Absolute Mediocrity

Odyssefs Drys-Pentzakis

***Abstract*—The study explores the generation of counterfactual explanations to induce uncertainty in Natural Language Processing (NLP) models by altering already existing counterfactual editors. Counterfactuals are commonly used to make the minimum changes in an input so that a model's prediction changes, however we look into the production of counterfactuals that maximize model uncertainty. Instead of preserving changes that shift classification probabilities from one extreme to another (e.g., from 0.1 to 0.9), we retain alterations that move the probability towards the middle (e.g., from 0.1 to 0.5). We inspect the MiCE and TextFooler counterfactual editors with the intention of customizing them to align with our specific objectives. We identify upper and lower limits for the percentage of the original input that we should modify to achieve these counterfactuals and investigate how variations in the production process impact our findings. We assess the model's predictions on the counterfactuals, differentiating between scenarios where the model correctly refrains from confidently classifying the sample due to ambiguity and situations where the sample's class is clear, yet the model exhibits uncertainty. This approach allows us to evaluate the model's reliability and gain insights into its internal mechanisms. Future work includes refining editor modifications, exploring different counterfactual editors, extending applications to various NLP tasks, and integrating calibrated models for improved reliability.**

I. Introduction

In recent years, the field of Explainable Artificial Intelligence (XAI) has emerged as a critical aspect of advancing machine learning and AI technologies. As machine learning models become increasingly complex and sophisticated, the need for transparency and interpretability in their decision-making processes becomes paramount. The opacity of black-box algorithms can hinder user trust, limit widespread adoption, and pose ethical concerns. Explainable AI seeks to bridge this gap by providing insights into the inner workings of these models, enabling users to understand how decisions are reached and facilitating the identification of biases or errors. Beyond fostering trust and accountability, the importance of Explainable AI extends to various industries, including healthcare, finance, and autonomous systems, where clear and interpretable decision-making is crucial.

Several methodologies have been developed within the Explainable AI framework to shed light on the decision-making processes of complex models. One noteworthy approach is the use of counterfactual explanations. Counterfactuals provide users with alternative scenarios that could have led to a different outcome, offering a tangible and comprehensible way to understand model predictions. By presenting instances where a slight change in input features results in an altered decision, counterfactuals not only enhance interpretability but also aid in identifying the features that significantly influence model outputs.

In this study, our objective is to generate counterfactuals for Natural Language Processing (NLP) models that, instead of altering the model's prediction, induce uncertainty in the model. Typically, a model, when provided with an input, generates a probability distribution reflecting the likelihood of the sample belonging to a specific class. The predicted class is the one with the highest probability in this distribution. In the context of binary classification, such as distinguishing between positive and negative reviews, two probabilities are produced, summing up to 1, with the larger of the two dictating the model's prediction. The model's confidence in its choice increases as the probability of a sample belonging to a particular class approaches 1. For instance, a prediction of 0.9 indicates greater confidence than a prediction of 0.6.

Existing counterfactual approaches typically seek to maximize the disparity in the model's output with minimal alterations. These methods essentially aim to make the classifier predict the contrasting class of the original input by editing it, striving to alter the probability of the sample belonging to a class as significantly as possible.

A very confident prediction is very valuable; when correct it affirms the model's reliability, and when wrong it sheds light to the model's shortcomings, but not all cases are black and white. To align models with human-like decision-making processes, it is crucial to acknowledge the pivotal role of uncertainty in these decisions. Therefore, it becomes essential not only to examine the factors contributing to a model's uncertainty but also to assess its performance when faced with ambiguous inputs. We say that the model is uncertain when it cannot decide which class a sample belongs to, outputting a classification probability around 0.5. Understanding the specific edits that lead to counterfactuals exhibiting this kind of uncertainty can provide valuable insights into the semantic boundaries within the model. This, in turn, allows for a more refined evaluation of the model's reliability and its ability to handle ambiguous scenarios.

To create a method that produces counterfactuals that maximize the uncertainty of the model in the prediction of its classification we require research in two topics. First, the topic of **Counterfactual Explanations** in NLP, what approaches have been made to generate them, and what challenges they face to provide valuable explanations and insight about the model. Next, since we base our definition of uncertainty on how close the prediction is to 0.5, the prediction should represent the actual likelihood of a sample belonging to a class. In order to achieve that, we will research the topic of **Calibrated Models**, models that go beyond traditional accuracy measures to ensure that the confidence of a model in its predictions aligns with its actual performance.

