Interpreting Text Classifiers by Learning Context-sensitive Influence of Words

Sawan Kumar*
IISc, Bangalore, India
sawankumar@iisc.ac.in

Kalpit Dixit
Amazon AWS AI, USA
kddixit@amazon.com

Kashif Shah
Amazon AWS AI, USA
shahkas@amazon.com

Abstract

Many existing approaches for interpreting text classification models focus on providing importance scores for parts of the input text, such as words, but without a way to test or improve the interpretation method itself. This has the effect of compounding the problem of understanding or building trust in the model, with the interpretation method itself adding to the opacity of the model. Further, importance scores on individual examples are usually not enough to provide a sufficient picture of model behavior. To address these concerns, we propose MOXIE (MOdeling conteXt-sensitive InfluencE of words) with an aim to enable a richer interface for a user to interact with the model being interpreted and to produce testable predictions. In particular, we aim to make predictions for importance scores, counterfactuals and learned biases with MOXIE. In addition, with a global learning objective, MOXIE provides a clear path for testing and improving itself. We evaluate the reliability and efficiency of MOXIE on the task of sentiment analysis.

Input text: *he played a homosexual character* **Model prediction**: Negative sentiment[1]
Question 1 (Importance scores): Which words had the most influence towards the prediction? Is the word '*homosexual*' among them? **Answer**: The word '*homosexual*' has the highest negative influence.
Question 2 (Counterfactuals): If so, which words instead would have made the prediction positive? **Answer**: If you replace the word '*homosexual*' with the word '*straight*', the model would have made a positive sentiment prediction.
Question 3 (Biases): Is there a general bias against the word '*homosexual*' compared to the word '*straight*'? **Answer**: Yes, there are a large number of contexts where the model predicts negatively with the word '*homosexual*', but positively with the word '*straight*'. Here are some examples: • *the most* ***homosexual*** *thing about this film* • *though it's equally* ***homosexual*** *in tone* • . . .

Table 1: *Example questions we aim to answer using MOXIE.* The first question has commonly been addressed in existing approaches. The ability of an interpretation method to answer the second and third questions enables a rich and testable interface.

1 Introduction

Interpretability, while under-specified as a goal, is a crucial requirement for artificial intelligence (AI) agents (Lipton, 2018). For text classification models, where much of the recent success has come from large and opaque neural network models (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019), a popular approach to enable interpretability is to provide importance scores for parts of the input text, such as words, or phrases. Given only these numbers, it is difficult for a user to understand or build trust in the model. Going beyond individual examples, such as scalable and testable methods to identify biases at a dataset level, are desired but currently missing. Questions can be raised about whether the methods of interpretation themselves are trustworthy. Recent analyses (Ghorbani et al., 2019) of such interpretation methods for computer vision tasks suggest that such skepticism is valid and important.

A method which aims to elucidate a black-box's behavior should not create additional black boxes. Measuring trustworthiness, or faithfulness[2], of interpretation methods, is itself a challenging task (Jacovi and Goldberg, 2020). Human evaluation is not only expensive, but as Jacovi and Goldberg (2020) note human-judgments of quality shouldn't

*Work done during an internship at Amazon AWS AI, USA.

[1]No offense is intended towards any particular community in this or in subsequent sections. Rather, we are interested in probing for unexpected biases.

[2]In this work, a faithful interpretation is one which is aligned with the model's reasoning process. The focus of this work is to make predictions testable by the model being interpreted and thus have a clear measure of faithfulness.

Proceedings of the First Workshop on Trustworthy Natural Language Processing, pages 55–67
June 10, 2021.

be used to test the faithfulness of importance scores. What needs testing is whether these scores reflect what has been learned by the model being interpreted, and not whether they are plausible scores.

We believe the aforementioned issues in existing methods that produce importance scores can be circumvented through the following changes.

A global learning objective: Several existing approaches rely on some heuristic to come up with importance scores, such as gradients (Wallace et al., 2019), attentions (Wiegreffe and Pinter, 2019), or locally valid classifiers (Ribeiro et al., 2016) (see Atanasova et al. (2020) for a broad survey). Instead, we propose to identify a global learning objective which, when learned, enables prediction of importance scores, with the assumption that if the learning objective was learned perfectly, we would completely trust the predictions. This would provide a clear path for testing and improving the interpretation method itself. Quick and automatic evaluation on a held-out test set allows progress using standard Machine Learning (ML) techniques.

