Real or Fake? Learning to Discriminate Machine from Human Generated Text

Anton Bakhtin[*♦] **Sam Gross**[*♦] **Myle Ott**[♦] **Yuntian Deng**[★]
Marc'Aurelio Ranzato[♦] **Arthur Szlam**[♦]
[♦] Facebook AI Research [★] Harvard University
{yolo,sgross,myleott,ranzato,aszlam}@fb.com dengyuntian@seas.harvard.edu

Abstract

Recent advances in generative modeling of text have demonstrated remarkable improvements in terms of fluency and coherency. In this work we investigate to which extent a machine can discriminate real from machine generated text. This is important in itself for automatic detection of computer generated stories, but can also serve as a tool for further improving text generation. We show that learning a dedicated scoring function to discriminate between real and fake text achieves higher precision than employing the likelihood of a generative model. The scoring functions generalize to other generators than those used for training as long as these generators have comparable model complexity and are trained on similar datasets.

1 Introduction

Natural language generation is a key component of dialogue, translation, summarization, captioning, etc. Text generation has been considered a daunting task for decades because of the symbolic and combinatorial nature of languages, and the long range dependencies characterizing natural text. Recent advances [Radford et al., 2019] have however shown that large models trained on large datasets can produce remarkably coherent and fluent text.

The main question we investigate in this work is whether we can automatically discriminate between machine generated and human generated text. This is important for two reasons. First, it has an obvious practical application in the detection of spam and, more generally, machine generated stories. Second, it is a stepping stone towards building better text generation models, as these discriminators could be used to improve the original models used to generate text.

The dominant approach to text modeling today is based on auto-regressive models [Elman, 1990, Graves, 2013, Sutskever et al., 2014], which factorize the joint distribution of words into the product of conditional distributions using the left to right order. Despite words being generated sequentially one token (word or sub-word) at the time from left to right, sequences are then scored holistically using heuristics like beam search, which alleviate the greediness of the generation procedure. Importantly, sequence scoring is performed using the *same* model used to generate text in the first place.

In this work, we first investigate whether decoupling generation and scoring yields better accuracy for the task of predicting whether a sequence is machine or human generated. Our experiments clearly indicate that learning a separate predictor is a better choice.

Second, we perform an extensive empirical study to assess how well such discriminator generalizes to generators not used at training time. On the bright side, our experiments show that the discriminator is indeed able to generalize to similar generators and even weaker generators trained on similar datasets. However, generalization deteriorates once the generators used at test time are stronger, or when the generators used at test time are trained on out-of-domain datasets.

[*]Equal contribution

Preprint. Under review.

2 Related Work

While there is ongoing effort towards generating text without specifying a word ordering [Kaiser et al., 2018, Stern et al., 2018, 2019, Gu et al., 2019, Ghazvininejad et al., 2019], the dominant approach to text generation is still based on auto-regressive models which generate one word at the time from left to right. The basic component of these models is a predictor of the next word in the sequence conditioned on the previously generated words. Such conditional distribution can be represented by a count-based ngram model or by parametric functions such as a recurrent neural network [Graves, 2013, Bahdanau et al., 2014], a convolutioinal neural network [Gehring et al., 2017], and more recently, attention-based models like the transformer [Vaswani et al., 2017]. Recently, Radford et al. [2019] demonstrated remarkable generations in terms of fluency and coherency by using a large transformer language model trained on a large dataset.

With few exceptions [Zhang et al., 2017], generation using auto-regressive models employs beam search, a heuristic to approximately select the most likely sequence according to the joint distribution defined by the generator. Importantly, such sequence level scoring aggregates intermediate log-likelihood scores produced by the conditional prerdictor. In other words, the same scoring model is used for generating candidates and for selecting the best candidate in the set. In this work instead, we decouple the two scoring functions, and train a dedicated scoring function for the latter task.

Learning scoring functions has a long history and is generally referred to as energy-based modeling [LeCun et al., 2006]. In energy-based models (EBMs), the goal of learning is to shape an energy (or scoring) function such that training data points have lower energy than spurious (e.g., noisy) data points. Inference (e.g., a completion task like predicting the next word in the sequence) reduces to finding mimina in the energy landscape. EBMs have been mostly applied to vision applications [Teh et al., 2003, Du and Mordatch, 2019] where search is easier thanks to the continuous nature of the input space. At training time, the energy is decreased at training data points (positive examples) and increased at spurious points (negative examples) which can be found via MCMC sampling [Hinton, 2002], gradient descent [LeCun et al., 2006], etc. Our work can be intepreted as a particular instance of EBM where negatives are produced by a pre-trained language model as opposed to the energy function itself. Learning a generator and a discriminator relates also to Generative Adversarial Networks [Goodfellow et al., 2014], except that in our case the generator is trained beforehand.

Using a separately trained scoring function to evaluate candidates in the beam has been recently proposed in the context of dialogue modeling by Kulikov et al. [2018]. There the negatives are randomly chosen next utterances from the training dataset. Here we use samples from a language model to build the set of negatives.

