GPTNo: A Deep Learning LLM to Beat the Turing Test

Stanford CS224N Custom Project, Mentor: Lisa

Sri Jaladi, William Li, Abhinav Sinha
Stanford University, Department of Computer Science
sjaladi@stanford.edu, willeli@stanford.edu, absinha@stanford.edu

Abstract

The objective of this project is to create a model that generates sentences which deceive AI Detectors, such as GPT-Zero, DetectGPT, and OpenAI AI detectors, while still preserving the original meaning, and sentence coherence. This paper discusses 2 strategies of achieving the goal: 1) Finetuning the BERT large model to be able to generate sentences with these new parameters 2) Adding a deep neural network to the output of BERT generated text to output new text which can fool these detectors. This will be achieved by replacing sentence components that have low perplexity, high probability of occurrence, and low variance. The project will employ three different loss functions to fine-tune the model and evaluate its performance against GPT-Zero's API. The primary focus will be on training the model to modify the fewest words possible in order to indistinguishably mimic real human language and meaning. The success of the model will be evaluated through testing and improving its performance against other detectors. The model proved to be highly effective, exhibiting the ability to deceive top-tier AI detectors in 70% of instances, all while maintaining sentence coherence and similarity.

1 Introduction

1.1 Contemporary Relevance

The escalating employment of interactive LLMs in public settings has led to an upsurge in research aimed at discerning between texts generated by humans and those generated by AI. Numerous instances of such research have been reported in recent news, with GPTZero being the most prominent [1]. Additionally, there are numerous other detectors, including OpenAI's own DetectGPT, and others that attempt to identify AI-generated text by assessing perplexity, i.e., the likelihood of a given word following the previous word for a specific model (such as GPT), as well as variance in perplexity [2]. Although some may view this as a game of cat and mouse, with detectors becoming more advanced and models likewise improving, this is not the case. Models like GPT-3.5 are primarily designed to provide succinct information rather than simulating human text or being more human-like.[3] Consequently, there has been no research aimed at altering GPT text in the proposed manner.

1.2 Problem and Task

Our aim is to develop a model that can enhance the human-like qualities of GPT-generated text. As AI continues to evolve, the next logical progression is for AI models to emulate human characteristics and improve conversational quality to make them more human-like. This development has numerous applications across a diverse range of fields, including but not limited to, education, where AI could act as a teacher and help children learn in a more personalized, empathetic, and relatable way. AI's human-like qualities could also be leveraged to provide therapy and mental health counseling to patients, where the presence of a familiar figure can have a positive impact on their mental well-being.

Specifically, we seek to create a model that can modify artificially-generated text while preserving maximum grammatical and semantic accuracy while minimizing the risk of detection as artificially generated. To accomplish this, we will build a mask model that utilizes a mask-and-fill technique to modify AI-generated text with minimal disruption. Our model will employ a pre-trained BERT model that we will fine-tune to mask and replace certain words in the input text to optimize a custom loss function [4]. Upon completion, our model will take as input artificially-generated text and generate an output that is as human-like as possible while minimizing the likelihood of detection as artificially generated.

2 Literature Review/Related Work

The first component researched is previous methods in AI detection. Tang et al. details a methodology utilizing linguistic and statistical features to detect AI-generated text [5]. The proposed methodology comprises of using feature extraction, selection, and classification to identify AI-generated text. The researchers were able to extract important feature differences between AI and Human-generated text using SVMs (support-vector-machines). The authors conducted a series of experiments on a dataset of 1,000

texts, half of which were generated using a GPT model and half of which were written by humans. The results indicate that the proposed achieved an accuracy of 95.1% and an F1 score of 0.95.

The paper Mask and Infill: Applying Masked Language Model to Sentiment Transfer addresses the task of sentiment transfer, which involves generating a new sentence with an altered sentiment while retaining the original meaning and content as much as possible [6]. Unlike previous research, which often involves entirely rephrasing or transforming a sentence, this paper preserves as much of the original sentence as possible, modifying only necessary words and phrases to achieve the desired sentiment. The paper does this via the mask-and-fill technique, which masks important words and phrases related to the sentence's sentiment and then modifies them appropriately. The results of this paper are promising and motivated the project's usage of the mask-and-fill technique.

Next, paper "A Perplexity-Based Method for Similar Languages Discrimination" tries to discriminate between similar languages based on perplexity [7]. The authors argue that similar languages should have similar perplexities, while less similar languages should have more distinct perplexities. The authors used a language modeling technique based on n-gram models to estimate the perplexity of each language, then calculated the pairwise Euclidean distances between the perplexity vectors and performed a cluster analysis to determine the ability of the perplexity-based method to discriminate between the languages. The results of the study showed that the perplexity-based method was able to accurately discriminate between the four pairs of languages with an accuracy rate of over 90%.

We learned from the first paper that there are particular strong features in sentences that could help distinguish AI-generated text and human-generated text. Then the third paper, discussed how perplexity is a major feature in many languages and can help distinguish between various similar languages. From there, we chose to use perplexity as the feature we wanted to manipulate as it could then it could help fool many of these models that try to detect AI. However, we wanted to change the perplexity of a sentence while keeping the sentence meaning the same, this led to us discovering the sentiment transfer paper as it took a separate feature (sentiment) and changed the sentence based on that feature while preserving meaning and coherence. This brings us to our approach.