Learning which changes to the original input cause the model to be uncertain can offer great insight in gauging its reliability and robustness. Simply knowing what modifications cause the model to be wrong is insufficient, we also want to know its behavior in ambiguous situations and define where the semantic borders blur. For instance, in automated medical diagnostics, an evaluator might want to know not just when a model changes its diagnosis, but how it behaves when a diagnosis isn't clear-cut. Counterfactuals that maximize model uncertainty can be used to augment datasets and retrain models, making them better at discerning classes when the input is not clear-cut, but also making them more confident in cases where they are uncertain but the input's class is evident. This way not only do we improve the model's accuracy, we also achieve a more human-like decision making process. For example, also in a medical diagnosis setting, if the necessary examinations are not made an evaluator should not make a diagnosis prematurely, but it should express a lack of confidence indicating that the information it is provided is lacking.

II. Related Work

Counterfactual explanations are widely used in the field of XAI since they provide valuable insights into the inner workings of the model and what influences its prediction. We find that opening the black box of the model and having access to its parameters and structure to understand it better is ineffective since the more advanced the models become the harder that becomes. An effective way counterfactuals are generated is through adversarial methods, having a generator alter the original input and try to make the model misclassify the altered input, which does not require opening the black box ([12]). John Morris et al. [8] then broke down adversarial attacks in NLP into 4 components, the goal function we aim to accomplish, the constraints that the edits are under, the way we transform the input and the search method we use to find the right transformed inputs. We find that in many other methods for generating counterfactual explanations, we can discern these components and this breakdown of the process is very helpful in producing a novel method, such as the one we are aiming to make. The goal function of a counterfactual explanation is rarely a single objective, usually being objectives such as minimizing the semantic difference between the original input and the counterfactual and maximizing the confidence score of the prediction. Susanne Dandl et al. [3] propose a way to attain counterfactuals that accomplish the different goals set for them by evaluating each counterfactual for each goal and returning sets of counterfactuals that each have different trade-offs. There are a lot of constraints that a counterfactual explanation for NLP has, some being of course that the input should be grammatically and semantically correct, and Poyiadzi et al. [9] added to these constraints that the explanations should be a feasible and actionable edit of the original input, a change that does not comply with this restraint is useless in a practical setting.

Many editors for generating counterfactual data have been developed, each of them having different methods and approaches. Since we are also aiming to build a counterfactual editor, each of them provides great ideas and a guideline for how we will work. TextFooler [7] was introduced in order to test how robust the model BERT is. It first computes the importance of each word in the sentence by calculating the prediction change before and after deleting the word. Then it ranks the words based on importance, finds the top N synonyms for each word, and replaces the word with the synonyms. The goal is to change the prediction of BERT by changing a word from a sentence with a synonym while maintaining semantic similarity between the original sentence and the altered sentence, so for each candidate that changes the prediction, we keep the candidates that have the highest semantic similarity. If there is no such candidate that changes the prediction we move to the next word until we alter the prediction. It works by computing the importance of each word in a sentence, replacing the words, from the most important to the least important, with synonyms, and checking if the prediction of BERT has changed. The MiCE [10] editor tries to make minimal edits for the prediction of the model to change, first, the editor is fine-tuned to infill masked spans of text in a targeted manner, and the fine-tuned editor masks a percentage of the tokens in the sentence, doing a binary search to find the optimal masking percentage. Then a model generates candidates for the masked tokens for which a prediction is made and the candidates with the highest contrast predictions are stored. It does so for every word in the sentence until a maximum number of edits is reached. Polyjuice [13] works by generating a set of counterfactuals with various relationships regarding the semantic change between the original instance and the counterfactual. The generation of counterfactuals is framed as a conditional text generation task using Language Models, and Polyjuice is trained by finetuning GPT-2 through a proposed prompting method. Another editor, DoCoGen [1] produces counterfactuals that change the domain of the input, for example from the domain of kitchen products to electronic products. The generation of domain counterfactuals starts by masking domain-specific terms in the sentence and then a generative model reconstructs the sentence. This editor shows how predictions are affected by the domain of the input.