GLTR [Strobelt et al., 2019] demonstrates an interactive tool to show how a large language model scores an input text, which in particular can be used to check if text was generated by a language model generator. While we share the same motivation, this work extends that study by showing that a dedicated discriminator can work better than original generator's log-likelihood score. Moreover, we provide an empirical assessment of the generalization ability of such discriminator when varying the type of generator and the datasets used for training it, as often times training of the discriminator is performed without access to the actual generator used at test time.

Finally and concurrent with this work, there has been a release of a training dataset of the GPT-2 language model generations [2], for the purpose of training discriminators capable of detecting machine generated text. While we share the same motivation, our work is a much broader investigation on the topic. We assess generalization of several discriminator architectures to not just one but several kinds of generators and datasets used for training.

3 Approach

In this section we first describe the architectures of the discriminator we considered, we then explain how we generate negatives and introduce the loss function used to train the discriminator. We conclude with a description of the testing protocol and metrics.

[2] `https://github.com/openai/gpt-2-output-dataset`

3.1 Learning to Score

Our goal is to build a function f_θ that scores the *joint compatibility* of an input sequence of tokens $(w_1, \ldots, w_n)$. The goal of training is to score golden sequences higher than other sequences. We parameterize the scoring function as a neural network, using the architectures described in §4.4.

At training time, the scoring function f_θ is trained using a variant of the ranking loss [Collobert et al., 2011]. Let x^+ be a positive sample consisting of a sequence of n tokens taken from the training set, $(x_1^-, ..., x_k^-)$ be a set of k "negatives" each derived from the same positive sequence but containing at least some machine generated tokens (see more details below), the standard ranking loss is:

$$\mathcal{L} = \sum_{i=1}^{k} \max\left(0, 1 - f_\theta(x^+) + f_\theta(x_i^-)\right). \tag{1}$$

In our work, we slightly modify the above loss by a) replacing the full sum with the term yielding the largest loss (corresponding to the most offending negative [LeCun et al., 2006]), and b) splitting the loss in three parts. The first loss is the same as above, dubbed $\mathcal{L}_{\text{all}}$. The second loss is the same as above but specialized in suffixes, as it is used *only* when the generated text appears at the *end* of the sequence; this is dubbed $\mathcal{L}_{\text{suff}}$. The third loss is the same as above but specializes in prefixes, as it is used *only* when the generated text appears at the *beginning* of the sequence; this is dubbed $\mathcal{L}_{\text{pref}}$. We therefore have three scoring functions, $(f_{\text{all}}, f_{\text{suff}}, f_{\text{pref}})$, one for each such loss, which are computed using three separate top linear layers. Empirically, we found that using these three ranking losses worked better than a single one (see ablation study in §4.8).

3.1.1 Generating Negatives

The most critical component of training an energy based model is the method used to generate *negatives*, i.e. inputs where the energy should score high or the score should be low (unlikely inputs). In settings with continuous variables, researchers have suggested MCMC [Teh et al., 2003] or Langevin dynamics [Du and Mordatch, 2019]. In this work instead, we use the fact that modern auto-regressive models for text are already quite good, and we use them for negative sampling.

We train two auto-regressive language models, a left-to-right one which will be used to produce suffixes, and a right-to-left one which will be used to generate prefixes. The negatives are generated by top-k sampling [Fan et al., 2018] setting k equal to 10. Given a trained language model (for instance, a left-to-right autoregressive model) and given a positive example

$$x^+ = (w_1, \ldots, w_n), \tag{2}$$

a negative can be written as:

$$x^- = (w_1, \ldots, w_i, \hat{w}_{i+1}, \ldots, \hat{w}_n), \tag{3}$$

where w_j for $j \in [1, i]$ with $i < n$ are ground truth words belonging to the common context and $\hat{w}_j$ for $j \in [i+1, n]$ are words generated by the language model. In the same way, we can sample a negative with a right-to-left model yielding:

$$x^- = (\hat{w}_1, \ldots, \hat{w}_{n-i}, w_{n-i+1}, \ldots, w_n). \tag{4}$$

4 Experiments

In this section we first describe the datasets and preprocessing used, and then provide the architecture details for both generators and scoring functions. We then present the main results of this work and extensively investigate the generalization ability of the scoring functions we have considered.

4.1 Scoring Model Evaluation

We evaluate the success of a scoring model in three settings: in-domain, cross-architecture, and cross-corpus. These settings are determined by the corpora used to train the training generator(s) and the corpora used to train the testing generators; and the train and test generator architectures. In each of these settings, given a testing corpus C_{test} and a testing generative model G_{test}, we start by taking a

number of examples from C_{test} as in (2) and for each, generate k negatives with G_{test} as in (3) or (4). In all the experiments we use $k = 10$.

The three settings are distinguished by the choices of C_{test} and architecture of G_{test} compared to C_{train} and G_{train} used when training the scorer. Notice that C_{train} is used for training first the training generators, and then the scoring function. In the **in-domain** setting, C_{test} is C_{train} (except the prefixes at test are from the test-set of the corpus), and G_{test} has the same architecture as G_{train} (but is trained from a different random seed). In the **cross-architecture** setting, again C_{test} is C_{train}, but the architecture of G_{test} is different from the architecture of G_{train}. In the **cross-corpus** setting, G_{test} has the same architecture as G_{train} but C_{test} is different than C_{train}, and G_{test} is trained on the training split of C_{test}, while G_{train} trained on the train split of C_{train}.

