An Analysis of Entropy-Based Techniques in Generative Language Models

Aneesh Durai

May 26, 2024

1 Introduction and Goals

Generative AI tools, such as GPT, are trained with a goal of minimizing their entropy. This framework is adopted by language models to best enhance their efficiency. Our goal is to explore how we can entropy and information in language models and how we can optimize it for generative tasks. We also investigate the dynamics of these entropy techniques to see if it helps optimize our models. In addition, we examine various methods that helps reduce the loss in language models, aiming to improve their performance and understand the implementation of entropy reduction frameworks.

2 Background/Context

This project delves deeply into the structure of language models by exploring their core learning components from the ground up. Using principles from information theory, we want to see which characteristics can be fine-tuned to enhance the model's performance. The focus of this project is to deeply understand these methodologies, but also gain an understanding of how we can use these methodologies in large-scale projects.

3 Related Work

The paper *Entropy-Regularized Token-Level Policy Optimization for Large Language Models*[4] introduces a novel approach that utilizes concepts from reinforcement learning and entropy regularization to optimize language model performance. This approach, known as EPTO, models language generation as a sequential decision-making process, similar to that of a Markov Decision Process. It employs entropy regularization in order to balance the trade-off between training data and prediction variability. This framework can be very useful for techniques in text generation application, such as contextual dialogue systems, by applying entropy dynamics effectively within language models.

Another entropy-based technique is found in *Cross Entropy of Language Models at Infinity-A New Bound of the Entropy Rate*[2], which explores an advanced neural language model's behavior under varying entropy rate condition. This study uses cross entropy to measure a model's prediction accuracy, providing insights into how entropy-based techniques are employed during model training. This approach can be instrumental in refining the regularization processes of LSTM models to better manage entropy dynamics.

Furthermore, *Experimenting with Power Divergences for Language Modeling*[3] dives into the use of power divergences within the framework of maximum likelihood estimation for language model training. This paper illustrates that practical application of different divergence measures to influence the training and validation dynamics and generative outcomes of language models, especially in cases when we are working with uneven distribution frequencies. This research can be valuable for adjusting probability masses when we have a unique dataset with a diverse array of unique words, effectively supporting both high and low-level frequency words.

Using the ideas from these papers, we can attempt to uncover the inner-workings of how information theory can be used to enhance a model's performance. From these papers, we know that language models typically integrate cross-entropy as an optimal loss function. However, models have other methods of improvement such as entropy regularization and implementing power divergences. If we start from a benchmark model, we can try integrating similar techniques to see how that model improves.

4 Experiment and Analysis

4.1 Control Model

The primarily goal of this experiment is to explore how integrating entropy-based terms can improve the performance of our language models, particularly focusing on text generation sequences. For this purpose, we developed a baseline LSTM model, using Shivam Bansal's *Beginner's Guide to Text Generations Using LSTMs*, with some key modifications to tailor it to our research obejctives:

```
def create_model(total_words, max_sequence_len, custom_loss=None):
    input_len = max_sequence_len - 1
    model = Sequential([
        Embedding(total_words, 10, input_length=input_len),
        LSTM(100),
        Dropout(0.1),
        Dense(total_words, activation='linear')
    ])
    loss = custom_loss
    model.compile(loss=loss, optimizer='adam')
    return model
```

In the original guide by Bansal, the model employs a softmax activation layer and the categorical cross-entropy loss function, which are standard for text generation modles. However, our project's focus on entropy-based optimization make it difficult for us to use this. Therefore, we will replace the softmax with a linear activation function and the cross-entropy loss with a mean-squared error (MSE). This is based on the idea that the traditional softmax, combined with entropy-based adjustments, with neglect the ffects of modifications based on probability-distribution-based outputs.