3 Approach

3.1 Loss Function

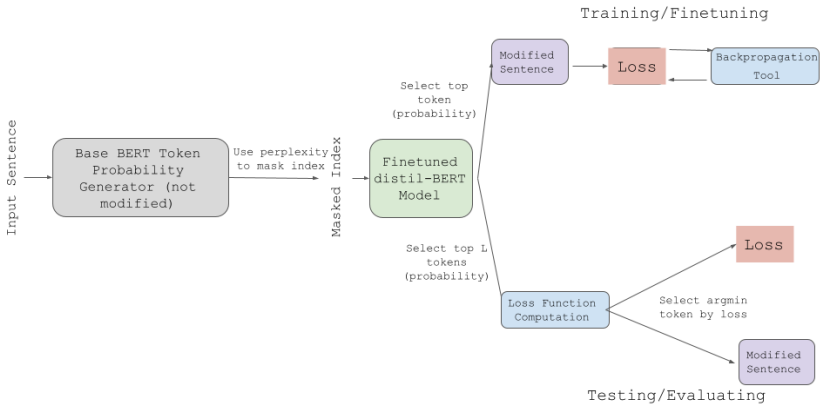

Figure 1: Visualization of Loss Computation

Our model aims to achieve three objectives: improve the perplexity of the text to better emulate human language, maintain grammatical and syntactical coherence, and preserve the meaning of the sentence. Denote N as the length of the modified sentence, S as the original sentence, S' as the modified sentence, w_i' as the i-th word/token of the modified sentence, f as a function which returns the sentence cohesion/grammar score, E_s as the original sentence's embedding, $E_{s'}$ as the modified sentence's embedding, and $P(w_i'|w_{i-1}', w_{i-2}', ...)$ as the probability of token/word w_i' occurring in the i-th position of the sentence given ALL the previous tokens in the sentence. Note that E_s and $E_{s'}$ are generated by Bert's sentence embedding tool [8], $P(w_i'|w_{i-1}', w_{i-2}', ...)$ is generated (for each token) via a pre-trained BERT model from HuggingFace [9], and f is computed via OpenAI's GPT-3.5 model's coherence scorer [10]. None of these listed models are ever modified throughout the project.

Under this definition, the loss function we minimize for is formulated as $-(\alpha l_1 + \beta l_2 + \gamma l_3)$ where l_1 is the perplexity score of the S', l_2 is S''s coherence score, and l_3 is the cosine similarity between the two embedded sentences. α, β, γ are constants we fine tune. The full formulation can be expressed as

$$Loss = -\left(\alpha 2^{-\frac{1}{N} \cdot log_2(\prod_{i=1}^{N} P(w_i'|w_{i-1}', w_{i-2}', ...))} + \beta f(S') + \gamma \frac{E_s^T E_{s'}}{|E_s||E_{s'}|}\right)$$

The only component of this entire formulation (all else is original) that our group did not create is the perplexity score [11]. The mask and fill technique is derived from a research paper by Wu et al[6]. However, all the coding is original and the research paper provided no code. The utilization of masking by the perplexity and multiple objective optimization is original.

3.2 Baseline

Due to the recent motivation of our project, namely GPTZero's recent relase, there exists limited in-depth research in this field, which impairs the full potential of our baseline. Nonetheless, we have implemented two baselines that serve as valuable benchmarks for results comparison

Baseline 1: The first baseline involves utilizing randomly generated text produced by GPT-3.5, which we have detailed in the experiments section, as the final text. Since the "new" sentence is the same as the original sentence, the coherence and similarity will be very high, at the cost of perplexity. We leverage this baseline to assess the loss values/function and evaluate the effectiveness of AI detectors on a simple AI-generated sentence without any alterations.

Baseline 2: The second baseline follows mask-and-fill to modify the sentence, but masks 5 random tokens and fills them with the highest probability token/word that is NOT equial to the original token at that position. In essence, we "find and replace" masked tokens in the sentence based on their highest probability alternative. This baseline tries to maximize coherence and similarity while making more minor changes (taking a high probability token) that can play an impact on perplexity.

These baselines are essential components of our project, as they represent "default" versions of the goal our project aims to complete. Our project generally aims to maximize perplexity while managing the trade-off this can have on coherence and similarity. The first baseline has no perplexity maximization while the second baseline has minor perplexity changing with a small trade-off. Comparing our model with these baselines helps us evaluate the efficacy of our model in managing the trade-off while maximizing perplexity.

3.3 Head Network

The Head Network model that we utilize is built on top of the BERT token probability generator (its on the head of the BERT model). The idea behind this model is to NOT actually finetune the BERT model due to BERT probabilities inherently encoding information about perplexity, coherence, and to a degree similarity.

The probabilities BERT gives for each potential token at an index are going to generally be inversely correlated with perplexity. The higher the probability, the more likely that token is to be at that index given the rest of the sentence, which lowers the perplexity (randomness). Further, BERT's model is trained to give grammatically correct and coherent sentences so tokens with higher probabilities generally correlate with higher coherence/grammar [12].

