrix $\mathbf{W}$, as explained in Section 2.

Being these the network architectures, it is not possible to differentiate L_G (Equation 6) with respect to the parameters of the generator. The problem arises with the argmax operation in the sequence of transformations that converts $\mathbf{f}_t$ into $\mathbf{u}_t$:

$$\mathbf{f}_t \xrightarrow{\text{softmax}} \mathbf{p}_t \xrightarrow{\text{argmax}} y_t \xrightarrow{\mathbf{W}[y_t]} \mathbf{u}_t , \tag{7}$$

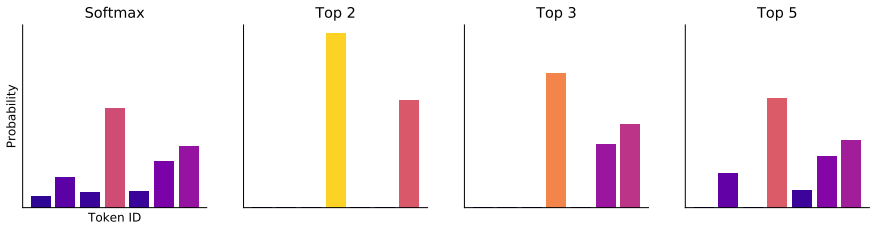

Fig. 2 On the left, a graphical example of a softmax normalization of a $\mathbf{f}_t$ distribution. The rest of the plots show the top-k softmax normalizations of $\mathbf{f}_t$ for different values of k.

where $\mathbf{f}_t$ is the unnormalized probability distribution over all the possible words in the vocabulary in the step t of the generation, $\mathbf{p}_t$ the softmax-normalized version of $\mathbf{f}_t$, y_t the argument of the maxima of $\mathbf{p}_t$, and $\mathbf{u}_t$ is the word vector corresponding to the token y_t. Green arrows indicate that the operation is differentiable, whereas red arrows that it is not.

The top-k softmax

We propose a novel alternative computation path that approximates $\mathbf{u}_t$ in a fully differentiable manner, allowing the generator to be trained with very convenient gradient-based methods. The idea behind this path is to generate a word vector $\tilde{\mathbf{u}}_t$, hopefully similar to $\mathbf{u}_t$, as a weighted average over the word vectors corresponding to the k most probable words according to $\mathbf{f}_t$. $k \geq 2$ is an integer parameter of the transformation. In short, the differentiable computation path is as follows:

$$\mathbf{f}_t \underset{\longleftarrow}{\xrightarrow{\text{top-k}}} \mathbf{k}_t, \tilde{\mathbf{f}}_t \underset{\longleftarrow}{\xrightarrow{\text{softmax}}} \mathbf{k}_t, \tilde{\mathbf{p}}_t \underset{\longleftarrow}{\xrightarrow{\sum_i \tilde{\mathbf{p}}_t[i] \cdot \mathbf{W}[\mathbf{k}_t[i]]}} \tilde{\mathbf{u}}_t \tag{8}$$

The first operation in Equation 8 performs a selection of the *top-k* elements in $\mathbf{f}_t$. It outputs $\mathbf{k}_t$ and $\tilde{\mathbf{f}}_t$. $\mathbf{k}_t$ are the indices corresponding to the k elements in $\mathbf{f}_t$ with the highest values, and $\tilde{\mathbf{f}}_t$ are those values. In other words, $\mathbf{k}_t$ represents the most probable words, and $\tilde{\mathbf{f}}_t$ their unnormalized probabilities. The second operation is just a softmax normalization of these k probabilities. It converts $\tilde{\mathbf{f}}_t$ into $\tilde{\mathbf{p}}_t$. See Figure 2 for a graphical example. Finally, the approximated word vector that will be fed to the discriminator's RNN is computed as the weighted average of the word vectors corresponding to tokens $\mathbf{k}_t$, where the weights are the probabilities $\tilde{\mathbf{p}}_t$:

$$\tilde{\mathbf{u}}_t = \sum_{i=1}^{k} \tilde{\mathbf{p}}_t[i] \cdot \mathbf{W}[\mathbf{k}_t[i]] \tag{9}$$

Note that in the whole process the differentiability has not been broken. Therefore, and in contrary to the previous computation path (Equation 7), the partial derivatives of $\tilde{\mathbf{u}}_t$ with respect to $\tilde{\mathbf{f}}_t$ exist and are non zero. In Section 5 we show that $\tilde{\mathbf{u}}_t$ is a good approximation of $\mathbf{u}_t$ when k is small. In fact, $\mathbf{u}_t$ is the nearest neighbor of $\tilde{\mathbf{u}}_t$ the 98% of the times with $k = 2$.