To evaluate the control model, we utilize seed texts to generate words, training the model on the New York Dataset provided in Bansal's guide. For experimentation purposes, we will split the data into 80% training and 20% validation. Here are the initial results:

Seed Text	**Next 10 Words Generated**
United States	[To To A A To A A To A A]
President Trump	[To A To A A To A A To A]
Donald Trump	[To A To A A To A A To A]
India and China	[A To A A To A A To A A]
New York	[To A A To A A To A A To]
Science and Technology	[To To A A A To A A To A]

The output suggests that the model is not effectively learning from the seed text. The results show that there is no relevance in the generated words and also no significant information gain taking place in the training process, with the model predominantly outputting simple structures. This suggests that our model requires further optimization in order to genuinely capitalize on the benefits of entropy-based approaches. In addition, there is no specific trend in either the training or validation loss across epochs which suggests that we need significant refinements in our model.

4.2 Entropy Regularization

So in order to address this challenge, we introduce entropy regularization into our model, which is inspired by the EPTO framework. This approach involves modifying the loss function to include a penalty for high entropy outputs, thus with the intention of promoting more precise and informative predictions. Here is the adjustment to our loss function to incorporate entropy regularization:

```
def entropy(y_pred):
    y_pred = K.clip(y_pred, K.epsilon(), 1)
    return -K.sum(y_pred * K.log(y_pred), axis=-1)

def original_loss(y_true, y_pred):
    return keras.losses.mean_squared_error(y_true, y_pred)

def custom_loss(y_true, y_pred):
    return keras.losses.mean_squared_error(y_true, y_pred) + 10e-8 *
       entropy(y_pred)
```

With this new model, we care about monitoring four key metrics: mean entropy, mean Kullback-Leibler (KL) divergence, training loss, and validation loss. Our initial observations suggest no significant changes in training or validation loss across epochs, suggesting that the model was not effectively learning or adapting based on the provided data. This indicates that there might be potential issues in the effectiveness of the entropy penalty.