While the similarity score is not as strongly correlated to BERT's per-token probabilities, there is a considerable number of instances where higher probability tokens are going to indicate higher similarity. Often times, the context of the whole sentence and neighboring words can generally give information as to what word should be in a specific index. In scenarios where the context of the sentence is sufficiently informative, Bert will account for this and will give higher probability to words that maintain stay consistent with this context and thus have higher similarity. However, sometimes there are also ssentences without enough context (such as "I am going to _") and the BERT token probabilities and similarity won't be well correlated.

Utilizing the fact that BERT's probabilities can be informative of each of the three components of our custom loss function, we create the "Head Network". This model/network works with a specific masked index and identifies how which token should fill this index. The input into this model is a tensor of the K largest probabilities of tokens that could fill this index (as given by BERT), where K is a hyperparameter. For any given index, the pre-trained BERT model is utilized to get the probability that each token appears at this index (given the rest of the sentence as the context). Then, the top-K largest probabilities from this are pulled and put into a vector in descending order. This tensor is then sent into the network. The full process can be visualized via figure 2 "Head Network Architecture".

Note that each of the K locations of the output from the model correspond to a single token from the input tensor. Specifically, the 1st location in the output corresponds to the 1st location of the input (which then corresponds to a token) and so on. The network's output value at each location represents an estimate for the loss-ratio if this location's corresponding token was used to fill this mask. The true loss-ratio value is defined as $\frac{loss_{new}}{loss_{old}}$ where $loss_{new}$ is equal to the loss of the sentence after filling this index with the token. $loss_{old}$ is simply equal to the loss value of the original sentence with itself (as done in the baseline). The goal of this network is to estimate these values. During training, the true loss-ratio values are computed and backpropogation is done using mean squared error between the estimated loss-ratio and the true loss-ratio).

During testing, the network is fed in the K largest probability values for a specific masked index. The network outputs the estimated loss-ratio for using each of the corresponding tokens to fill this index. Then, the token corresponding to the highest estimated loss-ratio is selected to fill the masked index. This process is repeated iteratively for the next masks in the sentence.

Unfortunately, this model is extremely computationally expensive as training requires each mask index to be replaced with a token and then the loss re-computed and backpropogated each time. To attempt to achieve some computational efficiency without sacrificing too much model performance, we employed two mechanisms of selecting which of the K highest probability tokens to train. These are $\epsilon-$greedy (commonly found in reinforcement learning) and greedy respectively. Under an $\epsilon-$greedy policy, with probability

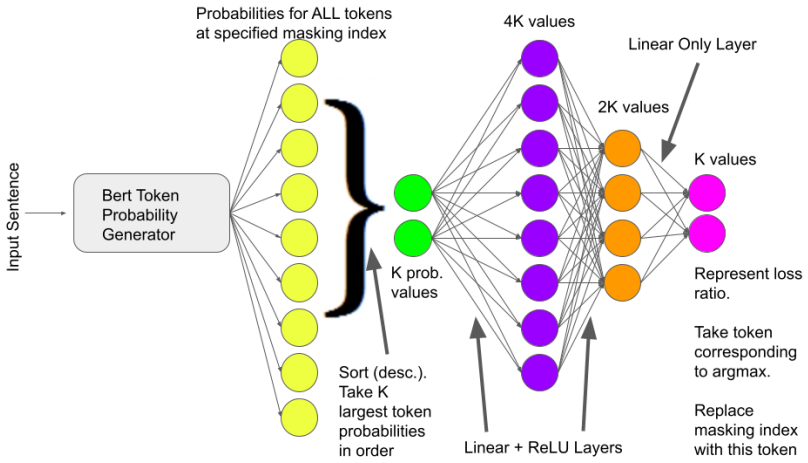

Figure 2: Head Network Architecture

of ϵ (which decays over time through the iterations), there are L (hyperparameter) locations/tokens randomly sampled from the K possible. With $1-\epsilon$ probability, the top L highest locations by currently estimated loss-ratio are chosen. In the greedy approach the top L tokens are always chosen with no randomization. The purpose of $\epsilon-$greedy is to try and explore training/selecting differently ranked tokens (by probability) instead of potentially locking on to some potentially sub-optimal locations.