Related approaches

Before continuing with our proposal for training the GAN, let us briefly compare the top-k softmax with alternative approaches to deal with the non-differentiable argmax operation. Apart from the aforementioned reinforcement learning-related methodologies based on [27], we are only aware of works [13] that in one way or another tackle this problem with the concrete or Gumbel-softmax distribution [18, 10]. This is a continuous relaxation of discrete random variables. In short, it transforms a probability distribution into a relaxed one-hot vector corresponding to a randomly taken sample from that distribution. That relaxed vector is different from the result of the top-k softmax in two important aspects. First, it is non-deterministic, which could be interesting but also unnecessary for our application. Second, all its elements are non-zero, which means that approximating a word vector as a weighted average according to those probabilities would imply mixing all the word vectors in the vocabulary, which seems again inadequate for our application.

A discrete version of this transformation is the Straight-Through Gumbel-softmax estimator [3, 10], which was used by [16] and [20]. It serves to approximate the gradients of a one hot vector sampled according to a probability distribution. Thus it avoids the problem of averaging over all the word vectors, but it is still non-deterministic. Moreover, the operation is still non-differentiable. Even though this method provides an estimation of the gradients in this scenario, but using it could be risky because it might cause discrepancies between the forward and backward passes, as stated in the original work [10].

Training procedure

The top-k softmax allows L_D to be differentiable with respect to the parameters of the generator. Thus gradient-based optimization methods can be applied to train both the generator and the discriminator. Let us now specify the general training loop and the pretraining strategies applied in this work.

Prior to the training of the dialogue system, we pretrain the word vector matrix in the same corpus that will be used later. Following the work of [14] and [9], we also pretrain the generator using the MLE criteria, and the discriminator with the responses generated by the pretrained generator and with responses from the corpus. In order to stabilize the rest of the training process and to avoid the catastrophic forgetting phenomenon of the discriminator, each time we sample a response of the generator to a given input, we add it to a corpus of generator's turns $\mathscr{C}_D$.

Now we enter the main training loop, where the generator and the discriminator will be trained adversarially. This loop will be run for many iterations. We start it training the generator to minimize the output of the discriminator according to Equation 6 during a number of iterations. Then we increase the corpus $\mathscr{C}_D$ with the current state of the generator, and train the discriminator during another number of iterations. More recent input-response pairs are taken with a higher probability than the older ones from $\mathscr{C}_D$ when training the discriminator.

We finally repeat this process of training the generator, adding samples to $\mathscr{C}_D$ and training the discriminator, but this time training the corpus with the MLE criteria. This approach is also taken in [14] and [9], and it aims at stabilizing the training process. In order to further stabilize it, we reduce the learning rate of the training optimizer throughout the global iterations.

This whole procedure is summarized in the Algorithm 1.

Algorithm 1 An Adversarial Training strategy for Neural Dialogue Models.

Require: Generator G, Discriminator D, Corpus $\mathscr{C}$, training hyper-parameters.
Pretrain word vector matrix $\mathbf{W}$ on $\mathscr{C}$.
Pretrain G minimizing L_{MLE} (Equation 4).
Initialize $\mathscr{C}_D$ with G's responses y to some inputs x.
Pretrain D minimizing L_D (Equation 5).
for the number of total iterations, and with a decaying learning rate **do**
  Update G minimizing L_G on inputs x in $\mathscr{C}$ (Equation 6).
  Add (x, y) pairs to $\mathscr{C}_D$ using G.
  Update D minimizing L_D.
  Update G minimizing L_{MLE} on $\mathscr{C}$.
  Add (x, y) pairs to $\mathscr{C}_D$ using G.
  Update D minimizing L_D.

4 Experimental Setup

All the experiments in this work were carried out with the OpenSubtitles2018 corpus [15], which is composed of around 400M utterances from movie subtitles. As proposed in [24], since the turns are not clearly indicated, we treat each utterance as the desired output for the previous one.

As for the text preprocessing, we removed some symbols and converted all the names, numbers and places to tags $<person>$, $<number>$ and $<place>$, respectively. This was done with the Spacy entity recognizer [8]. Finally we defined the vocabulary with most 30000 frequent words, and deleted every other token from the corpus. We pretrained 300 dimensional word vectors of those tokens on the corpus, with FastText [4]. These are then optimized again throughout the training process.