These approaches at counterfactual generation are evaluated by Filandrianos et al. [4] based on how often the output of a predictor flips to the desired class, fluency of the generated sentence and how minimal the changes were. However, because these metrics require having multiple editors to compare for them to be able to show which counterfactual is optimal, this paper introduces an inconsistency metric that can evaluate the consistency of the counterfactuals produced without the need for comparison. To further analyze these editors they propose a back translation-inspired evaluation methodology that iteratively feeds the editors back with their output, obtaining valuable insights into the behavior of both the predictor and the explainer models.

For our application that depends on the numerical value that the model outputs for each class, Calibrated Models are required for having an accurate estimation of how likely the prediction of the model is correct. Satthar et al. [11] introduce a new metric, the calibrated f1-score that takes into account the calibration of the system penalizing it when it is overconfident. Jagannatha et al. [6] then propose a calibration schema that calibrates the output by creating a set of positive events, which is a sequence of labels that the model can predict with high confidence, using binary class calibration methods to calibrate the confidence score of the positive events and combines the calibrated confidence scores of each positive event to end up with a calibrated confidence score for the whole input. Lastly, Chen et al. [2] first dispute the opinion that models calibrate themselves through training with the fact that as the more they are trained, their confidence increases even in cases where the prediction is wrong. Then they evaluate existing calibration methods, by dividing them into unlearnable and learnable methods, respectively methods that alter the confidence score after it has been produced, and methods that gather data directly and train the model. Learnable methods are proven to be superior to unlearnable, and through this paper, further insight is gained into calibration methods.

With the topic of the study being uncertainty in NLP, we also found that [14] proposes novel methods to study the benefits of characterizing model and data uncertainties and shows that explicitly modeling uncertainties is not only necessary to measure output confidence levels, but also useful at enhancing model performances in various NLP tasks. In addition, [5] provides a comprehensive review of uncertainty-relevant works in the NLP field and categorizes the sources of uncertainty in natural language into three types, including input, system, and output.

III. Methodology

A. Inspecting the Editors

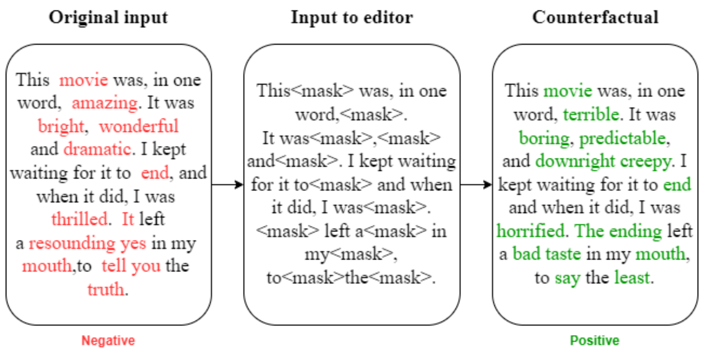
In order to produce counterfactuals that maximize model uncertainty we will build upon the MiCE [10] and TextFooler [7] editors.

Initially, we will examine the MiCE editor, which operates by taking an input and masking a percentage of it across various masking percentages. It employs a binary search to determine the optimal masking percentage, generates candidates using a generative model to fill the masked tokens, and conducts a beam search to identify candidates with the highest contrast predictions, subsequently storing them. In contrast, TextFooler adopts a different approach. It begins by assessing the importance of each word in the input through a process of removing the word and evaluating the resultant change in the model's prediction. Subsequently, for each word, TextFooler extracts synonyms and substitutes the word with them until the model predicts the desired contrast label.

MiCE and TextFooler differ in the way they produce counterfactuals. MiCE produces counterfactuals that may have different sizes than the original input since it uses a generative model. At the same time, TextFooler keeps the input size intact whilst only changing words with their synonyms.