3.4 Fine-Tuning

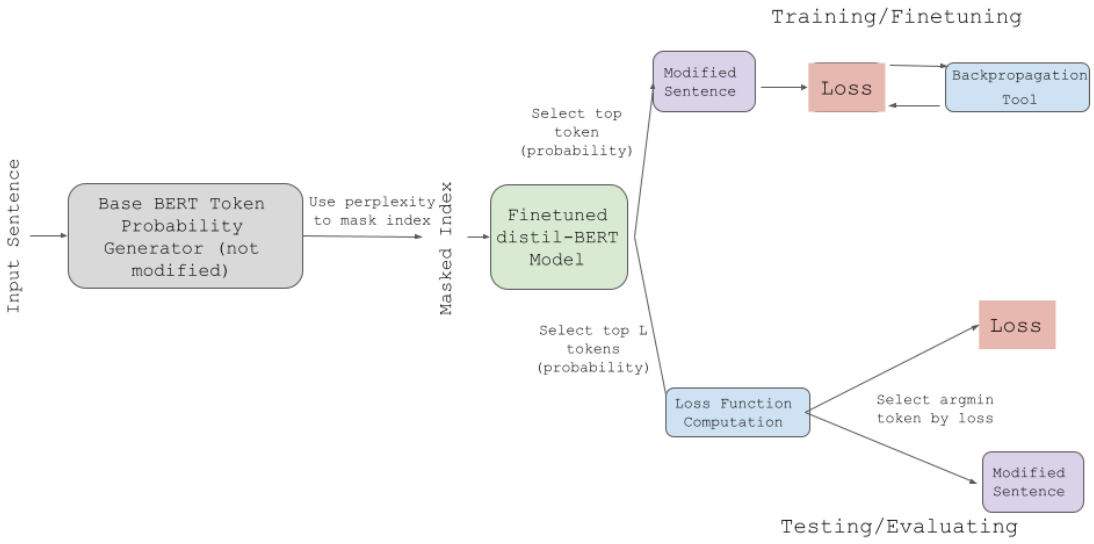

Figure 3: Finetune BERT Architecture/Process

In addition to the Head Network, we work to finetune a pre-trained BERT model learn the custom loss to produce high-coherence, high perplexity mask infills. We hypothesize that this will create lower loss valued sentences than the Head Network model because passing the top K sorted logits (and training on L) into the head network may filter out lower better candidates. In addition, finetuning BERT using the loss function directly will likely give better coherence and similarity than the Head Network as well. Through fine-tuning, we wish to bias the BERT to infill logits towards higher perplexity, higher coherence, and higher similarity candidates before feeding to our selection head.

During training, we fine-tune a DistilBERT model with eight frozen layers using a custom training and masking function. Because we are interested in greedily increasing perplexity, our team only considered replacing low perplexity words (highest potential for change). For each sentence in our training set, we use the Base BERT model to compute the perplexity of each token and mask only the lowest perplexity token.

During fine-tuning, we pass our fine-tuned DistilBERT model the tokenized masked sentences as inputs and the tokenized original sentences as labels. In our trainer, we took the fine-tuned model's output logits and computed the custom loss defined earlier with Base Bert. We then back-propagated the BERT cross-entropy loss (on the selected tokens for masking) scaled by the our custom loss through the last four model layers. In total, we trained over three epochs with the ADAM optimizer and a learning rate of $5e-5$.

Once our model was fine-tuned, to modify a ChatGPT sentence, we first use the Base BERT model to compute the NUM_MASK lowest perplexity tokens. For each token, we mask its index and pass it into our fine-tuned DistilBERT. The logit output of the fine-tuned model reflects the lowest estimated loss value logits . We then pass the top L (hyperparameter) of these logits to a final layer which computes the $argmin$ token based on the loss values if the sentence is masked with this logit/token. We then use this as the new sentence and repeat the process for each mask index to generate the final sentence.

Note the BERT Finetune model and HeadNetwork take two distinct approaches to the searching process of which tokens are viable for infilling. In HeadNetwork there are L tokens that are searched/selected to be trained on whereas only a single token is selected during the test/evaluate. Conversely, for BERT finetuning there is a single token selected during training but there are L tokens selected during testing time to minimize loss over. These models essentially test different approaches to searching during test vs train.

3.5 Data

In this study, we generated data in two steps. Firstly, we downloaded a list of approximately 6800 nouns from Kaggle. Secondly, we utilized OpenAI's GPT-3.5 Model via API to generate three essays for each noun prompt of "Write a long essay about *Noun*". Each essay was saved in its own separate data file, and all data files were stored in a designated data folder. We then randomly split the data into train, development, and test sets using an $80/10/10$ percent split. We generated a total of $\sim 40,000$ AI generated sentences in our data set. .

3.6 Evaluation method

There will be multiple evaluation methods that are utilized. In order to select the best model and hyperparameter we will utilize loss on the dev set and choose the model corresponding to the smallest. Next, we use AI-text-detectors and their returned probability of the text being AI-generated. We will also have a coherence score (generated by OpenAI's GPT3.5 model) and cosine similarity of the two sentences. In addition, we use an ablation study to evaluate results without certain components of the process. Finally, we also conduct a blind study on a group of Stanford students on their ability to tell if a text was generated by either AI or Human.

3.7 Experimental details

We ran our experiments by going training on the training set (a single sentence at a time) for each model and hyperparameters configuration. We then evaluated the performance of each model/hyperparameter set via its loss on the dev set. For both the Head Network and Finetune BERT Model we ran a total of 6 experiments in this format (train on train set, evaluate via dev set loss, and test best model). Due to the extremely high computational cost of parsing sentences and computing the loss function each time, each training run took approximately 20 hours which significantly restricted our ability to run thorough grid search techniques and thorough hyperparameter tuning.

In terms of set hyperparameters, we ran a decaying learning rate starting at 0.1 (using Adam optimizer) and decaying by 0.9 every 100 iterations. In addition, we set $K = 100$ and $L = 10$ where K is the number of top probabilities sent into the Head Network and L is the number of top tokens selected in the models. Finally, we utilized loss weights of $\alpha = 0.5, \beta = 20, \gamma = 20$ after testing on loss values of purely AI-generated sentences.