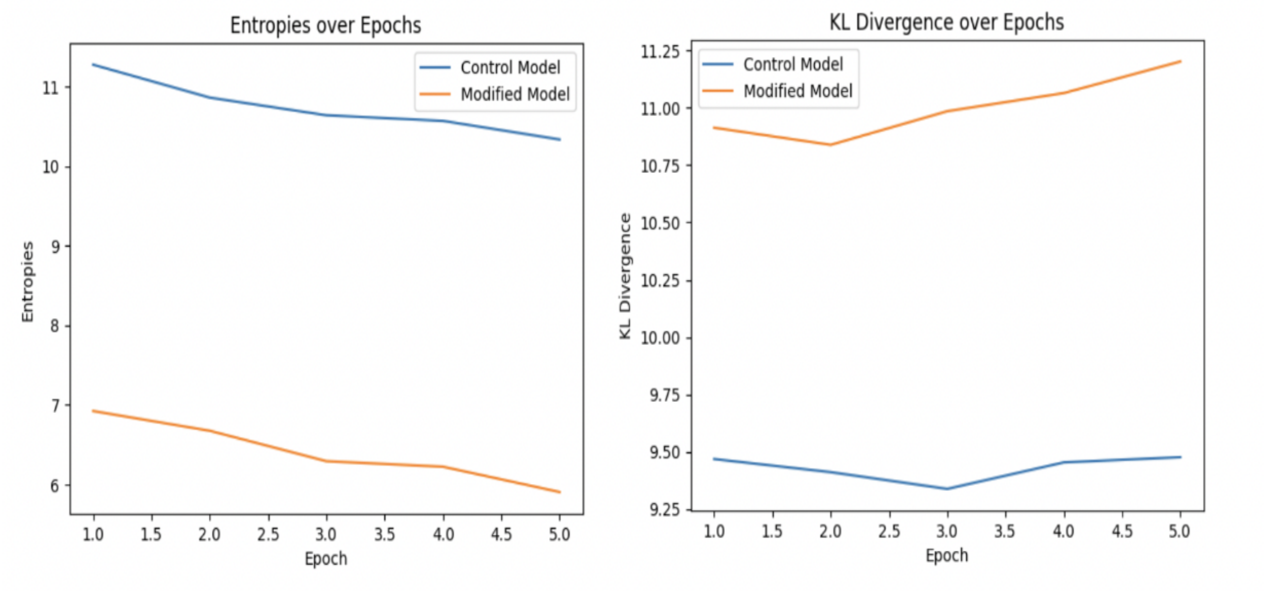

Figure 1: Control Model vs. Entropy Regularized Model Performance

As illustrated in Figure 1, the entropy levels at the beginning of training are slightly lower in the modified model than in the control, although the rate of change remains consistent between the models. This suggests that the added entropy penalty does not have any effect on improving our information gain as we were hoping for. On top of that, the KL divergence is actually increasing rather than decreasing, indicating a divergence between the predicted and actual probability distributions, which is not what we expect in a model that is learning.

This observation thus leads us to look into the weight of the entropy penalty, which could have an affect on the performance. In Figure 1, we use a weight of 10^{-7}, so adjusting this parameter could possibly optimize performance. Therefore, we experiment with various weights for the entropy regularization to explore what effect that has on the model behavior.

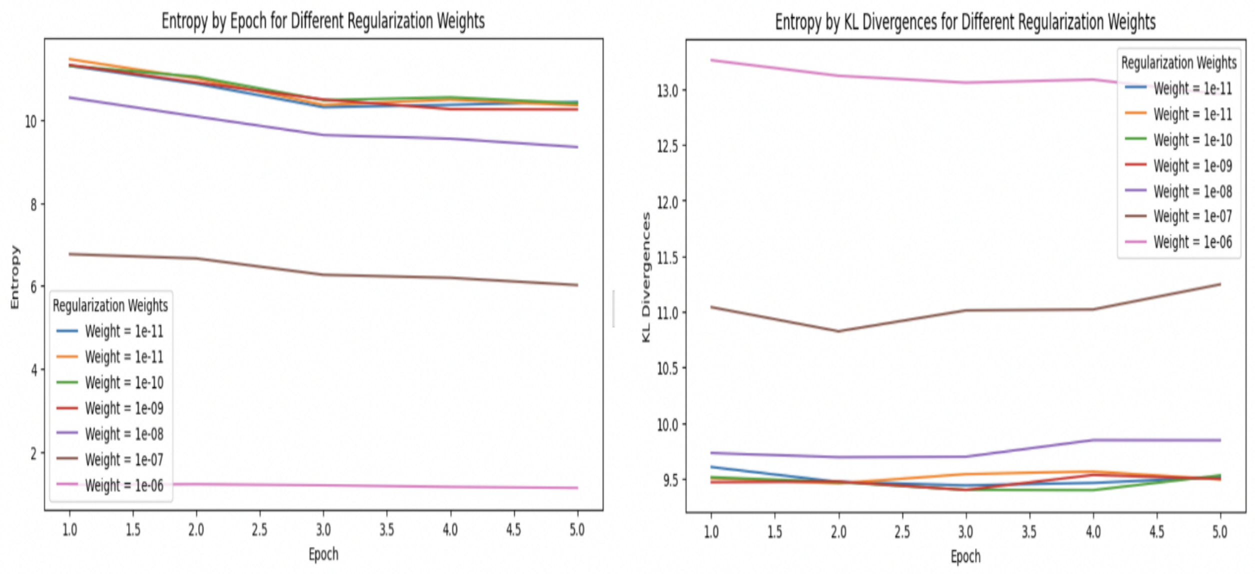

Figure 2: Performance of Entropy Regularization for Different Weights

Despite experimenting with different weights, neither the entropy nor KL divergence demonstrated a downward trend, suggesting that the regularization approach might be fundamentally misaligned with the learning dynamics of the models. More importantly, the validation and training loss remained relatively constant over all five epochs which is the strongest indication that the model is not learning. However, the graph indicates persistently high KL divergence that is flat lined. These findings suggest that we can explore an alternative strategy, regularizing the divergence, which directly penalizes the model based on the distance between the predicted and true distributions. We expect the KL divergence to decrease per epoch indicating that our predicted distribution is getting closer to the true distribution.