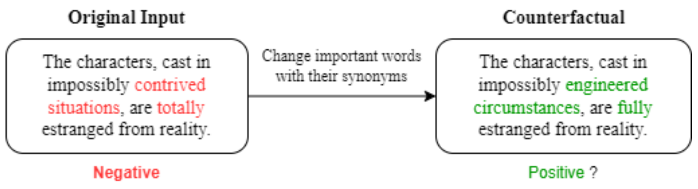
B. Modifications of the Editors

For our task, we will modify the editors to return counterfactuals that have a probability of belonging to the contrast class as close to 0.5 as possible (it does not matter for which class we make the model predict since we aim for uncertainty)



Original input
Input to editor
Counterfactual
This movie was, in one word, amazing. It was bright, wonderful and dramatic. I kept waiting for it to end, and when it did, I was thrilled. It left a resounding yes in my mouth,to tell you the truth.
This<mask> was, in one word,<mask>. It was<mask>,<mask> and<mask>. I kept waiting for it to<mask> and when it did, I was<mask>. <mask> left a<mask> in my<mask>, to<mask>the<mask>.
This movie was, in one word, terrible. It was boring, predictable, and downright creepy. I kept waiting for it to end and when it did, I was horrified. The ending left a bad taste in my mouth, to say the least.
Negative
Positive

Fig. 1: Example of MiCE generating a counterfactual



Original Input
Counterfactual
The characters, cast in impossibly contrived situations, are totally estranged from reality.
Change important words with their synonyms
The characters, cast in impossibly engineered circumstances, are fully estranged from reality.
Negative
Positive ?

Fig. 2: Example of TextFooler generating a counterfactual

[1]. To modify **MiCE** we will target the parts of the code that define how the edits are chosen and evaluated.

The areas in which we can intervene are the criteria score the candidates, from scoring them based on how close they are to the contrast label to scoring them based on how close they are to 0.5, and how the candidates are sorted, again with the same criteria as before. These changes will have little to no effect in the performance of the procedure and the computational power needed, but will have the biggest impact in the counterfactuals produced. We can also intervene in the upper and lower bounds of the masking percentage, to observe if there is an optimal percentage specifically for our task. We would like to avoid high masking percentages (over 60% of the original input) since we will retain too little of the original input and the minimality of the counterfactual will suffer, but we will also avoid very small masking percentages since we might have a hard time making significant changes. Finally, we can experiment with the maximum search levels, and when the procedure that finds edits ends, but with risks of the performance dropping in case the procedure has a hard time finding the desired counterfactuals.

To modify **TextFooler**, we could target the segments that choose which produced input to keep based on how close to 0.5 is the probability that the predictor gives and. We could also experiment with not only changing the words with the highest importance scores but also with words that are not so important, in order to see if less important changes make the model more uncertain. However, because of limited resources, **we were not able to experiment with TextFooler.**

The uncertainty of the model cannot be gauged if the probabilities produced by the model do not represent the actual likelihood of the sample belonging to a class. To accomplish this the model should be calibrated, which means that part of

[1]The code used is available at: https://github.com/OdysseasDrys/AbsoluteMediocrity

our procedure is to embed a calibrated model in the editor. Unfortunately, the MiCE editor uses its own predictor and leaves no room for modifications in the model, so we are unable to use a calibrated model for our predictions.

IV. Experiments

All the experiments were done on the IMDb dataset with the MiCE editor that uses its own custom predictor, which is finetuned to better fit the dataset distribution. Each experiment was run for 20 samples of the dataset, but the number of counterfactuals produced depends on the maximum number of search levels and edit rounds. We calculate the number of counterfactuals produced with:

$$\#\text{Counterf.} = \text{Edit Rounds} \times \text{Search Levels} \times \#\text{Inputs} \quad (1)$$

We experimented with the way the probabilities are sorted, the lower and upper bounds of the masking percentage, and the beam search. We also changed the condition that the code follows for stopping the search for a counterfactual, from the model predicting the contrast label to the model predicting with a probability between 0.4 and 0.6.

The MiCE editor takes in as parameters the maximum masking percentage, the maximum edit rounds, the maximum search levels, and the beam width of the beam search. In order to examine how these parameters affect the production of counterfactuals that maximize the model uncertainty we will run the editor for different parameters. Outputs with a probability between 0.4 and 0.6 will be classified as uncertain predictions.

M. E. R.	M. S. L.	B. W.	# Unc. Preds.	% of Unc. Preds
3	4	3	36	15%
3	3	3	33	18%
3	5	3	52	17%
4	4	4	28	8%
2	2	3	19	23%

TABLE I: Number of Uncertain Predictions for Different Parameters and Percentage of Counterfactuals that Produced Uncertain Predictions for 20 inputs of the IMDb Dataset. Columns are Max. Edit Rounds, Max. Search Levels and Beam Width respectively.