For the Head Network we ran experiments with the NUM_MASK hyperparameter at values of 3,5,10. For each of these mask counts, we experiment with an $\epsilon-$greedy policy (with decaying ϵ) and normal greedy policy alone. For the Finetuned BERT Model, we ran experiments with the NUM_MASK hyperparameter at values of 3,5,10 and ran experiments with the MASK_CUTOFF value at 0.001, 0.002, and 0.003. This gives the 6 experiments for each model.

4 Results

4.1 Head Network Model

The figure labeled "Custom Loss Values over training" illustrates the training loss for the Head Network and BERT Finetuning Model. As depicted in the graph, the lower bound of the majority of loss values for the Head Network (in general) exhibits gradual logarithmic decline over time, indicating the model's general improvement in minimizing the defined loss function. The significant noise in the data is due to this training process being stochastic. Due to this, each sentence trained on has a loss value on the graph. However, the initial loss values of each sentence before modification have extremely high variance, resulting in the noise.

Table 1 displays experiment results for the Head Network model. The best results (smallest loss) were generated when using 5 tokens to mask AND using an ϵ-greedy policy. Generally, the $\epsilon-$greedy policies performed better than purely greedy policies. In addition, with the NUM_MASK as 3, there simply was not sufficient tokens being modified (especially since 2/3 masks were often first and

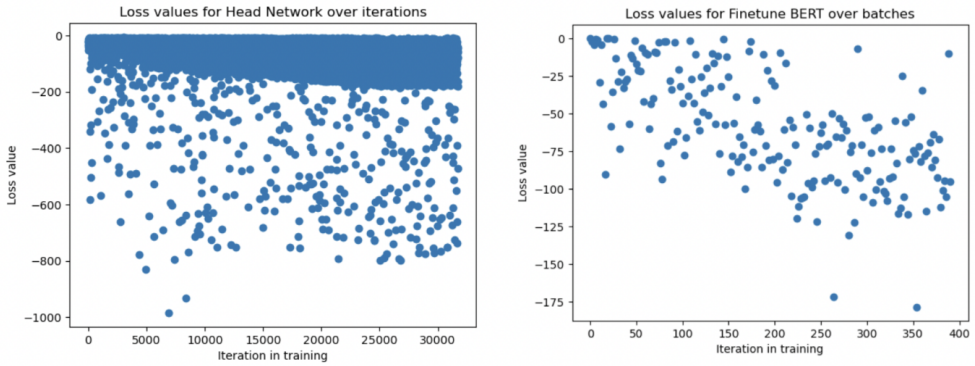

Figure 4: Custom Loss Values over training

Head Network Experiment Results				
NUM_MASK	**Epsilon-Greedy?**	**Avg Train Loss**	**Avg Dev Loss**	**Avg Test Loss**
3	**Yes**	-71.29	-48.23	-45.02
3	**No**	-69.93	-50.19	-48.47
5	**Yes**	-81.19	-60.91	-56.92
5	**No**	-75.42	-53.08	-51.89
10	**Yes**	-76.82	-52.49	-49.14
10	**No**	-68.67	-43.29	-41.46

Table 1: Head Network Average Loss Results for Different Experiments

last tokens), causing low perplexity and high loss. However, masking 10 tokens often caused the similarity and coherence to drop, resulting in higher loss than with a NUM_MASK of 5, which provided a good balance.

4.2 Fine Tuned BERT Model

The figure labeled "Custom Loss Values over training" illustrates the training loss for the Head Network and BERT Finetuning Model. The loss values for the BERT model are computed in batches of 80 sentences each. This results in less noise as the loss values are averaged over all the sentences in the batch. In addition, there is gradual decline amongst a majority of the loss values as they seem to generally cluster/converge around the $(-75, -100)$ range.

The 6 experiments for the Finetune BERT model are displayed in Table 2. Note that utilizing MASK_CUTOFF resulted in loss values and results that were considerably worse than using NUM_MASK as the masking method. DUe to the high variance in probabilities between sentences, it is difficult to get a good MASK_CUTOFF value that successfully generalizes to all sentences (it often masks too little or too much). The BERT model using the NUM_MASK parameter generally experienced similar results to the Head Network when modifying between NUM_MASK values of 3,5,10. However, the BERT model performed using NUM_MASK of 5 was the best performing model of the project.