Going beyond importance scores: Importance scores, even when generated using a theoretically inspired framework (Sundararajan et al., 2017), are generally hard to evaluate. Further, the aim of the interpretation method shouldn't be producing importance scores alone, but to enable a user to explore and understand model behavior[3], potentially over large datasets. In Table 1, we illustrate a way to do that through a set of questions that the interpretation method should answer. Here, we provide more details on the same.

Importance Scores 'Which parts of the input text were most influential for the prediction?' Such importance scores, popular in existing approaches, can provide useful insights but are hard to evaluate.

Counterfactuals 'Can it predict counterfactuals?' We define a good counterfactual as one with minimal changes to the input text while causing the model to change its decision. Such predictions can be revealing but easy to test. They can provide insights into model behavior across a potentially large vocabulary of words.

In this work, we consider counterfactuals obtained by replacing words in the input text with other words in the vocabulary. We limit to one replacement.

Biases 'Is the model biased against certain words?' For example, we could ask if the model is biased against LGBTQ words, such as the word '*homosexual*' compared to the word '*straight*'? One way to provide an answer to such a question is to evaluate a large number of contexts, replacing a word in the original context with the words'*homosexual*' and '*straight*'. Doing that however is prohibitive with large text classification models. If an interpretation method can do this in a reasonable time and accuracy, it enables a user access to model behavior across a large number of contexts.

Considering the preceding requirements, we propose MOXIE (MOdeling conteXt-sensitive InfluencE of words) to enable a reliable interface for a user to query a neural network based text classification model beyond model predictions. In MOXIE, we aim to learn the context-sensitive influence of words (see Figure 1 for the overall architecture). We show that learning this objective enables answers to the aforementioned questions (Section 3.2). Further, having a global learning objective provides an automatic way to test the interpretation method as a whole and improve it using the standard ML pipeline (Section 3.3). We evaluate the reliability and efficiency of MOXIE on the task of sentiment analysis (Section 4)[4].

2 Related Work

Word importance scores have been a popular area of research for interpreting text classifiers, including gradient based methods (Wallace et al., 2019), using nearest neighbors (Wallace et al., 2018), intrinsic model-provided scores such as attention (Wiegreffe and Pinter, 2019), and scores learned through perturbations of the test example (Ribeiro et al., 2016). There has also been effort to expand the scope to phrases (Murdoch et al., 2018), as well as provide hierarchical importance scores (Chen et al., 2020). However these methods tend to derive from an underlying heuristic applicable at the example level to get the importance scores. With

[3]The need for going beyond importance scores has also been realized and explored for user-centric explainable AI interface design (Liao et al., 2020).

[4]Note that we are not claiming to build inherently faithful mechanisms, but ones which allow inherent testing of their faithfulness. For example, a counterfactual or a bias prediction can be tested by the model under interpretation (see Section 4).

