Residual EBMs: Does Real vs. Fake Text Discrimination Generalize?

Anonymous authors
Paper under double-blind review

Abstract

Energy-based models (EBMs), a.k.a. un-normalized models, have had recent successes in continuous spaces. However, they have not been successfully applied to model text sequences. While decreasing the energy at training samples is straightforward, mining (negative) samples where the energy should be increased is difficult. In part, this is because standard gradient-based methods are not readily applicable when the input is high-dimensional and discrete. Here, we side-step this issue by generating negatives using pre-trained auto-regressive language models. The EBM then works in the *residual* of the language model; and is trained to discriminate real text from text generated by the auto-regressive models.

We investigate the *generalization ability* of residual EBMs, a pre-requisite for using them in other applications. We extensively analyze generalization for the task of classifying whether an input is *machine or human* generated, a natural task given the training loss and how we mine negatives. Overall, we observe that EBMs can generalize remarkably well to changes in the architecture of the generators producing negatives. However, EBMs exhibit more sensitivity to the training set used by such generators.

1 Introduction

Energy-based models (EBMs) have a long history in machine learning (Hopfield, 1982; Hinton, 2002; LeCun et al., 2006). Their appeal stems from the minimal assumptions they make about the generative process of the data. Unlike directed or auto-regressive models which are defined in terms of a sequence of conditional distributions, EBMs are defined in terms of a single scalar energy function, representing the joint compatibility between all input variables. EBMs are a strict generalization of probability models, as the energy function need not be normalized or even have convergent integral.

Training an EBM consists of decreasing the energy function at the observed training data points (a.k.a. positives), while increasing it at other data points (a.k.a. negatives) (LeCun et al., 2006). Different learning strategies mainly differ in how negatives are mined (Ranzato et al., 2007). Some find negatives by gradient descent, or using Monte Carlo methods like Gibbs sampling (Welling et al., 2005) and hybrid Monte Carlo (Teh et al., 2003), which enable the loss to approximate maximum likelihood training (Hinton, 2002). Other approaches instead use implicit negatives, by enforcing global constraints on the energy function, like sparsity of the internal representation (Ranzato et al., 2007), for instance. GANs (Goodfellow et al., 2014) can be interpreted as a particular form of EBM where the negatives are generated by a learned model.

While there are works exploring the use of EBMs for modeling images (Teh et al., 2003; Ranzato et al., 2013; Du & Mordatch, 2019), they have not been successfully applied to text. One reason is that text consists of sequences of discrete variables, which makes the energy function not differentiable with respect to its inputs. Therefore, it is not possible to mine negatives using gradient-based methods. Other approaches to mine negatives are also not immediately applicable or may be too inefficient to work at scale.

In this work, we start from the observation that current large auto-regressive locally-normalized language models are already strong (Radford et al., 2019), and therefore, it may be beneficial to use them to constrain the search space of negatives. We propose to learn in the *residual space* of a pre-trained language model (LM), which we accomplish by using such LM to generate negatives for the EBM. Given a dataset of positives and pre-generated negatives, the EBM can be trained using either a binary cross-entropy loss or a ranking loss, to teach the model to assign a lower energy to true human generated text than to the text generated by the pre-trained LM.

The question we ask in this work is whether such an EBM can generalize well. Understanding this is important for two reason. First, this generalization is a prerequisite for using residual EBMs for modeling text. Second, in our setting, this generalization question is equivalent to the question of whether it is possible for a learned model (the energy function) to discriminate real text from text generated by an auto-regressive model. Discriminating real vs. machine-generated text is an important task on its own that has recently gained a lot of attention (Gehrmann et al., 2019; Radford et al., 2019; Zellers et al., 2019).

Our contribution is an extensive study of the generalization ability of such residual EBMs, or in other words, the generalization ability of models trained to detect real text from machine generated text. In particular, we assess how well the energy function is robust to changes in the architecture of the generator and to changes in the data used to train the generator. The overall finding is that the energy function is remarkably robust, and the bigger the model and the longer the generation the better its performance. Moreover, the energy function is robust to changes in the architecture of the LM producing negatives at test time. However, it is sensitive to the training dataset of the test generator.

2 Related Work

Our work can be interpreted as a particular instance of EBMs (LeCun et al., 2006) 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.