In each case, the scoring function f_θ is applied to the positive and the k negative examples, and we measure performance in terms of *precision at 1* (P@1), which is the ratio between the number of times the ground truth sequence scores the highest over the number of sequences in the test set.

4.2 Corpora

We train models on three corpora whose statistics are reported in Appendix Tab. 7:

Books: The Toronto books corpus described in Zhu et al. [2015], Kiros et al. [2015], which consists of fiction books in 16 different genres, totaling about half a billion words.

CCNews: We collect a de-duplicated subset of the English portion of the CommonCrawl news dataset [Nagel, 2016], which totals around 16 Billion words.

Wikitext: The wikitext103 dataset from Merity et al. [2016], which consists of 103 million words from English Wikipedia articles.

While Wikitext and CCNews are factual, Books is fiction and comprises a wide variety of writing styles. The CCNews corpus has the narrowest domain and it is two orders of magnitude larger than Wikipedia. Overall, these datasets are interesting because they enable us to assess the ability of the scoring function to fit and generalize across various axes, from the amount of data available at training time to the richness of style and relatedness among the different data sources.

On Wikitetxt and Books, we extract positive sequences from windows of text that are 160 tokens long with a stride of 40. On the larger CCNews we do the same except that we stride by 160 tokens. This protocol to mine positives is used both at training and test time, although at test time we limit the evaluation to 60,000 randomly chosen positive samples.

We use a Byte Pair Encoding [Sennrich et al., 2015] in order to represent all the dataset with a common vocabulary. In particular, our vocabulary contains 50k tokens that was constructed from a byte level UTF-8 encoding of CC-NEWS corpus following Radford et al. [2019].

4.3 Generator Architectures

We use two different architectures for generating negatives: a fully convolutional network (Conv) [Dauphin et al., 2017] and a transformer based network (Transf) [Vaswani et al., 2017]. We have a medium and a large transformer model, yielding three language models (for each direction) in total: Conv, Transf, TransfBig. The convolutional model uses the "GCNN-14" architecture from Dauphin et al. [2017] with 12 convolutional layers, as implemented in "fconv_lm_dauphin_wikitext103" by Ott et al. [2019]. The transformer models are based on the architecture used in Baevski and Auli [2019]. The medium sized models uses 6 blocks each containing a multi-head attention module with 8 heads. The large models use 12 blocks each containing a multi-head attention module with 16 heads. The transformer models are also implemented in Ott et al. [2019] as "transformer_lm" and "transformer_lm_big". We also train a "huge" model on CCNews with about 10 times the parameters of the "transformer_lm_big". See Table 8 in Appendix for more details.

As described in Sec. 3.1.1, we use these language models to generate either a prefix or a suffix. Unless otherwise specified, positive sequences are 160 tokens long. With equal probability, we condition on either 120 or 140 tokens taken from either the beginning or the end of the original sequence when generating negatives.

	Books	CCNews	Wiki
Linear	56.9	47.7	34.6
BiLSTM	84.5	67.3	60.4
BiLSTM Big	86.8	70.2	62.4
BiTransf	89.7	79.6	69.7
UniTransf	**93.6**	**86.7**	**75.0**
TransfBig (language model log-likelihood)	1.1	2.2	3.2
TransfBig (language model negative log-likelihood)	68.2	62.2	53.3

Table 1: "In domain" generalization of scoring models (each row) on various text datasets. A column corresponds to the corpus used to get positives and to fit the train and test language models, which are TransfBig (§4.3) with different initial seeds. The score in each cell is the P@1 with 10 negatives. The final two rows are the training language model used as a scorer by taking either its logprob or negative logprob of the whole text (model generation are generally more likely than real text [Strobelt et al., 2019]).

4.4 Scoring Function Architectures

We consider three architectures for the scoring function:
Linear which computes a score via a bag of tokens: $f(w_1, ..., w_n) = (\sum_{i=1}^{n} u_{w_i})$, where u_i is a learnt scalar parameter corresponding to the i-th token in the vocabulary.
BiLSTM [Schuster and Kuldip, 1997, Graves and Schmidhuber, 2005] which computes a score through L bidirectional layers using LSTM recurrent units [Hochreiter and Schmidhuber, 1997], as in Linear(AvgPool$(h_{L,1}, \ldots, h_{L,n})$), where $h_{L,i}$ is the hidden state at position i and layer L which is the concatenation of the forward and backward hidden states, AvgPool averages hidden states over positions and Linear is a vector of parameters projecting the hidden state down to a scalar value. We consider two versions, referred to as BiLSTM and BiLSTM Big. Both have 4 layers, but BiLSTM has 512 units in both the embedding layer and the hidden layers, while BiLSTM Big has 758 units in the embedding layer and 2014 units in the hidden states.
Transformer [Vaswani et al., 2017, Devlin et al., 2018] which computes a score similarly to the BiLSTM's scorer, except that each bi-LSTM layer is replaced by a either a bidirectional Transformer layer (BiTransf), or a Transformer with causal self-attention (UniTransf). For unidirectional models we use the same averaging technique as with BiLSTM models. For bidirectional models the score is computed via: $f(w_1, ..., w_n) = u^\top h_{L,1} + b$, where $h_{L,1}$ is the top layer hidden state at the first position (as common practice also in prior work [Devlin et al., 2018]). BiTransf uses the BERT-Base [Devlin et al., 2018] configuration: 12 bidirectional transformer layers with 768 units and 12-head attention. It is initialized from a publicly available pretrained *bert-base-cased* model[3]. UniTransf has instead 12 layers with 1024 units and 16 attention heads per layer. Following Radford et al. [2019] we used the language modeling task for pretraining.