We can observe from Table 1 that the deciding factor for producing uncertain predictions is the maximum search levels parameter, which is to be expected since the editor has more attempts at finding the desired counterfactual. However, increasing the search levels also greatly increases the runtime of the editor (row number 3 of Table 1 took 7 hours to complete for 20 inputs!).

Certain inputs, regardless of the editing parameters employed, proved challenging to generate counterfactuals that maximize model uncertainty, consistently residing at the extreme boundaries of the classification. Conversely, some inputs demonstrated a heightened susceptibility to modification, illustrating the substantial influence of the text's structure and length on our outcomes.

Observing the masking percentages that produced uncertain predictions from Fig. 3 we can see that small masking percentages often do not change the original input enough for the model to change its prediction. In addition, big masking percentages do not only fail to produce uncertain predictions but they also change the original input far too much for the counterfactual to have value for the explainability of the model.

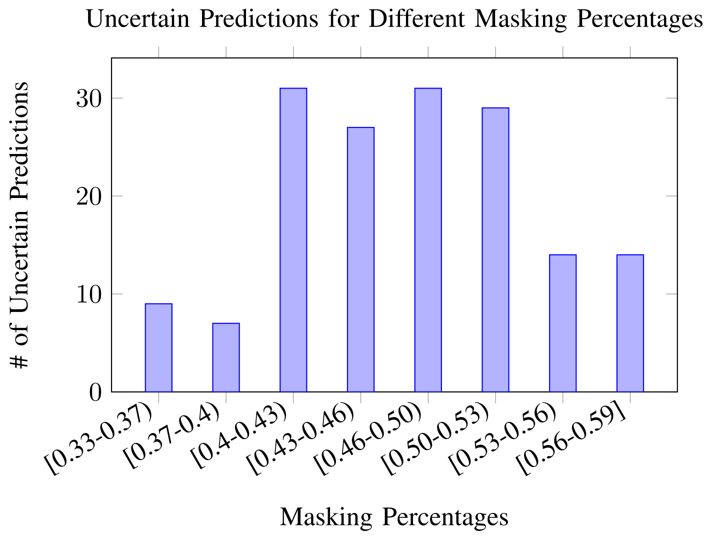
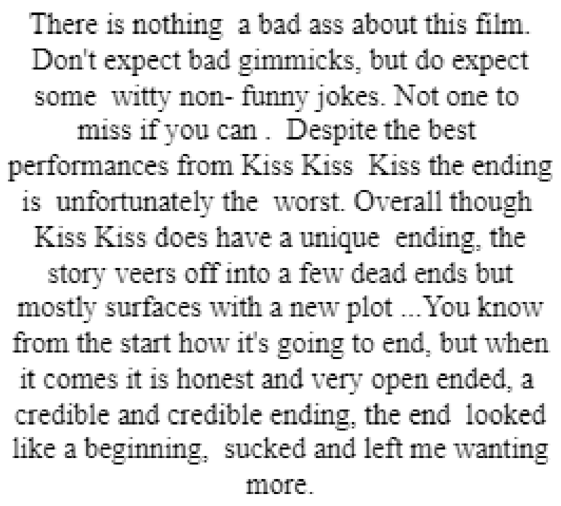


Fig. 3: Distribution of uncertain predictions across different masking percentages made from the MiCE editor. This contains all the counterfactuals produced from the 5 experimental setups from Table 1. Each bar represents the number of uncertain predictions for a specific range of masking percentages.

V. Evaluation

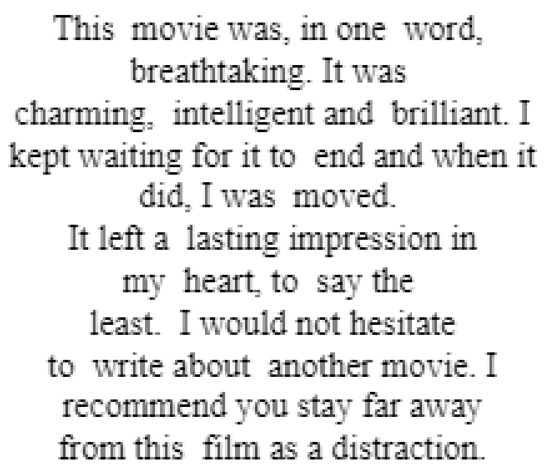
We examine the outputs of the MiCE editor [10] that caused uncertain predictions so that we can see in which cases the output is ambiguous and the model correctly gives predictions around 0.5, and in which cases the output is clear but the model does not give a clear answer. We find that most edited inputs are not clear-cut, but there are cases in which the prediction should be certain and the model fails to classify it.