4.3 Divergence Regularization

So in order to enhance our model performance, we incorporate KL divergence into our loss function, which ensures that there is a penalty for divergence between the predicted and actual distributions. This approach is consistent with that of recent research where divergence metrics can significantly influence model optimization.

```python
def kl_divergence(y_true, y_pred):
    y_true = K.clip(y_true, K.epsilon(), 1)
    y_pred = K.clip(y_pred, K.epsilon(), 1)
    return K.sum(y_true * K.log(y_true / y_pred), axis=-1)

def original_loss(y_true, y_pred):
    return keras.losses.mean_squared_error(y_true, y_pred)

def custom_loss(y_true, y_pred):
    kl_div = kl_divergence(y_true, y_pred)
    return keras.losses.mean_squared_error(y_true, y_pred) + 10e-8 *
        kl_divergence(y_true, y_pred)
```

Given the significant role of the regularization weight in entropy regularization, we need to perform hyperparameter tuning for the KL divergence regularization as well to finely tune the model's dynamics. We can experiment with varied weights and evaluate the patterns of the entropy, KL divergence, training loss, and validation loss.

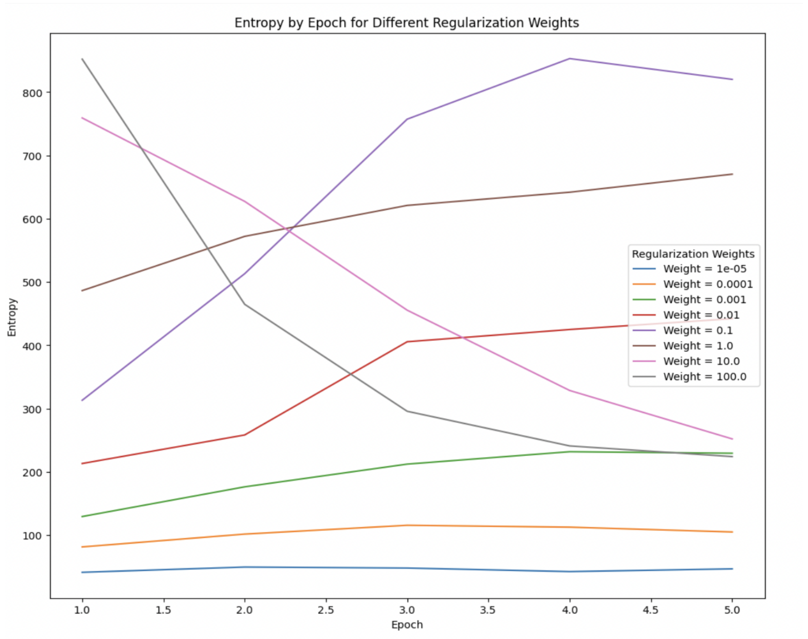

Figure 3: Entropies of KL Divergence Regularization for Different Weights

The results show diverse patterns. However, a key indicator of a well-performing learning model is its consistent decrease in entropy, suggesting a progressive learning across epochs. Notably, the weight of 100 demonstrated this with the entropy levels converging as the training progressed. However, we still need to look at the other metrics to ensure we are not over fitting.