4.3 Overall Results

We note Table 3 as an overview of the results for the model. We generate the three components of our loss (the three tasks we try to combine) individually to identify which model's perform well in which categories. While the Head Network model consistently scores the least probability of being AI-generated (as evaluated by multiple detectors), it suffers significantly in terms of coherence

Finetuning Bert Experiment Results				
NUM_MASK	**MASK_CUTOFF**	**Avg Train Loss**	**Avg Dev Loss**	**Avg Test Loss**
3	**N/A**	-76.23	-63.07	-62.35
5	**N/A**	-86.24	-68.55	-65.29
10	**N/A**	-83.19	-66.31	-63.11
N/A	**0.001**	-64.27	-51.08	-50.78
N/A	**0.002**	-66.92	-52.27	-52.35
N/A	**0.003**	-67.09	-52.89	-52.61

Table 2: Finetuned BERT Average Loss Results for Different Experiments

	Perplexity	Coherence	Similarity	p(GPT-Zero)	p(DetectGPT)	p(OpenAI)
GPT-3.5 Text	37.166	1.0	1.0	.972	.77	.63
Random Masks	46.950	.54	.582	.724	.53	.35
Finetuned BERT	63.025	.77	.761	.452	.42	.29
Head Network	67.923	.52	.537	.387	.35	.26
Human*	125.320					

Table 3: Mean AI Detectability, Perplexity, Coherence, and Similarity on Test Data
*= *The human perplexity was generated based on 10000 random text samples from research papers by the authors*

	Correctly Classified	Incorrectly Classified
GPT-3.5 Generated Text	**9.45**	**10.55**
Human Generated Text	**10.7**	**9.3**
Model Generated Text	**7.5**	**12.5**

Table 4: Does it pass the Turing Test? Students determining AI or Human Text

and similarity. The Finetune BERT model is able to maintain relatively high coherence and similarity and still achieves a sub-50% probability of being AI-generated. Having a sub-50% average probability indicates that on average the modified sentences are NOT evaluated as AI-generated. As such, due to the lower loss values generated by the BERT Finetune Model in its best experiments, we claim the BERT finetuned model performs better in comparison to the Head Network. In addition, the perplexity values are consistent with the probabilities as the Head Network has the highest perplexity (excluding human). However, it is important to note that the models utilized are still quite different in terms of perplexity when compared with human-text (each having perplexity nearly half that of human).

Finally table 4 gives the results of a study designed to determine whether Stanford students could differentiate between human-generated sentences and sentences generated by the GPT-3.5 language model. We randomly selected a group of 20 Stanford undergraduates and provided them with a list of 60 sentences (20 by humans, 20 by GPT-3.5, 10 modified by BERT Finetuned, and 10 modified by Head Network), asking them to categorize each sentence as either AI-generated or human-generated. Note that students were unable to consistently differentiate between human-generated and AI-generated text, with responses appearing to be largely based on chance. However, it's noteworthy that in some instances, the students mistook text generated by our models for human-generated text. This likely results from our model's generating slightly less coherent text, which is unlike an LLM. Overall, however, humans could not consistently differentiate human-generated text and AI-generated text.

5 Analysis

5.1 Qualitative Results Breakdown

The results section of this research paper highlights the success of the model in generating sentences that were not detected by AI generators as being AI-generated, while also maintaining coherence and meaning to a significant extent.

The results can speak for themselves but isolating some examples from our model can help us understand what our model is computing behind-the-scenes and a great qualitative understanding of the model. Table 5 highlights these results displaying breakdowns of a good, medium, and bad examples from the Fine-Tuned BERT Model. The good example as you can see changes "gets workers" to "recruits recruits" because the use of a word twice in different contexts has a high increase in perplexity. Secondly, changing "higher-paying" to "good-paying", although less formal, decreases the formality of the sentence increasing the perplexity. Overall this is a good example of our model's successes as it shows the coherence stays similar (1 vs .9) while changing the sentence and fooling the detector (.985 to .261, perplexity goes from 15.33 to 57.00). The next sentence shows an example where the model performs as intended/expected without necessarily great results. The sentence does lose some coherence as it makes less sense (coherence from 1 to .7) and the meaning has slightly changed; however, it is clear what the model is trying to do. It takes the low perplexity word (for the sentence) "famous" and changes it to "German" a much higher perplexity word, while still relating to the sentence as the creation of the Bicycle was in Germany. It also changed "economically" to "linguistically" which makes less coherent sense but increases the perplexity as linguistically is less commonly used than economically. This is clearly shown in the perplexity increase (20 to 44) and the AI detectability from 98.1% to 28.2%. Overall, this is a decent example showing the overall performance of the model. The last example is what we consider a "bad example." Although we brought up our perplexity by a factor of 530% which is an astronomical increase, we sacrificed our coherence (from 1.0 to .2). This example also shows a major limitation of the model. When there is a low perplexity subject i.e, in this case, automation, it will want to change it to alcohol to allow for higher perplexity, but this could fundamentally change the meaning of the sentence which is difficult to punish by our loss function. Balancing loss hyperparameters was primarily based on the "average" situations, which can be skewed by outliers. In this case there was still a small coherence

Sentence	Perplexity	p(GPT-Zero)	Coherence
Initial: It gets workers with the necessary skills and knowledge to enter the workforce and can open the door to higher-paying jobs and advancement opportunities.	15.33	.985	1.0
Changed: It recruits recruits with the necessary skills and knowledge to enter the workforce and can open the door to good-paying jobs and career growth.	57.00	.261	.9
Initial: The famous bicycle has had a profound impact on the world, both culturally and economically.	20.00	.981	1.0
Changed: The German bicycle has generated a profound impact on the world, both culturally and linguistically.	54.00	.282	.7
Initial: Automation has brought about many benefits, such as increased productivity, improved safety, and cost savings.	16.00	.991	1.0
Changed: Alcohol has testified about many benefits, expressed as increased productivity, improved safety, and money savings.	101.01	.051	.2