The majority of the counterfactuals aimed at maximizing model uncertainty exhibit ambiguity, and those with a masking percentage below 0.5 maintain their linguistic fluency. As the editing of inputs advances, there is a noticeable shift in the text's structure, with the introduction of additional spaces between words and a more abrupt, less naturally flowing expression. Despite this, a discernible pattern emerges during the evaluation of the generated review, indicating a trend that leads to uncertainty in the predictor. The editor adheres to a "human-like" approach to altering the expression of an opinion, opting for modifications in how the movie is characterized rather than subtle changes in wording that might cause confusion as shown in [7]. We give examples of the two different cases where the model gives predictions very close to 0.5.



There is nothing a bad ass about this film. Don't expect bad gimmicks, but do expect some witty non- funny jokes. Not one to miss if you can . Despite the best performances from Kiss Kiss Kiss the ending is unfortunately the worst. Overall though Kiss Kiss does have a unique ending, the story veers off into a few dead ends but mostly surfaces with a new plot ...You know from the start how it's going to end, but when it comes it is honest and very open ended, a credible and credible ending, the end looked like a beginning, sucked and left me wanting more.

Fig. 4: Movie Review Produced by the MiCE Editor that is Ambiguous even for Human Reader and also for the Model



This movie was, in one word, breathtaking. It was charming, intelligent and brilliant. I kept waiting for it to end and when it did, I was moved. It left a lasting impression in my heart, to say the least. I would not hesitate to write about another movie. I recommend you stay far away from this film as a distraction.

Fig. 5: Movie Review Produced by the MiCE Editor that is Evidently Positive but Confuses the Model

In Figure 4, it becomes evident that employing a pattern of alternating between negative and positive adjectives when describing the film or its various aspects results in a review that not only perplexes the model but also confuses human readers. The editor successfully generates counterfactuals that maximize model uncertainty through the strategic use of contradictions, mirroring a distinctly ”human” approach to introducing ambiguity into a classification.

However, in Figure 5, the review is primarily positive, saying how intelligent and brilliant the movie was etc. but the concluding phrase, though not entirely negative as it suggests avoiding the film for those seeking distraction, is sufficient to introduce uncertainty into the model's prediction.

Interestingly, the editor consistently avoided employing descriptors like ”mediocre,” ”not good but not bad,” or ”average,” which typically leave a human reader in a state of indecision regarding a movie's quality. While this pattern could be attributed to the editor's pretraining focus on generating counterfactuals for predicting the contrasting label of the original, the chosen approach appears more intuitive. Contradictions introduced by the editor seem to effectively induce uncertainty in the reader, aligning with the goal of maximizing model uncertainty.

The effectiveness of our edits was significantly influenced by the length and structure of the inputs. Longer and more intricate reviews exhibited heightened volatility in predictions when edited, whereas shorter and clearer reviews were more easily modified to induce confusion in the model.

Upon evaluating the model's performance with the generated counterfactuals, we conclude that the model demonstrates moderate calibration. This assessment is based on the observation that none of the outputs resulting in uncertain predictions exhibited extreme negativity or positivity, with most of them being ambiguous and confusing.

VI. Conclusion

In this study, we explored the generation of counterfactual explanations to induce uncertainty in Natural Language Processing (NLP) models. The motivation behind this research was to understand how models exhibit uncertainty in their predictions and to identify factors contributing to ambiguous outputs. By modifying existing counterfactual editors, specifically building upon the MiCE and TextFooler editors, we aimed to produce counterfactuals that maximize model uncertainty rather than changing predictions.

Our experiments with the MiCE editor on the IMDb dataset revealed that the maximum search levels parameter significantly influences the production of uncertain predictions. Higher search levels allow the editor more attempts at finding the desired counterfactual, but also increase runtime substantially. We observed that small masking percentages often fail to change the input enough for a prediction shift, while large masking percentages result in counterfactuals that change the input excessively, diminishing their interpretability.