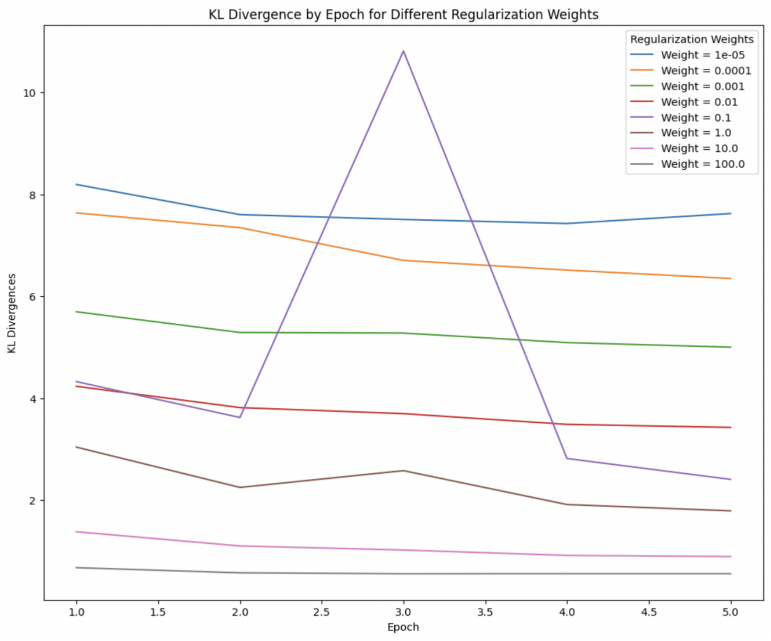

Figure 4: KL Divergences of KL Divergence Regularization for Different Weights

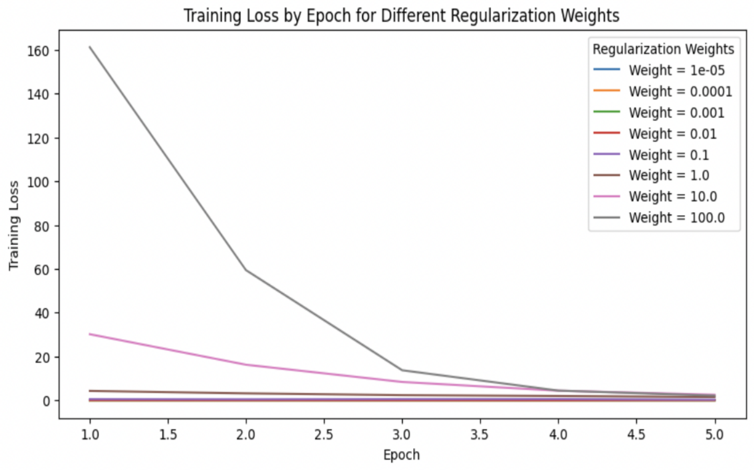

Figure 5: Training Loss of KL Divergence Regularization for Different Weights

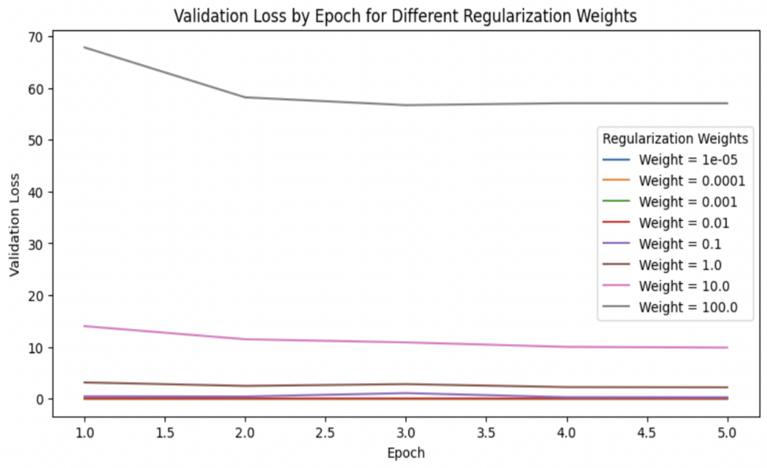

Figure 6: Validation Loss of KL Divergence Regularization for Different Weights

Figure 4 and 5 indicate that weights 10 and 100 yield decreasing KL divergences and converging training losses, indicating that they are strong candidates. However, Figure 6 highlights a potential issue with weight 100, as the validation loss remains relatively high compared to the training loss. In contrast, the weight 10 demonstrates a more comparable performance between the training and validation losses, suggesting that it is a more optimal choice for our model.

4.4 Predictions

So utilizing our modified model configured with a weight of 10 for KL divergence regularization, we conducted text generation experiment on the same dataset. One notable component that we needed to change in our text generating function is that we are now focusing on highest probability outputs to better reflect the model's new potential.