For all models, we use Adam [Kingma and Ba, 2014] optimizer with warmup. Training is stopped after processing 2.5M samples without any improvement on the validation set. We use data-parallel synchronous multi-GPU training with up to 8 nodes, each with 8 Nvidia V100 GPUs. To improve training speed, we use mixed precision training[4]. Following common practice we clip the norm of the gradient vector [Pascanu et al., 2013]. More details about hyper-parameter setting can be found in Appendix Tab. 10, while Tab. 9 in Appendix reports the number of parameters of each scoring function.

4.5 In-domain generalization

In Table 1 we report the results of the in-domain generalization experiment using our large language model, TransfBig. We observe that when the scoring models have similar representational power compared with the generator (UniTransf, see Table 9), they are able to distinguish real from fake completions fairly accurately, reaching a p@1 with 10 negatives of more than 90% on the Books dataset (which is easier since it exhibits the larger variety of style and topics), and attaining above 80%

[3] `https://github.com/huggingface/pytorch-pretrained-BERT`
[4] `https://github.com/NVIDIA/apex`

	Conv	Transf	TransfBig
Conv	**97.8**	75.1	57.0
Transf	84.4	**88.7**	70.2
TransfBig	79.6	86.8	**75.0**

Table 2: Cross-architecture generalization in terms of P@1 with 10 negatives using the Wikitext dataset. Each row is a model architecture used for generating the training negatives, and each column is a model architecture for generating the testing negatives.

on the more challenging CCNews dataset (for which generation is easier and hence discrimination harder). Weaker scoring models are able to do comparably or better at discriminating real from fake than the training generator used as a scorer by taking the negative log probability of the sequence as a score. In conclusion, learning a dedicated discriminator is much more effective than using the original language model generator as a scorer, and discriminators whose complexities match the generators work better.

4.6 Cross-architecture generalization

In Table 2, we assess how well the best scoring function we have, UniTransf, generalizes to different generator architectures at test time, using Wikitext as dataset. As a reference, the test perplexity of Conv, Transf and TransfBig is 35.4, 33.5 and 24.5, respectively. Therefore, TransfBig is expected to produce higher quality negatives.

Unsurprisingly, each scoring model does best at discriminating the generator architecture it was trained on (see results along the diagonal). Perhaps surprisingly, there is not a clear relationship between train-generative-model perplexity and generalization. On one hand, we see that the mean result of training with generations from the big transformer (averaging values in last row) is slightly worse than the mean score of training on the medium transformer (averaging values in the row before the last). Thus training with only strong negatives can make it easier to be exploited by weaker negatives. On the other hand, scoring functions do exhibit good generalization to negatives produced by weaker generators at test time. For instance, if negatives are generated by the strongest generator, TransfBig, at training time but then the scoring function is tested using negatives from Transf, P@1 is more than 10% higher (last row of the table). On the contrary, harder negatives at test time are not discriminated well (see the decrease of P@1 along the first row).

This is further confirmed by the experiments in Tab. 4, where we compare on CCNews testing with negatives produced by TransfBig (which matches the training architecture) versus TransfHuge which is an almost ten times bigger generator. In this case, P@1 decreases by 35%.

Overall, the scoring function exhibits good generalization as long as negatives are produced by weaker generators. Otherwise, P@1 drops significantly although not catastrophically.

4.7 Cross-Corpus generalization

In table 3 we show the results of generalizing across corpora using UniTransf as scoring function and TransfBig as generator both at training and test time. We can see that the models generalize less well across corpora than they did across architectures; for instance, when testing on Wikitext a scoring function trained with either Books or CCNews, P@1 does not even reach 40%. However, training on the union of two of the corpora gives a large benefit over training on just one or the other when testing on the third. Finally, training on the union of *all* the three corpora (last row) yields a scoring function that is fairly robust to the testing conditions, with just a slight decrease of precision compared to the ideal case of exact match betwteen corpus used at training and test time.

We also used the publically available GPT2 medium [Radford et al., 2019] and the TransfHuge model we trained on CCNews to generate test negatives. In Table 4, the first row is "in-domain" (and so matches the corresponding row in Table 1). The next row is cross-architecture in the middle cell; and cross-corpus and cross architecture in the other cells since TransfHuge is trained on CCNews. The final row is cross-corpus and cross architecture since GPT2 was trained another dataset which we do not have access to, and it also has a different architecture.