Since the discriminator is learned after the generator has been trained, it learns from the *residual error* of the generator, and therefore, our training procedure is a particular instance of a "cascade" model (Viola & Jones, 2001) and "boosting" (Freund & Schapire, 1997).

Using a separately trained scoring function to evaluate and rank candidate outputs has a long history which dates back to work on parsing and machine translation (Shen et al., 2004). In that work however, the goal was to improve a weak generator by employing a linear reranker taking as input relatively few hand-design features. The approach has been recently re-discovered in the context of dialogue modeling by Kulikov et al. (2018), but here negatives are randomly chosen next utterances from the training dataset.

Several recent works have studied whether machine generations can be detected automatically, but they do not study how these findings *generalize* to settings where generator architectures and corpora are different between training and test time. For example, Zellers et al. (2019) (GROVER) assume that the generator is known and apply only slight fine-tuning in order to train the energy function. Similarly, Gehrmann et al. (2019) (GLTR) assume knowledge of the generator; these Authors say "*We further hypothesize that these methods generalize to black-box scenarios, as long as the fake text follows a similar sampling assumption and is generated by a large language model*"; our work answers precisely this question, provides a rigorous experimental protocol and quantitative results.

Finally, there has been a release of a training dataset of the GPT-2 language model generations (Radford & Wu, 2019) 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 corpora used for training (including GPT-2).

3 Energy-Based Models for Text

In this section, we describe how we train the energy based model and how we mine negatives.

3.1 LEARNING

Our goal is to learn an energy function $E(w_1, \ldots, w_n|c; \theta) \in \mathbb{R}$ that scores the *joint compatibility* of an input sequence of tokens $(w_1, \ldots, w_n)$ given some context c and a set of parameters θ. The context depends on the application, it could be the preceding text, some keywords, a bag of words, a title, etc. In this work for simplicity, c is an affix from which we condition the generation.

The goal of training is to assign to golden sequences, i.e. sequences taken from a dataset of human generated text, lower energy than other sequences. We parameterize the energy function as a neural network, using the architectures described in §4.3.

At training time, the energy function can be trained using a variety of different losses. In this work, we consider two choices: the binary cross-entropy loss and the ranking loss (Collobert et al., 2011). As the findings are similar, unless otherwise specified we will refer to the binary cross-entropy loss, and report results with the ranking loss in Appendix C.

Let x^+ be a positive sample taken from the training set, and consisting of a *sequence* of n tokens given some context c. Let $(x_1^-, ..., x_k^-)$ be a set of k negative samples each derived from the same context c as above, all containing at least some machine generated tokens. We train our energy function using the (per-sample) binary cross-entropy loss:

$$\mathcal{L}_{\text{BCE}} = -\log(\sigma(-E(x^+|c; \theta))) + \log(\sigma(-E(\hat{x}^-|c; \theta))) \tag{1}$$

where $\hat{x}^-$ is the most offending negative (LeCun et al., 2006), i.e. its index is the solution of $\arg\min_{i=1}^{k} E(\hat{x}_i^-|c; \theta)$, and σ is the sigmoid function: $\sigma(u) = \frac{1}{1+\exp(u)}$.

3.2 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 (unlikely inputs). In settings with continuous variables, researchers have suggested MCMC (Teh et al., 2003) or Langevin dynamics (Du & 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 assuming the prefix is the context, and a right-to-left one which will be used to generate prefixes assuming the suffix is the context. 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 $x^+ = (w_{i+1}, \ldots, w_{i+n})$, for a given context: $c = (w_1, \ldots, w_i)$, a negative can be written as: $x^- = (\hat{w}_{i+1}, \ldots, \hat{w}_{i+n})$, where w_j for $j \in [1, i+n]$ are ground truth words, the first i of them belonging to the common context, and $\hat{w}_j$ for $j \in [i+1, i+n]$ are words generated by the language model conditioned on c. In the same way, we can sample a negative with a right-to-left model yielding $x^- = (\hat{w}_1, \ldots, \hat{w}_n)$, for a given context $c = (w_{n+1}, \ldots, w_{n+i})$.

4 EXPERIMENTAL SETUP

In this section we first describe the datasets and preprocessing used, provide architecture details for both generators and scoring functions, and finally introduce the evaluation settings.