Seed Text	**Next 10 Words Generated**
United States	[Hair Gop Hair Progressive Race Progressive Gop Hair Race Progressive]
President Trump	[Progressive Race Complain Complain Progressive Complain Progressive Complain Race Complain]
Donald Trump	[Hair Progressive Gop Gop Gop Complain Gop Progressive Race Gop]
India and China	[Gop Race Gop Race Hair Progressive Progressive Complain Race Complain]
New York	[Race Progressive Progressive Complain Progressive Race Gop Progressive Complain Race]
Science and Technology	[Hair Progressive Progressive Progressive Hair Race Gop Complain Progressive Progressive]

These results demonstrate a clear advancement compared to our initial model outputs, which predominantly generated repetitive and irrelevant words such as "To" and "A". The current outpts are more diverse and contextually varied, suggesting that our model has a better learning capability. In this specific data set, there are a lot of terms associated with President Trump's presidency which could explain why seed terms like "Science and Technology" are yielding terms like "Progressive" and "Hair."

For a more thorough evaluation, we can also compare this with the performance of our other regularization weight candidate, 100. This yields the following outputs.

Seed Text	**Next 10 Words Generated**
United States	[Crash An Left An 23 An Flight 23 Flight An]
President Trump	[23 An 23 Left An Left Left 23 Flight An]
Donald Trump	[23 Flight 23 23 An 23 Flight An Left An]
India and China	[23 An 23 Flight Left An An Flight An 23]
New York	[23 Left 23 23 Flight Crash Left Crash An Crash]
Science and Technology	[An Flight 23 23 Crash Flight Left Flight An Left]

While this adjustment shows an improvement from the control model's outcomes, the generated text still lacks any relevance with the seed text indicating that there is potential overfitting with this higher regularization weight. Thus, we conclude that a weight of 10 offers the most optimal balance, providing a more meaningful enhancement to the model.

5 Results

Our experiment conclusively shows that KL divergence regularization offers a substantial benefit for text generation language models when compared to other methods. Even though we had promising entropy regularization, it did not exaclty perform optimally given our model's configuration. Instead, divergence regularization, which was inspired by Labeau's work on power divergences, provided a more stable and consistent training and validation loss result. That is a strong indication of effective learning.

Some of the challenges with this experiment included constructing a fundamental baseline model and tackling with initial accurate predictions. These issues were resolved by experimenting with different model architectures as well as implementing a generating function that adopts a maximum likelihood strategy.

6 Discussions and Conclusions

This project provided a strong theoretical understanding of the learning theory associated with language mdoels through hands on-experimentation. The standard practice is usually using built-in frameworks in TensorFlow or similar liraries. The application of information theory in refining these models proved to be a more enlightening approach to better understand how language models work and also opening possibilites for working with Large Language Models (LLMs) in the future.

Some other possibilities for this project include tackling it from a broader spectrum. For example, using categorical cross-entropy loss as a benchmark could prove valuable to further validate the enhancement we made in our modified model. In addition, it would help to explore this project with a wide array of data sets to see if there is are any other insights that we may have overseen in this experiment. In addition, there might be different model architectures that are more appropriate for this experiment that are worth looking into. Our experiment simply served as a framework on how we can potentially answer our questions using more advanced tools.

7 Code

The simulation can be found here.

8 Bibliography

1. Bansal, Shivam. "Beginner's Guide to Text Generation using LSTMs." Kaggle. Retrieved from `https://www.kaggle.com/code/shivamb/beginners-guide-to-text-generation-using-lstms`.
2. Ben-Nun, Tal, and Shai Shalev-Shwartz. "Cross Entropy of Neural Language Models at Infinity-A New Bound of the Entropy Rate." *arXiv preprint* arXiv:2104.00690 (2021).
3. Muller, Raphael, and Ahmed S. Zaki. "Experimenting with Power Divergences for Language Modeling." *arXiv preprint* arXiv:2102.11225 (2021).
4. Press, Ofir, and Lior Wolf. "Entropy-Regularized Token-Level Policy Optimization for Large Language Models." *arXiv preprint* arXiv:2105.09680 (2021).