	Books	CCNews	Wiki
Wiki	51.2	54.9	**75.0**
Books	**93.8**	41.7	26.2
Books + Wiki	93.0	60.0	69.7
CCNews	47.5	**86.7**	36.3
CCNews + Wiki	63.5	**86.7**	74.0
Books + CCNews	90.9	86.3	42.6
ALL	91.0	86.6	74.1

Table 3: Cross-corpora generalization using TransfBig generator and UniTransf scoring function. Each row specifies the corpora used at training time. Each column shows the corpus used at test time. The score in each cell is P@1 with 10 negatives.

	Books	CCNews	Wiki
TransfBig	**93.6**	**86.7**	**75.0**
TransfHuge	47.3	51.4	40.2
GPT2 Radford et al. [2019]	79.9	44.2	53.4

Table 4: Training negatives are produced by TransfBig trained on the datasets shown in the columns. Rows correspond to the generator architectures used to produce negatives at test time. TransfHuge is trained on the training portion of CCNews. TransfHuge has similar architecture as TtransfBig but much bigger, with a total of 1.4 billion parameters. In the last row we report P@1 using negatives produced by GPT2 medium, totalling 345 million parameters and trained on the web corpus described in Radford et al. [2019]. Columns correspond to the corpus used for test prefixes. The score in each cell is p@1 with 10 negatives.

4.8 Ablation Study

First, we investigate the robustness of the best scoring function, UniTransf, with respect to generation length and location of the generated text. Results in Table 5 show that the scoring function works better on easier tasks despite the fact that the distribution of negatives is different from the one seen at training time. For instance, if negatives are generated using a left-to-right language model producing the last 20 tokens at training time but at test time negatives have the last 40 tokens generated (a considerably easier discrimination task), P@1 increases from 53.8% to 55.5%. On the contrary, P@1 drops by 20% if at training time negatives have 40 generated tokens but only 20 at test time. Similarly, the scoring function does not generalize well to filling gaps in other positions than those used for training (see poor performance on cells on the anti-diagonal).

Finally, we study the impact of the variants of ranking loss in equation (1). Table 6 shows the baseline performance when using the holistic term $\mathcal{L}_{\text{all}}$ as opposed to also the losses specific to detecting machine generated text in the prefix and suffix (first two rows). The composite loss yields almost a 1% improvement in precision. While computing the gradient using more negatives only marginally helps, selecting the most offending negative in that set is the strategy that offers the biggest gain, more than 2% when averaging over 3 negatives as opposed to taking the one that increases the loss the most. In our work, we use this latter setting.

4.9 Stability to Other Negative Distributions

In the previous sections we have seen that the scoring functions are not robust to negatives generated from a model trained on a different corpus. However, even in that case, a negative is still a sample from an auto-regressive neural network. In Section E in the Appendix, we show examples where changing a few entities can cause large jumps in score (from negative to positive), and so fool the scorer. More generally, we see that the scoring function is not robust to truly out-of-domain samples. For example, the scorer will score blocks of randomly generated text higher than real text.

These behaviors are evidence that the scoring functions have learned the regularities of *generated* text, as opposed to learning the regularities of real text. We surmise that it does so because modeling the latter would be much more difficult than the former. By modeling generated text, the scoring

		left to right		right to left	
		40 tokens	20 tokens	40 tokens	20 tokens
left to right	40 tokens	69.7	49.2	20.5	17.6
	20 tokens	55.5	53.8	14.5	13.6
right to left	40 tokens	19.2	16.8	83.1	61.9
	20 tokens	17.1	15.6	70.9	67.1

Table 5: Generaralization across different generation tasks using UniTransf scoring function and TransfBig generator on Wikitext. Each row specifies the datasets of negatives used at training time. Each column specifies the dataset of negatives used at test time. We vary the length of the generation (either 20 or 40 tokens) and the position of the generated text (either at the beginning or the end of the sequence). The score in each cell is P@1 with 10 negatives.

	P@1
$\mathcal{L}_{\text{all}}$	66.8
$\mathcal{L}_{\text{all}}$+$\mathcal{L}_{\text{suff}}$+$\mathcal{L}_{\text{pref}}$	67.6
$\mathcal{L}_{\text{all}}$+$\mathcal{L}_{\text{suff}}$+$\mathcal{L}_{\text{pref}}$, 3 random negatives	68.0
$\mathcal{L}_{\text{all}}$+$\mathcal{L}_{\text{suff}}$+$\mathcal{L}_{\text{pref}}$, worst negative out of 3	70.3
$\mathcal{L}_{\text{all}}$+$\mathcal{L}_{\text{suff}}$+$\mathcal{L}_{\text{pref}}$, worst negative out of 6	70.6

Table 6: Ablation study investigating different variants of the ranking loss introduced in §3.1 using TransfBig generator and BiTransf scoring function on Wikitext.