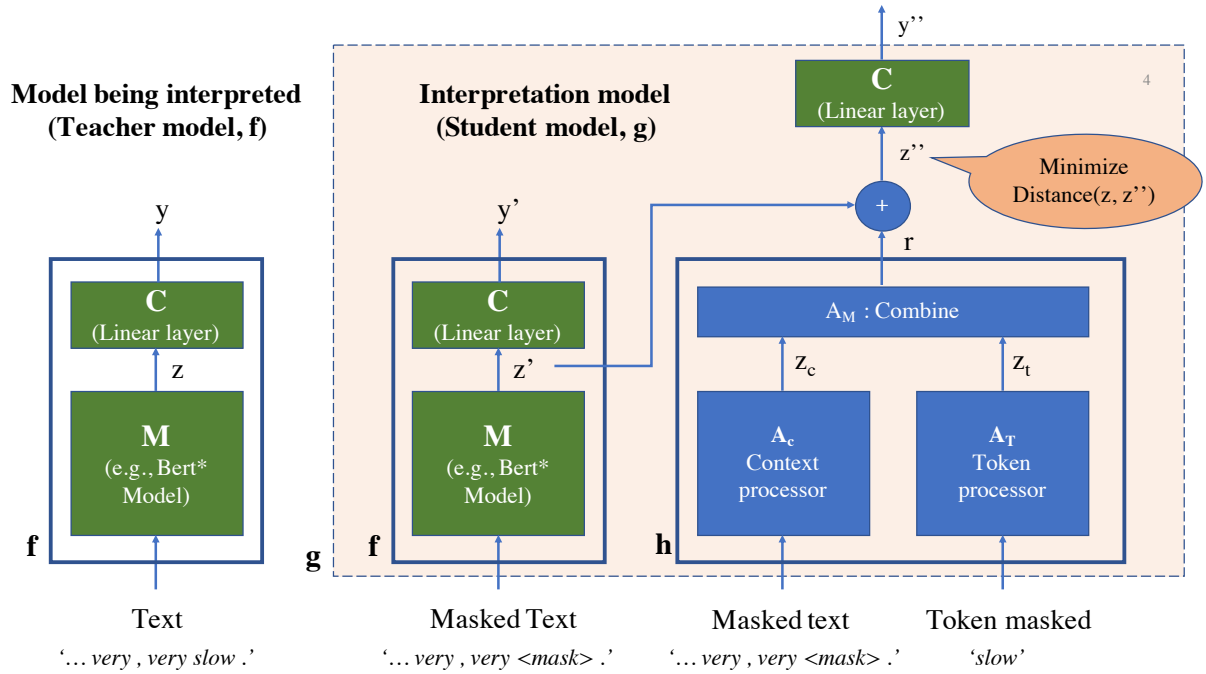

Figure 1: *Overall architecture of MOXIE*: The model being interpreted (f) which we call the teacher model is shown on the left. It processes an input text such as '*... very very slow*' to produce a representation z through module M and label scores y through a linear classification layer C. When presented with the same input but the word '*slow*' masked, it produces outputs z' and y' respectively. We learn the difference in the two representations $(z - z')$ as a proxy for the context-sensitive influence of the word '*slow*' in the student model (g). This is done by processing the masked context and the token masked through arbitrarily complex modules A_C and A_T which produce fixed length representations z_c and z_t respectively. The combine module (A_M) takes these as input to produce the output r. We learn by minimizing the mean square error between z and $z'' = z' + r$. Keeping the combine module shallow allows the processing of a large number of tokens for a given context and vice versa in a reasonable time. Please see Section 3.1 for details on the architecture and Section 3.2 for how this architecture enables answers to the motivating questions.

perturbation methods, where a locally valid classifier is learned near the test example (Ribeiro et al., 2016), there is a hyperparameter dependence as well as stochasticity at the level of test examples.

While it's not inherently problematic to use such heuristics, it makes it hard to improve upon the method, as we need to rely on indirect measures to evaluate the method. Further, recent work has shown that the skepticism in the existing methods is valid and important (Ghorbani et al., 2019). In this work, we use a global learning objective which allows us to make predictions of importance scores.

Apart from word importance scores, explanation by example style method have been studied (Han et al., 2020). Like word importance based methods, however, these methods don't provide a clear recipe for further analysis of the model. In this work, we aim to produce testable predictions such as counterfactuals and potential biases.

Measuring faithfulness of an interpretation model can be hard. Jacovi and Goldberg (2020) suggest that human evaluation shouldn't be used. In this work, we circumvent the hard problem of evaluating the faithfulness of an interpretation method by making it output predictions which can be tested by the model being interpreted.

3 MOXIE

The overall architecture employed to learn MOXIE is shown in Figure 1. We introduce the notation and describe the architecture in detail in Section 3.1. In Section 3.2, we discuss how MOXIE provides answers to the motivating questions.

3.1 Notation and Architecture

Let x denote a text sequence $x_1x_2...x_n$. We denote by x^i_{mask} the same sequence but the ith token x_i replaced by a mask token: $x_1x_2 \ldots x_{i-1}\langle\text{mask}\rangle x_{i+1} \ldots x_n$.

In the following, we refer to the model being

interpreted as the teacher model and the learned interpretation model as the student model.

Teacher model: The teacher model f is composed of a representation module M and a linear classification layer C, and produces a representation $z = M(x)$ and label scores $y = C(z)$ for a text input x. The label prediction is obtained as the label with the highest score: $l = \text{argmax}(y)$. We believe this covers a fairly general class of text classifiers: $y = f(x) = C(M(x)) = C(z)$.

Student model: With mask token mask_t for the teacher model, we create masked input $x^i_{\text{mask}_\text{t}}$ for which the teacher model outputs $z'_i = M(x^i_{\text{mask}_\text{t}})$.

As a proxy for the context-sensitive influence of the token x_i, we aim to model $z - z'_i$ in the student model. For this, we use the following submodules:

- **Context processor** A_C processes masked text to produce a context representation. In particular, with mask token mask_s for the context processor, we create the masked input $x^i_{\text{mask}_\text{s}}$ for which the context processor outputs $z_{c,i} = A_C(x^i_{\text{mask}_\text{s}})$. Note that the mask token could be different for the teacher model and the context processor. We fine-tune a pre-trained roberta-base (Liu et al., 2019) model to learn the context processor, where we take the output at the mask token position as $z_{c,i}$.