4.1 CORPORA

We train models on three corpora coming from different domains. We report more detailed statistics about the sizes of these corpora in Appendix Table 8:

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 energy 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 Wikitext 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 the CC-News corpus following Radford et al. (2019).

4.2 Generator Architectures

We mainly use a transformer based network (Vaswani et al., 2017) to generate negatives. We have a medium, large and huge transformer model based on the architecture used in Baevski & Auli (2019), yielding three language models in total: TransfSmall, TransfBig and TransfHuge; see details also in Appendix B.

The small sized models use 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 huge models use 48 blocks each containing a multi-head attention module with 25 heads. Transformer models are also implemented in Ott et al. (2019) as "transformer_lm", "transformer_lm_big", and "transformer_lm_gpt2_big". The TransfHuge has 10x the number of parameters than TransfBig and it is trained on CCNews only. For each architecture except for TransfHuge we train two models on each each dataset: left to right and right to left.

In addition to the transformer generator, we also consider a 12-layer convolutional architecture (Conv) (Dauphin et al., 2017), and we also use a the third-party trained GPT2 models (Radford et al., 2019) as described in §5.3.

As described in §3.2, we use these language models to generate either a prefix or a suffix. Unless otherwise specified, the context is long either 120 or 140 tokens (with equal probability). Positive and negative examples have 40 or 20 tokens depending on the context size, for an overall length of 160 tokens in all cases. In preliminary experiments, we found that increasing the size of the generations and reducing the size of the context makes the learning task significantly easier. We analyze the effect of the context size in §5.5.

4.3 EBM Architectures

We consider three architectures for the energy function:

Linear which computes an energy value 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 & Kuldip, 1997; Graves & Schmidhuber, 2005) which computes an energy value through L bidirectional layers using LSTM recurrent units (Hochreiter & 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 "BiLSTMsmall" and "BiLSTMbig". Both have 4 layers, but BiLSTMsmall has 512 units in both the embedding layer and the hidden layers, while BiLSTMbig 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 an energy value similarly to the BiLSTM's, 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 energy 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-Large

	CORPUS: $C_{\text{train}} = C_{\text{test}}$	GENERATOR ARCHITECTURE: $A_{\text{train}} = A_{\text{test}}$
in-domain	✓	✓
cross-architecture	✓	✗
cross-corpus	✗	✓
unseen	✗	✗

Table 1: Four evaluation settings considered in this work, described in §4.4.

architecture (Devlin et al., 2018) initialized from Liu et al. (2019). It uses 24 self-attention layers with 1024 units and 16-head attention each. UniTransf has instead 12 layers with 1024 units and 16 attention heads per layer and it is initialized from a language modeling task as in Radford et al. (2019).

For all models, we use Adam (Kingma & 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[1]. 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 Table 11, while Table 10 in Appendix reports the number of parameters of each energy function.

4.4 EVALUATION

We evaluate the generalization of a residual EBM in four settings: in-domain, cross-architecture, cross-corpus, and unseen.

These settings are determined by the corpora C_{train} used to train the training generator G_{train} with architecture A_{train} and the corpora C_{test} used to train the testing generator G_{test} with architecture A_{test}. Note that $G_{\text{train}} \neq G_{\text{test}}$ even if $A_{\text{test}} = A_{\text{train}}$ as we use different training seeds. In all cases, C_{train} is used for fitting the G_{train} and also for the positives for the EBM.

In the **in-domain** setting, C_{test} is C_{train} (but any affixes used as conditioning during testing are from the test-set of the corpus), and $A_{\text{test}} = A_{\text{train}}$. In the **cross-architecture** setting, again C_{test} is C_{train}, but A_{test} is different from A_{train}. In the **cross-corpus** setting, $A_{\text{test}} = A_{\text{train}}$ 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 the **unseen** setting, both C_{test} is different than C_{train} and A_{test} is different from A_{train}.

In all settings, we report performance in terms of average classification accuracy balancing the positive and negative classes.

5 RESULTS

We now present the main results of this work and extensively investigate the generalization ability of the energy functions we have considered.