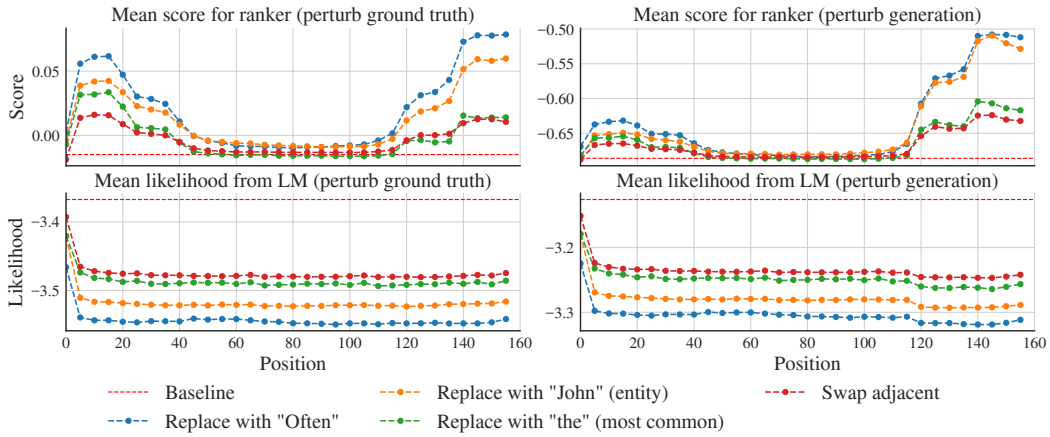

Figure 1: Effect of applying various perturbations (word replacement and swap of adjecent words) to groundtruth and generated sequences at different positions in terms of discriminator score and generator log-likelihood (averaged over the whole test set of Wikitext). The score is only affected by corruptions at either end of the sequence. These out-of-domain corruptions invariably increase the score. However, all perturbations lower the log-likelohood of the sequence.

function assigns high score to anything that is not generated by its training generator. While not surprising, this might be considered a liability of such scoring functions. However, as a model of text, the scoring functions should be considered as working on the *residuals* of the language models used to generate negatives. For the examples in Appendix Section E, the language model records a large *decrease* in likelihood after the change in entity; and the language models of course give much lower likelihood to random text than gold or generated text. Therefore, the scoring function needs not to be accurate on examples that are already very unlikely according to these language models.

In Figure 1 we show the average effects of applying various perturbations to sequences from Wikitext103 on an in-domain scorer and a language model to each location (from 1 to 160) in the sequence. We see that for all perturbations, the scorer increases its score, but the language model decreases its

score (the likelihood). We also see that the scorer is more sensitive to the ends of the text, which is where the negatives were different from real text at training time.

5 Final Remarks

Through an extensive empirical analysis, we have investigated how accurately machine generated text can be detected automatically. Our experiments show that training a dedicated scoring function for this task works significantly better than using the (negative) log-likelihood scores of the language model generating the data under consideration. We then assessed generalization performance of such scoring function, and found that it generalizes well as long as language model generators used at test time are trained on similar datasets and use architectures that are similar or weaker than those used to train the scoring function.

While the lack of broader generalization across corpora limits the applicabilty to downstream tasks where there is no control on the data and models used for training the generators, we believe that results could further improve by scaling up the size of the scoring function and by adding a larger variety of negatives, including sequences generated adversarially by the same scoring function.

References

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByxZX20qFQ`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12 (Aug):2493–2537, 2011.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *CoRR*, abs/1903.08689, 2019. URL `http://arxiv.org/abs/1903.08689`.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Association for Computational Linguistics*, 2018.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*, 2017.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Constant-time machine translation with conditional masked language models. *CoRR*, abs/1904.09324, 2019. URL `http://arxiv.org/abs/1904.09324`.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5–6):602—-610, 2005.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *CoRR*, abs/1902.01370, 2019. URL `http://arxiv.org/abs/1902.01370`.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735—1780, 1997.

Lukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. *arXiv:1803.03382v6*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

Ilya Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. Importance of a search strategy in neural dialogue modelling. *arXiv preprint arXiv:1811.00907*, 2018.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. *Predicting Structured Outputs*, 2006. MIT Press.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Sebastian Nagel. Cc-news. `http://web.archive.org/save/http://commoncrawl.org/2016/10/news-dataset-available/`, 2016.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. `https://openai.com/blog/better-language-models`, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. `https://openai.com/blog/language-unsupervised/`, 2018.

Mike Schuster and K. Paliwal Kuldip. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *NeurIPS*, 2018.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. *CoRR*, abs/1902.03249, 2019. URL http://arxiv.org/abs/1902.03249.

Hendrik Strobelt, Sebastian Gehrmann, and Alexander Rush. Glitr. http://web.archive.org/web/20190507175455/http://gltr.io/dist/, 2019.

I Sutskever, O Vinyals, and QV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 2014.

Y. W. Teh, M. Welling, S. Osindero, and Hinton G. E. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 2001.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. Improving neural machine translation through phrase-based forced decoding. *CoRR*, abs/1711.00309, 2017. URL http://arxiv.org/abs/1711.00309.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

A Corpora sizes

Dataset	Train	Valid	Test
Books	690	7.3	8.0
CCNews	21718	1.0	3.4
Wikitext	113	0.2	0.3

Table 7: Number of BPE tokens in millions for each dataset.