Let us now give details about the architecture of the sequence-to-sequence generator. The deep bidirectional RNN encoder is made of two LSTM networks (one per direction) of 4 layers, 512 cells each. On the other hand, the decoder's LSTM has 4 layers of 1028 cells. The MLP that converts $\mathbf{o}_t$ and $\mathbf{c}_t$ into $\tilde{\mathbf{o}}_t$ (see Section 2 for more details) has one *leaky*-ReLU layer. The size of $\tilde{\mathbf{o}}_t$ is 500. The MLP that computes the attention score has two layers. The first one is a 250-sized hyperbolic tangent layer, and the second is a linear output layer that computes the scalar score.

Regarding the discriminator, its two deep bidirectional encoders share the same architecture: two LSTM networks of two layers, 128 cells each. This vector is then fed to a MLP of two layers: a *leaky*-ReLU layer of size 100 followed by a single sigmoidal unit. The chosen value for the k parameter of the top-k softmax was 2.

The most promising hyper-parameters we have found for the training procedure are summarized next. First of all, we used the Adam optimizer [12] with batch size of 256 throughout all the optimization processes. We pretrained the generator during 50000 training iterations with a fixed learning rate of 0.001. We sampled 125000 responses from that generator and then pretrained the discriminator during 1000 iterations, with the same learning rate. All the batches fed to the discriminator were balanced: there was a human example per each generator's example. Human and generator's example were uncorrelated; they did not share the input.

The main iteration loop was run 200 times. The initial learning rate was 0.001 with a decaying factor of 0.995 when training the discriminator and the generator with the MLE criteria. It was ten times smaller when training the generator to minimize the output of the discriminator. Every MLE step was run during 50 iterations, and every step of minimizing the discriminator's output was run during 35 iterations. After each of these steps, 5000 input-response pairs were sampled from the generator, and the discriminator was trained during 40 iterations.

It is worth mentioning that we did not vary the architectural hyper-parameters much during our experiments. They are similar to many other sequence-to-sequence networks in the literature. On the other hand, selecting good and stable training hyper-parameters is challenging. This requires a deeper and more specific research that we leave for future work.

5 Experiments

We now present an experimental validation of the proposed differentiable sequence generative adversarial network for dialogue generation in two series of experiments. First we validate our differentiable GAN architecture measuring the quality of the word vectors obtained with after the top-k softmax computation path presented in Equation 8. Additionally, we compare a neural dialogue trained with this computation path and with the adversarial learning procedure summarized in the Algorithm 1 with a standard MLE model.

Approximated word vectors

We fed 1000 random inputs from the corpus to the dialogue system, and computed which was the closest word vector to each approximated one according to the euclidean distance, for different values of k. With $k = 2$, the closest word vector was the correct one the 98% of the times if we consider all the produced tokens, and the 97% if we do not consider repetitions. This two percentages decrease to 83%/69% respectively with $k = 3$, and to 74%/60% with $k = 4$. Figure 3 shows this statistic for more values of k. We therefore conclude that the proposed method to make the output of the discriminator differentiable with respect to the generator's parameters is appropriate, at least with $k = 2$.

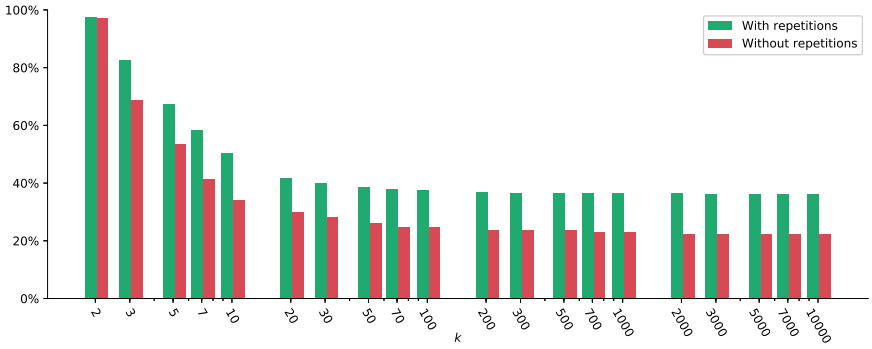

Fig. 3 Number of times that the actual word vector is the nearest neighbor of an approximated word vector produced by the top-k softmax, for different values of k.

Comparison between the MLE baseline and the GAN

Let us show a preliminary comparison between the pretrained MLE dialogue model with the final system after the adversarial learning. We asked 10 human evaluators to interact freely with the two systems during some few minutes, which resulted in dialogues of 25 turns on average. Then they were asked to decide which of them was better in terms of 1) the variety of the responses, 2) coherence and 3) informativeness. 7 out of the 10 evaluators opined that the final system was more variate and informative, and there was a draw in terms of coherence.