- **Token processor** A_T processes the token which was masked to produce representation $z_{t,i} = A_T(x_i)$. Note that we can mask spans as well with the same architecture, where x_i denotes a span of tokens instead of one. For all our experiments, we fine-tune a pre-trained RoBERTa-base model to learn the token processor, where we take the output at the first token position as $z_{t,i}$.

- **Combine** module A_M combines the outputs from the context and token processors to produce representation r.

In summary, the sub-module h takes the input x and token location i to produce output r_i:

$$r_i = h(x, i) = A_M(A_C(x^i_{\text{mask}_\text{s}}), A_T(x_i)) \quad (1)$$

To get label predictions, we add z'_i to r_i and feed it to the teacher model classification layer C. In summary, the student model g takes as input x and token location i to make predictions y''_i:

$$y''_i = g(x, i) = C(z'_i + h(x, i)) = C(z'_i + r_i) \quad (2)$$

Modules h and g provide token influence and label scores respectively. We learn the parameters of the student model by minimizing the mean square error between z and z''_i.

Keeping the combine module shallow is crucial as it allows evaluating a large number of tokens in a given context and vice versa quickly (Section 3.2). For all our experiments, we first concatenate $z_{c,i} + z_{t,i}$, $z_{c,i} - z_{t,i}$ and $z_{c,i} \odot z_{t,i}$ to obtain z_{concat}, where $\odot$ represents element wise multiplication. $z_{\text{concat},i}$ is then processed using two linear layers:

$$A_M(z_{c,i}, z_{t,i}) = W_2(\tanh(W_1 z_{\text{concat},i} + b_1)) + b_2 \quad (3)$$

where W_1, b_1, W_2, and b_2 are learnable parameters. The parameter sizes are constrained by the input and output dimensions and assuming W_1 to be a square matrix.

3.2 Using MOXIE

3.2.1 Importance Scores

MOXIE provides two kinds of token-level scores.
Influence scores can be obtained from predictions of the sub-module h, $r_i = h(x, i)$:

$$\hat{s}_i = \text{softmax}(C(r_i)) \quad (4)$$

For binary classification, we map the score to the range $[-1, 1]$ and select the score of the positive label: $s_i = 2 * \hat{s}_i[+\text{ve}] + 1$. The sign of the score s_i can then be interpreted as indicative of the sentiment (positive or negative), while its magnitude indicates the strength of the influence.
Unlike ratios aim to give an estimate of the ratio of words in the vocabulary which when used to replace a token lead to a different prediction. The student model architecture allows us to pre-compute and store token representations through the token processor (A_T) for a large vocabulary, and evaluate the impact each token in the vocabulary might have in a given context. This requires running the context processor and the teacher model only once. Let V be a vocabulary of words, then for each word w^j, we can pre-compute and store token embeddings E_V such that $E^j_V = A_T(w^j)$. For example x with label l, teacher model representations z and z'_i for the full and masked input, and context processor output $z_{c,i}$, the unlike ratio u_i can be computed as:

$$\begin{aligned} r_{V,i} &= A_M(z_{c,i}, E_V) \\ y_{V,i} &= C(z + r_{V,i}) \\ u_i &= \frac{|\{w : w \in V, \text{argmax}(y_{V,i}) \neq l\}|}{|V|} \end{aligned} \quad (5)$$

If the unlike ratio u_i for a token x_i is 0, it would imply that the model prediction is completely determined by the rest of the context. On the other hand, an unlike ratio close to 1.0 would indicate that the word x_i is important for the prediction as replacing it with any word is likely to change the decision. In this work we restrict the vocabulary V using the part-of-speech (POS) tag of the token in consideration (see Appendix C for details).

Finally, getting **phrase-level scores** is easy with MOXIE when the student model is trained by masking spans and not just words.

Please see Section 4.3 for details and evaluation.

3.2.2 Counterfactuals

As discussed in the preceding section, the student model allows making predictions for a large number of token replacements for a given context. As before, we restrict the vocabulary of possible replacements using the POS tag of the token in consideration. To generate potential counterfactuals, we get predictions from the student model for all replacements and select the ones with label predictions different from the teacher model's label. Please see Section 4.4 for details and evaluation.