B Model Sizes

	Generators				
	Conv	Transf	TransfBig	TransHuge	GPT2 med
embed.	13	26	51	77	51
others	164	19	151	1360	329
total	176	45	203	1437	380

Table 8: Number of parameters (in millions) for the generator language models. The computational cost is directly related to the number of parameters in other layers than the input embedding layer (second row).

	Scoring Functions				
	Linear	BiLSTM	BiLSTM Big	BiTransf	UniTransf
embed.	0.1	26	39	38	51
others	0	23	90	86	151
total	0.1	49	129	125	203

Table 9: Number of parameters (in millions) for the scoring functions (right). The computational cost is directly related to the number of parameters in other layers than the input embedding layer (second row).

C Hyper-parameter Setting

All models are implemented usign the PyTorch framework [Paszke et al., 2017] and are optimized using Adam [Kingma and Ba, 2014].

To train our biggest model (UniTransf) on our biggest dataset (CC-NEWS) we used 8 machines each with 8 GPUs in synchronous mode using data parallelism. The resulting large batch size speeds up training when combined with float16 reduced precision and cosine scheduling of the learning rate without any restarts [Loshchilov and Hutter, 2016], i.e. we decay the learning rate to zero over the course of "max steps" updates and then stop training. Using these methods, we reduced training time by five times compared to a single node training. For all other configurations we used a single node with up to 8 GPUs and inverse square root decay.

	max lr	bsz (per GPU)	GPUs	fp16	warmup steps	max steps
Linear	0.01	1024	1	+	1000	-
BiLSM	0.0002	128	8	+	1000	-
BiTransf	0.0001	64	8	+	1000	-
UniTransf (CC-NEWS)	0.0003	32	64	+	2000	180000
UniTransf (rest)	0.0003	32	8	-	2000	-

Table 10: Hyper-parameter values used in our scoring functions.

D Score Distributions

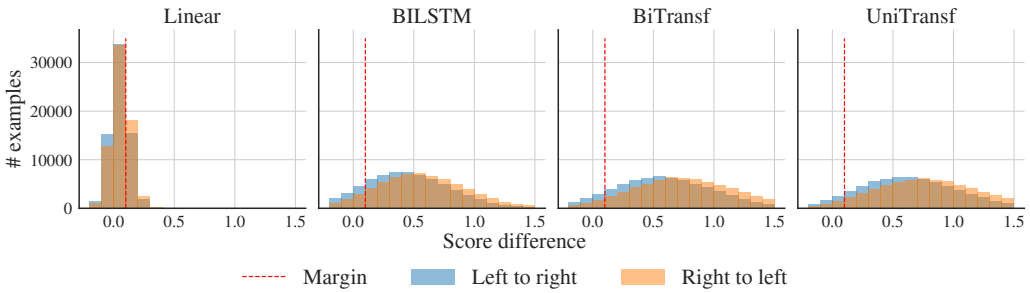

Figure 2: Distributions of score differences between pairs of ground truth completions and generated ones for different scoring models. We show results for two generations (left to right and right to left) from Wikitext dataset. In both cases we generate 40 tokens. Examples on the right of the red line ($margin = 0.1$) have zero loss.

E Perturbing the Energy Function

In this section we show that we can change a few words to make a negative example become a "positive" one as judged by the scoring function, and vice versa, by using gradient information.

Below here, we show an example of ground truth sentence from the Wikitext dataset.

<EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[Derek Jacobi. He was cast in the 2005 theatre productions of the Philip Ridley play Mercury Fur, which was performed at the Drum Theatre in Plymouth and the Menier Chocolate Factory in London. He was]

Here the block has 160 BPE tokens, where the first 120 tokens (black font) are used as context and the remaining 40 are the ground truth completion. Next, we use a language model to generate 10 negatives:

Negative 1 <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[Chris Elliott in 2006 as the character. Boulter has appeared in various television specials dealing with the series since its inception. <EOS> After graduating with a degree in drama, Boulter worked as a]

Negative 2 <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[Stephen Fry in the episode "You're All Alone" and in the episode "The Longest Day". <EOS> He auditioned for the role in the series in 2003 but was not cast. In 2005]

⋮

Negative 10 <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[Ian Somerhalder on the BBC series Top Gear; this was followed up in 2007 by a role in the BBC science-fiction series Doctor Who. In 2008 Boulter appeared in the BBC]

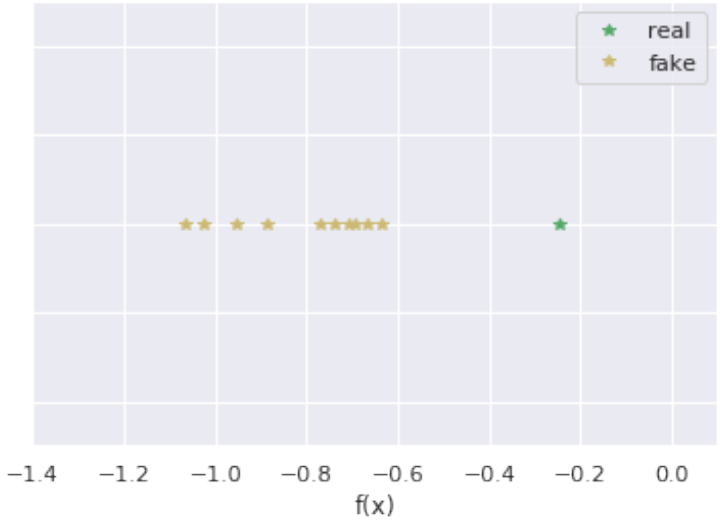

Figure 3: Real and fake (negatives generated from TransfBig language model) completions as scored by the learned scoring function. The scoring function is able to separate them well. These scores are calcuated based on the single example reported in the main text of §E.