The evaluation of uncertain predictions highlighted cases where the model correctly refrained from confidently classifying ambiguous inputs and instances where the model exhibited uncertainty even in clear-cut scenarios. The editor produces These insights provide valuable information on the semantic boundaries within the model, aiding in a more nuanced evaluation of reliability and robustness.

In conclusion, this research contributes to the growing field of Explainable AI by emphasizing the importance of understanding and utilizing model uncertainty. Future work could involve further refinement of counterfactual generation methods, exploration of additional editor modifications, and application of the proposed approach to various NLP tasks. Ultimately, the pursuit of more interpretable and reliable models remains crucial for the broader adoption of AI technologies across different domains.

VII. Future Work

The current study lays the groundwork for future investigations in multiple directions. Firstly, enhancing the proposed method by refining editor modifications and experimenting with additional parameters could lead to more effective generation of counterfactuals maximizing model uncertainty.

Further exploration of different counterfactual editors, such as TextFooler, with modifications to induce uncertainty is essential. Investigating the impact of less important changes and incorporating diverse approaches in counterfactual generation methods could provide a more comprehensive understanding of model behavior.

Additionally, extending the application of this approach to various NLP tasks beyond sentiment analysis on the IMDb dataset is vital. Evaluating model uncertainty in different contexts, such as named entity recognition or question answering, can offer insights into the generalizability of the proposed methodology.

Furthermore, integrating calibrated models into the counterfactual generation process could enhance the reliability of uncertainty assessments. Calibrating the model's confidence to align with actual performance would contribute to more accurate predictions.

In conclusion, the future work should focus on refining the proposed methodology, exploring alternative approaches, and applying the approach to diverse NLP tasks to advance the understanding of model uncertainty in artificial intelligence systems.

References

[1] Nitay Calderon, Eyal Ben-David, Amir Feder, and Roi Reichart. Docogen: Domain counterfactual generation for low resource domain adaptation. *ArXiv*, abs/2202.12350, 2022. Cited 24 times.

[2] Yangyi Chen, Lifan Yuan, Ganqu Cui, Zhiyuan Liu, and Heng Ji. A close look into the calibration of pre-trained language models. *ArXiv*, abs/2211.00151, 2022. Cited 14 times.

[3] Susanne Dandl, Christoph Molnar, Martin Binder, and B. Bischl. Multi-objective counterfactual explanations. In *Parallel Problem Solving from Nature*, 2020. Cited 169 times.

[4] Giorgos Filandrianos, Edmund Dervakos, Orfeas Menis-Mastromichalakis, Chrysoula Zerva, and G. Stamou. Counterfactuals of counterfactuals: a back-translation-inspired approach to analyse counterfactual editors. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[5] Mengting Hu, Zhen Zhang, Shiwan Zhao, Minlie Huang, and Bingzhe Wu. Uncertainty in natural language processing: Sources, quantification, and applications. *ArXiv*, abs/2306.04459, 2023. Cited 6 times.

[6] Abhyuday N. Jagannatha and Hong Yu. Calibrating structured output predictors for natural language processing. *Proceedings of the conference. Association for Computational Linguistics. Meeting*, 2020:2078–2092, 2020. Cited 24 times.

[7] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *AAAI Conference on Artificial Intelligence*, 2019. Cited 651 times.

[8] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Conference on Empirical Methods in Natural Language Processing*, 2020. Cited 421 times.

[9] Rafael Poyiadzi, Kacper Sokol, Raúl Santos-Rodríguez, Tijl De Bie, and Peter A. Flach. Face: Feasible and actionable counterfactual explanations. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2019. Cited 243 times.

[10] Alexis Ross, Ana Marasović, and Matthew E. Peters. Explaining nlp models via minimal contrastive editing (mice). *ArXiv*, abs/2012.13985, 2020. Cited 75 times.

[11] F. Sharmila Satthar, Roger Evans, and Gulden Uchyigit. A calibration method for evaluation of sentiment analysis. In *Recent Advances in Natural Language Processing*, 2017. Cited 2 times.

[12] Sandra Wachter, Brent Daniel Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Cybersecurity*, 2017. Cited 1644 times.

[13] Tongshuang Sherry Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S. Weld. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Annual Meeting of the Association for Computational Linguistics*, 2021. Cited 139 times.

[14] Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. *ArXiv*, abs/1811.07253, 2018. Cited 95 times.