5.1 IN-DOMAIN GENERALIZATION

In Table 2 we report the results of the in-domain generalization experiment using our large language model, TransfBig. We observe that when the EBMs have similar representational power compared with the generator (UniTransf, see Table 10), they are able to distinguish real from fake completions fairly accurately, reaching an accuracy of more than 90% on the Books dataset (which is easier since it exhibits the larger variety of style and topics), and attaining above 88% on the more challenging CCNews dataset (for which generation is easier and hence discrimination harder). The Wikipedia dataset has lower accuracy because the EBM overfits to this smaller dataset.

[1]`https://github.com/NVIDIA/apex`

	Books	CCNews	Wiki
Linear	59.8	58.6	56.3
BiLSTMsmall	84.7	77.6	71.0
BiLSTMbig	86.7	80.1	72.8
UniTransf	91.7	88.4	76.4
TransfBig (log-likelihood)	57.1	50.8	50.5

Table 2: "In domain" generalization accuracy of EBMs (each row) on various text corpora. A column corresponds to the corpus used to get positives and to fit the train and test language models, which are TransfBig (§4.2) with different initial seeds. The last row is the accuracy when using as energy the log-probability of the training language model over the whole sequence.

	Conv	TransfSmall
Conv	92.9	81.2
TransfSmall	86.5	87.9

Table 3: Cross-architecture generalization accuracy using the Wikitext dataset for both training and testing ($C_{\text{train}} = C_{\text{test}}$). Each row is a model architecture used for generating the training negatives (A_{train}), and each column is a model architecture for generating the testing negatives (A_{test}). The energy function is UniTransf.

Weaker energy models are able to do comparably or better at discriminating real from fake than the training generator used as a discriminator by taking the log probability of the sequence as energy.

5.2 Cross-architecture generalization

In Table 3, we assess how well the UniTransf energy function generalizes to different generator architectures at test time, namely Conv and TransfSmall. As a reference on the Wikitext dataset, the test perplexity of Conv and TransfSmall are 35.4 and 33.5, respectively. Therefore, these two generators attain roughly the same perplexity, despite Conv having about 4 times more parameters, see Table 9.

Surprisingly, UniTransf has significantly harder time discriminating TransfSmall negatives with an in-domain rate of 87.9%, compared to 92.9% of Conv. Also, UniTransf trained with TransfSmall negatives is more robust to the (weaker) Conv generations, than vice versa, with a mild 1.4% accuracy drop. However, if we average values across rows, we see that UniTransf tested with mixed negatives is just slightly more accurate when training with the harder negatives produced by TransfSmall.

5.3 Cross-Corpus generalization

In Table 4 we show the results of generalizing across corpora using UniTransf as an energy function and TransfBig as generator both at training and test time. We observe that models generalize less well across corpora; for instance, when testing on Wikitext an energy function trained with either Books or CCNews, the accuracy is 59.1% and 65.5%, respectively. 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 two rows) yields an energy function that is very robust to the testing conditions, with an accuracy which is on par if not better than training on in-domain data, even for the largest CC-News dataset (second column).

We also tested the bidirectional transformer energy function BiTransf with 355M parameters (almost twice as UniTransf), and found that on CC-News it improves accuracy by more than 5% when it is trained on the union of all corpora, confirming the finding that bigger models trained on more data can achieve substantially better discrimination. As BiTransf was pre-trained using the whole Wikipedia rather than the training part of Wikitext103, we do not report its accuracy on Wiki test set.

TRAIN CORPORA	TEST CORPORA Books	CCNews	Wiki
Wiki	70.9	73.6	76.4
Books	91.7	63.5	59.1
Books + Wiki	91.5	73.6	78.3
CCNews	60.6	88.4	65.5
Books + CCNews	90.4	88.5	68.3
CCNews + Wiki	73.5	88.3	81.0
ALL (UniTransf)	90.0	87.9	80.5
ALL (BiTransf)	91.6	93.0	-

Table 4: Cross-corpora generalization accuracy using TransfBig generator and UniTransf energy function (except for the last row which used a bidirectional transformer). Each row specifies the corpora used at training time, C_{train}. Each column shows the corpus used at test time, C_{test}.