Table 5: Model Sentence Analysis

	Coherence	Similarity	$p-$ GPT-Zero	$p-$ DetectGPT	$p-$OpenAI
Baseline 1	**1.0**	**1.0**	**.972**	**.77**	**.63**
Fine Tune BERT	**.73**	**.683**	**.452**	**.42**	**.29**
Fine Tune BERT $\alpha = 0$	**.95**	**.834**	**.965**	**.75**	**.65**
Fine Tune BERT$\beta = 0$	**.67**	**.725**	**.352**	**.36**	**.23**
Fine Tune BERT $\gamma = 0$	**.82**	**.470**	**.335**	**.34**	**.21**

Table 6: Ablation Studies

contribution to the loss and the model still produced this output which shows how there is a bias to increase perplexity. Overall this analysis shows a pretty good example of how the model works and gives us a qualitative understanding of what our model does.

5.2 Ablation Study

However, to supplement this analysis, we propose an Ablation study that involves masking out components of our custom loss function. Specifically, we reran our model and set α, β, and γ to 0 in three respective iterations to determine which component was the most valuable in fine-tuning and fine-tuning the BERT model. This additional analysis will provide a deeper understanding of our model's underlying mechanisms and further enhance the overall contribution of our research. Table 6 presents the results of this study. Each component performed its intended task without any major surprises. The authors observed that when they removed perplexity from the equation, the model returned to the initial baseline as the sample sentences from GPT-3.5 had very high coherence and was very similar original sentence. When similarity was removed, coherence increased while the likelihood of being detected by AI decreased. Additionally, removing coherence from the loss function only reduced the model's coherence by about 6%, while everything else remained relatively similar. This surprising result's hypothesis was attributed to BERT taking into account the coherence of a sentence when generating token distributions. With limited computation power, it may be worthwhile to remove coherence all-together. Generally, the similarity and perplexity loss components were the most critical components, with perplexity loss making the model undetectable by AI and similarity maintaining the sentence's closeness to the initial sentence by a significant margin. In addition, the probability of detection drops greatest when removing similarity, indicating this is the most constraining factor (for our model) to reducing detection even further.

6 Conclusion

The project, overall, exceeded our expectations and did a great job of Fine-Tuning BERT to change sentences to fool AI detectors (up to a 60% drop in probability of being detecting) while keeping the sentence similar in meaning (tested via cosine-similarity) and coherent. This follows the hypothesis of using perplexity to distinguish between the 2 languages i.e AI and Human and then using mask-encoding to transfer the meaning by implementing a head-network and finetuning a BERT model. In the future we aim to mask more than single words at a time and convert into phrase masking, which could help with coherence. Additional future work is to try using a different loss function which could optimize for more components than just perplexity, coherence, and similarity (and maybe find a better way to test for similarity), use different features from sentences in combination with perplexity as highlighted by paper 2, combining both our Head-Network in conjunction with out BERT-Fine Tuned model as together they can help improve each others faults, and finally we could also explore a direction where we use Double Deep Q-Networks in combination with this Fine-Tuned version of BERT and model this problem in a "game" format.

References

[1] Edward Tian. Gptzero. 2022.

[2] Eric Mitchell. Detectgpt: Zero-shot machine-generated text detection using probability curvature. 2023.

[3] OpenAI. Openai gpt-3.5, 2022.

[4] HuggingFace. Bert finetuning, 2022.

[5] et al. Ruixiang Tang. The science of detecting llm-generated texts, 2017.

[6] Liangjun Zang Jizhong Han Xing Wu, Tao Zhang and Songlin Hu. Mask and infill: Applying masked language model to sentiment transfer. 2019.

[7] et al. Pablo Gamallo. A perplexity-based method for similar languages discrimination. 2017.

[8] Boris Power Joanne Jang Arvind Neelakantan, Lilian Weng. Text and code embeddings by contrastive pre-training, 2022.

[9] Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. Bidirectional encoder representations from transformers, 2022.

[10] Nick Ryder Tom B. Brown, Benjamin Mann. Openai grammar correction, 2018.

[11] Chiara Campagnola. Perplexity in language models. 2020.

[12] Evaluating Document Coherence Modeling. Aili shen, meladel mistica, bahar salehi, hang li, timothy baldwin, jianzhong qi. 2021.

A Appendix (optional)

All diagrams are created by the researchers (authors). If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the 6-8 page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc., that you couldn't fit into the main paper. However, your grader *does not* have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.