On this example, using the big transformer model, UniTransf, as the scoring function, we are able to separate real from fake examples as shown (Figure 3). We want to perturb these negatives to violate the margin. To do so, we make use of the gradient information from the scoring function $\nabla_x f_\theta(x)$ and use a first order Taylor expansion to approximate the effect of a token replacement (we abuse our notations and use x to denote embeddings in this analysis). Given the original sample x, we change one word x_i to x'_i to arrive at x'. The score of x' is approximately:

$$f_\theta(x) + \nabla_{x_i} f_\theta(x) \cdot (x'_i - x_i)$$

Using this approximation, we can search for those token replacements that increase/decrease the score the most. We can easily change a negative sample to a positive one by replacing the 5 words highlighted below. In paratheses, we report both score and language model perplexity.

Original negative (score -0.77, PPL 20.77) <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors

Mark Strong and[Chris][Elliott] in 2006 as the character. Boulter has appeared in various television specials[dealing] with the series since its inception. <EOS> After graduating with a degree in[drama], Boulter worked as a

Perturbed negative (score 0.00, PPL 117.30) <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[Gor]$_{(-0.0.64,\ 28.97)}$[Trem]$_{(-0.56,\ 38.86)}$ in 2006 as the character. Boulter has appeared in various television specials[relates]$_{(-0.77,\ 24.60)}$ with the series since its inception. <EOS> After[Health]$_{(-0.35,\ 39.52)}$ with a degree in[edited]$_{(-0.49,\ 27.45)}$, Boulter worked as a

In the above example, we also show the (score, PPL) for replacing a single token in the subscripts. Similarly, we can replace a few words and make a positive sample become negative.

Original positive (score -0.25, PPL 77.68) <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[Derek] Jacobi. He was cast in the 2005 theatre productions of the Philip Ridley play Mercury Fur, which was performed at the[Drum] Theatre in[Plymouth] and the[Men]ier[Chocolate] Factory in London. He was

Perturbed positive (score -0.78, PPL 142.85) <EOS> =Robert Boulter= <EOS> <EOS> Robert Boulter is an English film, television and theatre actor. He had a guest-starring role on the television series The Bill in 2000. This was followed by a starring role in the play Herons written by Simon Stephens, which was performed in 2001 at the Royal Court Theatre. He had a guest role in the television series Judge John Deed in 2002. In 2004 Boulter landed a role as "Craig" in the episode "Teddy's Story" of the television series The Long Firm; he starred alongside actors Mark Strong and[connected]$_{(-0.30,\ 118.30)}$ Jacobi. He was cast in the 2005 theatre productions of the Philip Ridley play Mercury Fur, which was performed at the[C]$_{(-0.28,\ 75.36)}$ Theatre in[London]$_{(-0.47,\ 62.29)}$ and the[Vaughan]$_{(-0.40,\ 93.77)}$ier[cerning]$_{(-0.32,\ 100.71)}$ Factory in London. He was

As shown in Figure 4, we can easily "fool" the discriminator by editing a few words. However, these edited sentences have a very low probability (high PPL) under the generator we used. This explains why the discriminator gets fooled, because it has never seen such negatives during training.

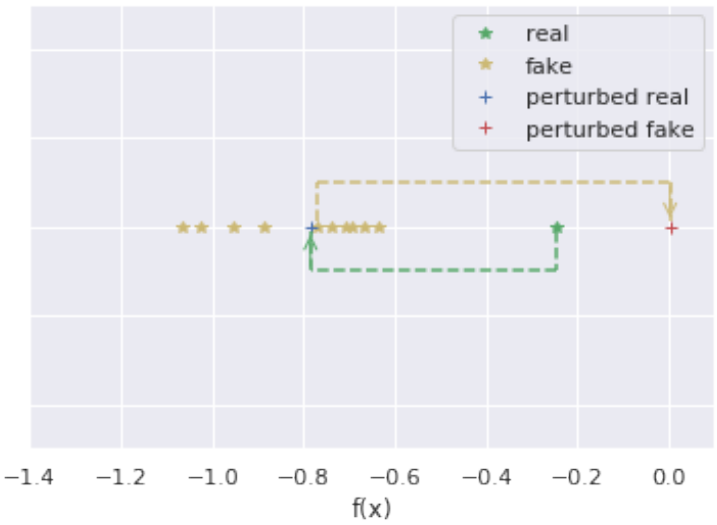

Figure 4: By changing a few words we can make a negative sample become real as scored by the scoring function, and vice versa.