Energy function →		UniTransf trained on ALL			BiTransf trained on ALL	
Test domain →		Books	CCNews	Wiki	Books	CCNews
Generator model ↓	M params ↓					
GPT2 small (WebText)	137	90.9	80.4	82.6	92.0	87.6
GPT2 medium (WebText)	380	88.1	73.2	78.8	89.6	78.4
TransfHuge (CC-News)	1427	81.8	66.1	75.1	82.3	66.9

Table 5: Generalization of BiTransf and UniTransf energy function to state of the art generators. The energy functions (discriminators) are trained on the concatenation of all three corpora (same as in table 4 ALL rows). Test time negatives are generated by models specified in the rows, with their training set in parenthesis and model size in millions of parameters. Note that both the training corpus and GPT2 generator are "unseen" by the energy function during training.

5.4 Generalization in the Wild

In Table 5 we test the generalization of the energy functions to GPT-2 generators (Radford et al., 2019)[2] that were trained on a completely different dataset, namely WebText (Radford et al., 2019) a dataset of 8 million web pages. This is an instance of unseen generalization since $C_{\text{train}} \neq C_{\text{test}}$, and $A_{\text{train}} \neq A_{\text{test}}$. We also consider generations from TransfHuge (last row) whose configuration is similar to the unreleased biggest GPT2 model with 1.4 billion parameters, 7 times bigger than TransfBig, the generator used at training time.

Expectedly as the generator gets bigger the discrimination tasks gets harder. When the energy function is confronted with generations from the GPT2 small model, which is smaller than the training generator, the accuracy is close to the in-domain setting, however. For instance the BiTransf accuracy increases by 0.4% on the Book corpus and decreases by 5.4% on the CCNews corpus compared to the fully in-domain results of Table 4. That suggests that for a known domain, a big enough energy model trained with a single big generator can efficiently discriminate a *block-box* generator. Of course, accuracy decreases as the black-box generator is made bigger (GPT-2 medium).

Finally, we investigate generalization of the energy function to a new domain, such as samples from the dataset of GPT-2 generations (Radford & Wu, 2019). For each model the dataset has a 250k generated texts with either top-k sampling or random sampling. Also, the dataset provides samples from the WebText corpus that was used to train the generator models, and that we use to discriminate against.

To adapt our models to this task we split the text segments into sets of intersecting blocks of 160 tokens. During training we treat all blocks in a set as either positives or negatives. During evaluation we take the mean prediction over all blocks in a segment as a prediction for the whole segment.

[2]We use pytorch versions of officially released checkpoints for small (137M params) and medium (380M params) architectures from HuggingFace repository at `https://github.com/huggingface/pytorch-transformers`.

Energy Function →	TF-IDF*	BiTransf	
Test setting →	in-domain	in-domain	generalization
Small (137) top-k	96.79	99.09	-
Small (137) random	88.29	99.80	-
Med (380) top-k	95.22	98.07	97.37
Med (380) random	88.94	99.43	97.35
Big (762) top-k	94.43	96.50	93.58
Big (762) random	77.16	99.42	95.96
Huge (1542) top-k	94.43	95.01	90.17
Huge (1542) random	77.31	99.00	91.76

Table 6: Generalization of the energy function to *unconditional* generation from various GPT2 models (model size in parantheses, followed by sampling method used). Each row contains the accuracy on the corresponding test set. TF-IDF results are taken from Radford & Wu (2019).

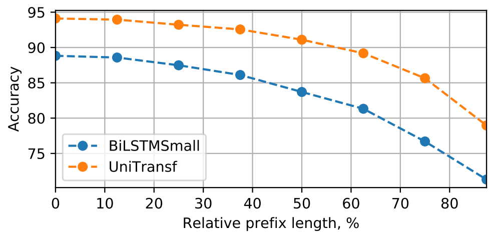

Figure 1: Discrimination accuracy as a function of the ratio between the prefix length and the total length of the sequence on the Wikitext dataset.

	Accuracy
1 random negative	82.9
3 random negatives	84.2
worst negative out of 3	84.6

Table 7: Effect of different strategies to mine negatives using TransfBig generator and BiLSTMSmall energy function on Book Corpus.

In Table 6 we report results of the BiTransf energy function compared to the TF-IDF baseline provided with the dataset. We consider two cases. In the in-domain setting, we finetune the energy function on the train set of each of the datasets, following the same protocol used by the provided TF-IDF baseline. In generalization mode, we finetune only on the generations from the small GPT2 model (both top-k and random sampling), and apply the model to the other datasets.