A.1 Full Literature Review

A.1.1 Research in AI detection

The first area of interest is how AI detectors detect for AI-generated text. One approach is to use perplexity-based techniques that assess the likelihood of a given word following the previous word, evaluating likelihood via a model such as GPT, BERT, XL-NET, etc. Variance in perplexity is also taken into account in some cases. Another method involves analyzing the features of the text, such as sentence structure, grammar, and syntax. Machine learning techniques, such as support vector machines (SVMs) and deep neural networks, have been employed to improve the accuracy of AI detection models. Applications of such software include detecting fake news and identifying social media bots. One noteworthy paper presents a detailed analysis of how Rice University researchers developed a methodology to detect AI-generated text. The authors propose using linguistic and statistical features to distinguish texts generated by Language Models (LLMs) from those written by humans. The proposed methodology comprises of the following steps:

1] Feature extraction: A set of linguistic and statistical features is extracted from each text, including measures of syntactic complexity, lexical diversity, and coherence.
2] Feature selection: A feature selection algorithm is employed to identify the most discriminative features, which are likely to differ between LLM-generated and human-written texts.
3] Classification: A machine learning algorithm is used to classify each text as either LLM-generated or human-written based on the selected features.

To evaluate the methodology, the authors conducted a series of experiments on a dataset of 1,000 texts, half of which were generated using GPT-3.5 and half of which were written by humans. The results indicate that the proposed methodology achieves high levels of accuracy, with a classification accuracy of 95.1% and an F1 score of 0.95 while keeping feature selection fairly small. The limitations of the study, however, include a small dataset of only 1,000 texts that may not be representative of all AI-generated texts, no exploration of different LLM models or training settings, focus only on distinguishing LLM-generated from human-written texts, and lack of detailed analysis of classification errors.

A.1.2 Sentiment Transfer

Another area of interest we had was how to slightly modify a sentence to change a feature (perplexity in this case). This led us to investigate sentiment transfer, which involves changing the sentiment of a sentence while preserving its meaning. The paper we read was Mask and Infill: Applying Masked Language Model to Sentiment Transfer.
The authors' objective is to address the problem of sentiment transfer, which involves generating a new sentence with an altered sentiment while retaining the original meaning and content as much as possible. They aim to create a model that can automatically perform this task, as humans can naturally modify a sentence's sentiment while maintaining its meaning. The paper offers several significant contributions to the ongoing research on sentiment transfer. Its primary objective and most innovative contribution is to minimize changes to a sentence while converting its sentiment from positive to negative or vice versa. Unlike previous research, which often involves entirely rephrasing or transforming a sentence, this paper strives to preserve as much of the original sentence as possible, modifying only the necessary words and phrases to achieve the desired sentiment. The paper does this via the mask-and-fill technique, which masks important words and phrases related to the sentence's sentiment and then modifies them accordingly.
Although the paper presents promising results, there are some limitations to the proposed method of sentiment transfer. Firstly, the evaluation is limited to only two sentiment transfer datasets, Yelp and Amazon, which may not be representative of other sentiment transfer tasks, making it unclear whether the approach can generalize to other types of sentiment transfer tasks beyond customer reviews. Lastly, the datasets used for evaluation are imbalanced, with more positive examples than negative, which may bias the results towards positive sentiment transfer.
Despite these limitations, the techniques utilized in the paper are promising, and further exploration of these sentence-generation techniques in broader contexts is warranted. The main reason for selecting this paper was its relevance to the group's interest in changing features of a sentence while preserving its meaning and structure. As such, technique utilized in the project of mask-and-fill is derived from this paper.

A.1.3 Perplexity Analysis

Next, paper "A Perplexity-Based Method for Similar Languages Discrimination" by Pablo Gamallo, Jose Ramon Pichel, and Iñaki Alegria, published in the Journal of Quantitative Linguistics in 2017, proposes a novel approach to discriminate between similar languages based on perplexity. The paper addresses an important challenge in natural language processing (NLP) and computational linguistics: how to distinguish between languages that share many commonalities, such as Spanish and Portuguese or Dutch and German.

The authors propose perplexity as a discriminative feature to differentiate between similar languages. They argue that similar languages should have similar perplexities, while less similar languages should have more distinct perplexities. The authors proposed methodology was as followed: They collected a corpus of text from four pairs of similar languages: Spanish and Portuguese, Dutch and German, Swedish and Norwegian, and Catalan and Occitan. They used a language modeling technique based on n-gram models to estimate the perplexity of each language. They then calculated the pairwise Euclidean distances between the perplexity vectors and performed a cluster analysis to determine the ability of the perplexity-based method to discriminate between the languages. The results of the study showed that the perplexity-based method was able to accurately discriminate between the four pairs of languages with an accuracy rate of over 90%.

This paper helped motivated the increase of perplexity to help fool many of these AI detectors. This study proved that perplexity could be a method to distinguish between similar languages, in our case we could use it to distinguish between our "AI-generated" language and human language (as they might have many of the same features). Taking motivation from the paper and that detectors like GPT-Zero already utilize perplexity, we attempt to maximize perplexity of our modified data.

A.1.4 Research Application in Project

We learned from the first paper that there are particular strong features in sentences that could help distinguish AI-generated text and human-generated text. Then the third paper, discussed how perplexity is a major feature in many languages and can help distinguish between various similar languages. From there, we chose to use perplexity as the feature we wanted to manipulate as it could then it could help fool many of these models that try to detect AI. However, we wanted to change the perplexity of a sentence while keeping the sentence meaning the same, this led to us discovering the sentiment transfer paper as it took a separate feature (sentiment) and changed the sentence based on that feature while preserving meaning and coherence. This brings us to our approach.