This can also be seen in Table 1. It shows responses to the same inputs in different stages of the training procedure. Not only are the baseline and final models compared in the table, but it also lets us gain an insight into the short-term effect of each of generator's minimizing the output of the discriminator. It tends to complex and enrich the model's responses, sometimes at the cost of losing some coherence.

6 Conclusion

We have presented a novel methodology to allow text generating models be trained in the GAN framework with gradient based optimization methods, the top-k softmax, and we have validated it in the open domain dialogue generation task. We have shown that good approximations of the word vector corresponding to each of the tokens generated by the dialogue system can be obtained with the top-k softmax. Moreover, we have demonstrated through a human evaluation process that a dialogue model trained in these conditions produces more variate and informative responses than the baseline MLE model, while being as coherent as it. Ultimately, the intersection between dialogue systems and GANs is a very promising area of research. We expect many more ideas from the two fields will be combined, and that many more applications of the GANs in the dialogue research will arise.

Table 1 Some responses to the same inputs in four different stages of the training procedure. *A* is the response of the pretrained MLE model. *B* is a response of an intermediate model just after a MLE step. *C* is the response of the same model than *B*, but just after a step of minimizing the output of the discriminator. *D* is the output of the final model.

Input utterance	Response
hi!	**A**: I'm <person> <person>. **B**: hi. **C**: hi, <person>. **D**: hello, <person>.
I love you.	**A**: I love you, too. **B**: I love you **C**: I love you too, <person> **D**: you know, I think maybe we could talk...
who is she?	**A**: I don' know. **B**: she's my friend **C**: you'll never regret again. **D**: she's my girlfriend, <person>, and...
what's the color of the sky?	**A**: it's a blue light. **B**: the blue blue. **C**: it's the red of the moon. **D**: it's a beautiful blue sky
see you	**A**: <person> <person> **B**: I'm sorry, I'm sorry **C**: I'm sorry, I didn't mean to interrupt you **D**: see you later, <person>

Acknowledgements This work has been partially funded by the Basque Government under grant PRE_2017_1_0357, by the University of the Basque Country UPV/EHU under grant PIF17/310, and by the H2020 RIA EMPATHIC (Grant N: 769872).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in Neural Information Processing Systems, pp. 1171–1179 (2015)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
5. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S.: Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349 (2015)
6. Ghazvininejad, M., Brockett, C., Chang, M.W., Dolan, B., Gao, J., Yih, W.t., Galley, M.: A knowledge-grounded neural conversation model. arXiv preprint arXiv:1702.01932 (2017)

7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
8. Honnibal, M., Montani, I.: spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. To appear (2017)
9. Hori, T., Wang, W., Koji, Y., Hori, C., Harsham, B., Hershey, J.R.: Adversarial training and decoding strategies for end-to-end neural conversation models. Computer Speech & Language **54**, 122–139 (2019)
10. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
11. Kannan, A., Vinyals, O.: Adversarial evaluation of dialogue models. arXiv preprint arXiv:1701.08198 (2017)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Kusner, M.J., Hernández-Lobato, J.M.: Gans for sequences of discrete elements with the gumbel-softmax distribution. arXiv preprint arXiv:1611.04051 (2016)
14. Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., Jurafsky, D.: Adversarial learning for neural dialogue generation. arXiv preprint arXiv:1701.06547 (2017)
15. Lison, P., Tiedemann, J.: Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles (2016)
16. Lu, J., Kannan, A., Yang, J., Parikh, D., Batra, D.: Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In: Advances in Neural Information Processing Systems, pp. 314–324 (2017)
17. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
18. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016)
19. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A.C., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. In: AAAI, vol. 16, pp. 3776–3784 (2016)
20. Shetty, R., Rohrbach, M., Hendricks, L.A., Fritz, M., Schiele, B.: Speaking the same language: Matching machine to human captions by adversarial training. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)
21. Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.Y., Gao, J., Dolan, B.: A neural network approach to context-sensitive generation of conversationaesponses. arXiv preprint arXiv:1506.06714 (2015)
22. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp. 3104–3112 (2014)
23. Tuan, Y.L., Lee, H.Y.: Improving conditional sequence generative adversarial networks by stepwise evaluation. IEEE/ACM Transactions on Audio, Speech, and Language Processing (2019)
24. Vinyals, O., Le, Q.: A neural conversational model. arXiv preprint arXiv:1506.05869 (2015)
25. Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., Liu, T.Y.: Adversarial neural machine translation. arXiv preprint arXiv:1704.06933 (2017)
26. Xu, J., Ren, X., Lin, J., Sun, X.: Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3940–3949 (2018)
27. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: AAAI, pp. 2852–2858 (2017)