Unsurprisingly, in-domain BiTransf beats TF-IDF baseline getting almost 100% across the board. However in generalization mode, we can outperform the TF-IDF baseline only when the generator is less than three times bigger than what was used at training time.

Interestingly, our energy function was trained using a fixed length input with a prefix. These generalization results are significantly higher than the in-domain experiment of Table 2 because the unconditional task is significantly easier, a topic further discussed next.

5.5 Ablation Study

First, we investigate the dependency between performance of the energy functions and length of the prefix. We trained BiLSTMSmall and UniTransf models on examples with varying prefix length from the Wikitext corpus, and computed the accuracy for each prefix length independently. Figure 1 shows that as the prefix length increases (and the generation gets shorter), the discrimination task gets harder and the difference between the models more prominent. The unconditional case, i.e. zero prefix length, is the easiest, while prefixes of length 120 and 140 that are the main experimental setup in this work, are the hardest.

Finally, in Table 7 we study the impact of the number of negatives and using the most offending negative in the loss of Eq. (1). Using more negatives and harder negatives improves accuracy.

5.6 Stability to Other Negative Distributions

In the previous sections we have seen that the energy function is less robust to negatives generated from a model trained on a different corpus. However, even in that case, a negative is still a sample

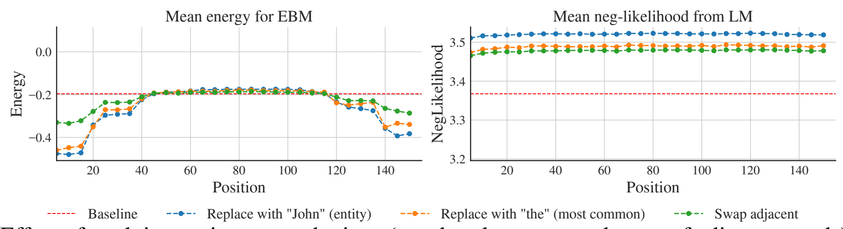

Figure 2: Effect of applying various perturbations (word replacement and swap of adjacent words) to ground-truth sequences at different positions in terms of energy function and generator negative log-likelihood (averaged over the whole test set of Wikitext). The energy is only affected by corruptions at either end of the sequence. These out-of-domain corruptions invariably decrease the energy. However, all perturbations increase the negative log-likelihood of the sequence.

from an auto-regressive neural network. In Appendix F, we show examples where changing a few entities can cause large jumps in the energy (from negative to positive or vice versa), and so fool the EBM. More generally, we see that the energy function is not robust to truly out-of-domain samples. For example, the energy will score blocks of randomly generated text lower than real text.

These behaviors are evidence that the energy 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 energy function assigns low score to anything that is not generated by its training generator.

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

In Figure 2 we show the average effects of applying various perturbations to sequences from Wikitext103 on an in-domain energy and language model at each location (from 1 to 160) in the sequence. We see that for all perturbations, the energy decreases its value, but the language model increases its negative log likelihood. We also see that the energy function is more sensitive to the ends of the text, which is where the negatives were different from real text at training time.

6 Final Remarks

The EBM framework could potentially unlock more expressive models of text, as they are not limited to scoring a single word at a time as current locally normalized auto-regressive models do. Unfortunately, training EBMs is challenging because generating negatives using the energy function itself is still an open research problem, and does not scale well in practice. In this work, we propose a simple solution, which is to leverage generations produced by pre-trained language models as negative samples.

As a preliminary yet necessary step in this direction we have investigated the generalization ability of such EBMs. We found that EBMs, when trained on large datasets, achieve good generalization. For instance, they behave nicely when tested with negatives produced by generators that have rather different architectures. The generalization is less good when generators are trained on other corpora, but EBMs re-gain robustness once we train them on even bigger composite datasets.

In the future, we can improve EBMs for text by simply making their architectures bigger and increasing the diversity and size of their training datasets. Of course, further scaling up of EBMs will pose formidable engineering challenges.

On the application side, a natural application of the current formulation of EBMs is real/fake text discrimination. We believe that this is important application in its own right, and that EBMs can be very powerful, as demonstrated by their superior performance compared to discriminating using the original language model log-likelihood.

We additionally hope to broaden the scope of applications of EBMs for text, including learning generic representations of text and using EBMs to improve text generation.

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.

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*, pp. 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.

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.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. GLTR: statistical detection and visualization of generated text. *CoRR*, abs/1906.04043, 2019. URL http://arxiv.org/abs/1906.04043.

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.

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.

John Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *National Academy of Sciences of the USA*, volume 79, pp. 2554—2558, 1982.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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*, pp. 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.

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 and Jeff Wu, 2019. URL `https://github.com/openai/gpt-2-output-dataset/blob/master/README.md`.

M. Ranzato, V. Mnih, J. Susskind, and G.E. Hinton. Modeling natural images using gated mrfs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(9):2206–2222, 2013.

Marc'Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun. A unified energy-based framework for unsupervised learning. In *11-th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2007.

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.

Libin Shen, Anoop Sarkar, and Franz J. Och. Discriminative reranking for machine translation. In *Association of Computational Linguistics*, 2004.

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*, pp. 5998–6008, 2017.

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

Max Welling, Michal Rosen-Zvi, and Geoffrey E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Neural Information Processing Systems*, 2005.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In *Neural Information Procesing Systems*, 2019.

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 8: Number of BPE tokens in millions for each dataset.

B Model Sizes

	Generators							
					Pre-trained GPT2			
	Conv	TransfSmall	TransfBig	TransfHuge	small	med	large	huge
embed.	13	26	51	77	39	52	-	-
others	164	19	151	1360	97	327	-	-
total	176	45	203	1437	137[a]	380[a]	762[b]	1542[b]

Table 9: 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).

[a]We use models from HuggingFace repository (`https://github.com/huggingface/pytorch-transformers`) and report here the sizes of these models as they were used to generate data for table 5. Note that the OpenAI GPT2 repository (`https://github.com/openai/gpt-2`) defines models sizes as 124M and 355M for small and medium model correspondingly.

[b]As reported in Radford et al. (2019).

	EBM Functions				
	Linear	BiLSTM	BiLSTM Big	UniTransf	BiTransf
embed.	0.1	26	39	51	51
others	0	23	90	151	304
total	0.1	49	129	203	355

Table 10: Number of parameters in millions for the scoring functions. The computational cost is directly related to the number of parameters in other layers than the input embedding layer (second row).

C Ranking Loss

The (per-sample) ranking loss is:

$$\mathcal{L}_{\text{R}} = \max\left(0, 1 + E(x^+|c;\theta) - E(x_i^-|c;\theta)\right). \tag{2}$$

In this case we also refer to the negative energy as the model *score*. The ranking loss makes the energy values *local*, as the loss takes as input the difference of energies for a pairs of positive and negative that share the *same* context. Instead, the binary cross entropy loss of Eq. 1 encourages a more *global* and absolute scoring as the loss forces all positive examples to have negative energy, and all negative samples to have positive energy, regardless of the context. Therefore, the binary cross entropy loss is perhaps more interpretable as it is not context dependent, but the task is also harder to learn. Empirically, we found similar findings with both losses.

When the energy function is trained using the ranking loss of eq. 2, we evaluate the model using *precision at 1* (P@1), which is the ratio between the number of times the ground truth sequence scores the lowest over its set of negatives, averaged over the number of sequences in the test set.

D Hyper-parameter Setting

All models are implemented using the PyTorch framework (Paszke et al., 2017) and are optimized using Adam (Kingma & Ba, 2014).

To train our biggest models (UniTransf and BiTransf) 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 & 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	-
BiLSTM	0.0002	128	8	+	1000	-
UniTransf	0.0003	32	64	+	2000	180000
BiTransf	0.00005	20	64	+	2000	180000

Table 11: Hyper-parameter values used in our scoring functions.

E Score Distributions

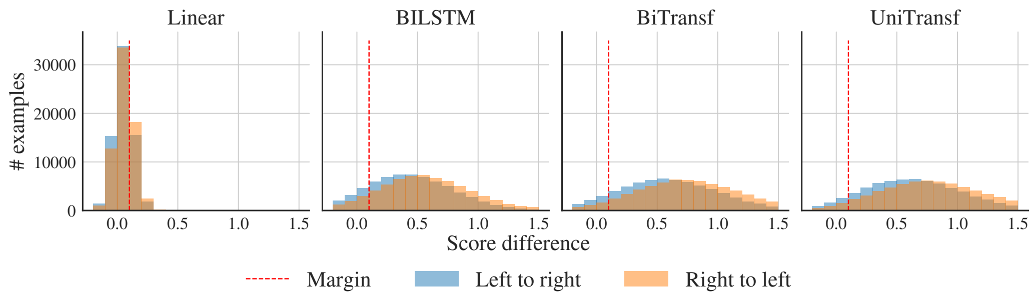

Figure 3: Distributions of score (negative energy) differences between pairs of ground truth completions and generated ones for different scoring models. We show results for two generations (left to right and right to left) from Wikitext dataset. In both cases we generate 40 tokens. Examples on the right of the red line ($margin = 0.1$) have zero rankiing loss.

F 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 energy function, and vice versa, by using gradient information.

Below here, we show an example of a 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]

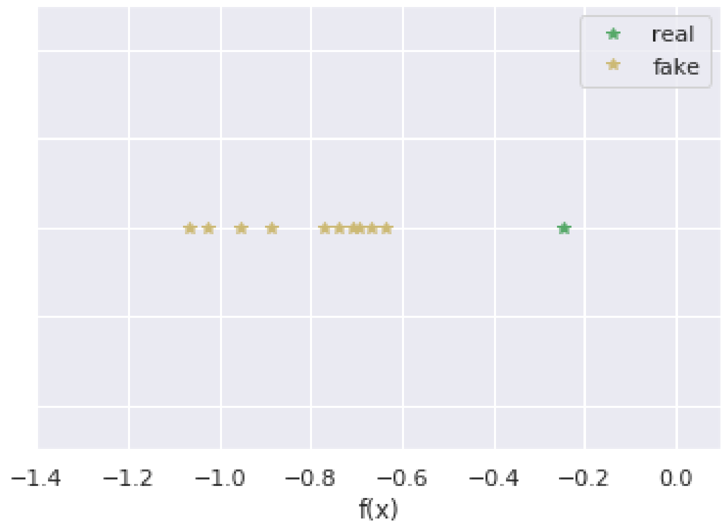

Figure 4: Real and fake (negatives generated from TransfBig language model) completions as scored by the learned energy function. The energy function is able to separate them well. These scores are calcuated based on the single example reported in the main text of §F. $f(x)$ is the negative energy.

In this example, using the big transformer model, UniTransf, as the energy function, we are able to separate real from fake examples as shown (Figure 4). We want to perturb these negatives to violate the margin. To do so, we make use of the gradient information from the energy function $\nabla_x E_\theta(x)$ 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:

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

Using this approximation, we can search for those token replacements that increase/decrease the energy 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 5, 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.

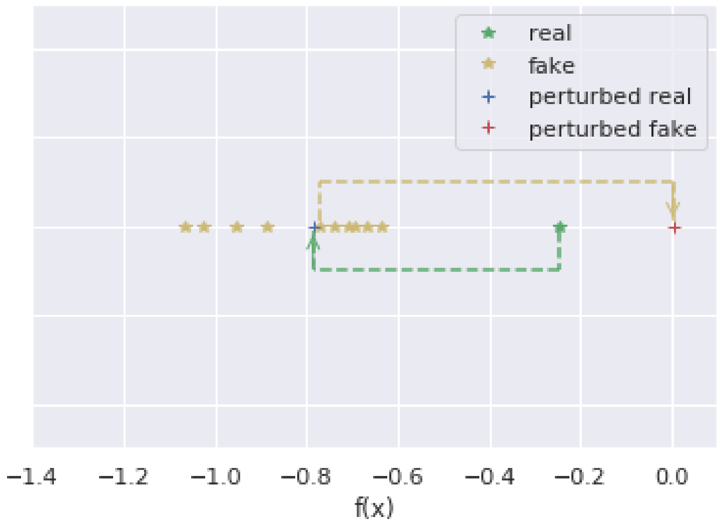

Figure 5: By changing a few words we can make a negative sample become real as scored by the (negative) ennergy function, and vice versa.