3.2.3 Biases

Modeling the context-sensitive influence of words in MOXIE enables analyzing the effect of a word in a large number of contexts. We can pre-compute and store representations for a large number of contexts using the teacher model and the context processor of the student model. Given a query word, we can then analyze how it influences the predictions across different contexts. Pairwise queries, i.e., queries involving two words can reveal relative biases against a word compared to the other. Please see Section 4.5 for details and evaluation.

3.3 Improving the Interpretation Method

The student model g introduced in the preceding section is expected to approximate the teacher model f, and the accuracy of the same can be measured easily (see Section 4.2). We expect that as this accuracy increases, the answers to the preceding questions will become more reliable. Thus, MOXIE provides a straightforward way to improve itself. The standard ML pipeline involving testing on a held-out set can be employed.

4 Experiments

We aim to answer the following questions:

Q1 How well does the student model approximate the teacher model? (Section 4.2)

Q2 How does MOXIE compare with methods which access test example neighborhoods to generate importance scores? (Section 4.3)

Q3 Can MOXIE reliably produce counterfactuals? (Section 4.4)

Q4 Can MOXIE predict potential biases against certain words? (Section 4.5)

We use the task of binary sentiment classification on the Stanford Sentiment Treebank-2 (SST-2) dataset (Socher et al., 2013; Wang et al., 2018) for training and evaluation. In Section 4.1.2, we provide text preprocessing details. We evaluate the student model accuracy against the teacher model (Q1) across four models: bert-base-cased (Devlin et al., 2019), roberta-base (Liu et al., 2019), xlmr-base (Conneau et al., 2019), RoBERTa-large (Liu et al., 2019). For the rest of the evaluation, we use RoBERTa-base as the teacher model. We use the Hugging Face transformers library v3.0.2 (Wolf et al., 2019) for our experiments.

4.1 Experimental Setup

4.1.1 Training Details

As models to be interpreted (teacher models), we fine-tuned bert-base-cased, RoBERTa-base, xlmr-base and RoBERTa-large on the SST-2 train set. We trained each model for 3 epochs.

For the interpretation models (student models), we initialize the context processor and token processor with a pre-trained RoBERTa-base model. We then train the context processor, token processor and combine module parameters jointly for 10 epochs with model selection using dev set (using all-correct accuracy, see Section 4.2 for details).

For both teacher and student models, we use the AdamW (Loshchilov and Hutter, 2018) optimizer with an initial learning rate of $2e-5$ (see Appendix A for other training details).

For all experiments, for training, we generate context-token pairs by masking spans obtained from a constituency parser (the span masked is fed to the token processor). For all evaluation, we use a word tokenizer unless otherwise specified. Training with spans compared to words didn't lead to much difference in the overall results (as measured in Section 4.2), and we retained the span version to potentially enable phrase level scores.

Model	Teacher baseline (Context-only)	Student Model (Context &Token)
bert-base-cased	73.48	87.64
RoBERTa-base	78.64	89.24
xlmr-base	74.08	86.93
RoBERTa-large	82.37	89.74

Table 2: *Evaluation of the student model and a context-only teacher baseline against teacher model predictions on the test set* using the all-correct accuracy metric. The context-only teacher model baseline does better than chance but the student model provides gains across all teacher models. This indicates that the student model learns context-token interactions. Please see Section 4.2 for details.

Text: it 's a charming and often affecting journey **Prediction:** +ve **Top 2 scores:** charming (0.38), affecting (0.12)
Text: unflinchingly bleak and desperate **Prediction**: -ve **Top 2 scores:** bleak (-0.99), desperate (-0.92)
Text: allows us to hope that nolan is posed to embark a major career as a commercial yet inventive filmmaker . **Prediction:** +ve **Top 2 scores:** allows (0.97), inventive (0.91)

Table 3: *Word importance scores* on the first three dev set examples. Top two scores are shown.

4.1.2 Tokenization and POS Tagging

We use the nltk (Bird et al., 2009) tokenizer for getting word level tokens. For training by masking spans, we obtain spans from benepar (Kitaev and Klein, 2018), a constituency parser plugin for nltk. We use nltk's averaged_perceptron_tagger for obtaining POS tags, and use the universal_tagset.

4.2 Evaluating Student Model on the Test Set

In this section, we measure how well the student model approximates the teacher model. The student model provides a prediction at the token level: $g(x, i)$. We define an example level **all-correct accuracy** metric: the set of predictions for an example are considered correct only if all predictions match the reference label.

As a baseline, we consider token level predictions from the teacher model obtained from masked contexts: $f(x^i_{\text{mask}_t})$. If the student model improves over this baseline, it would suggest having learned context-token interactions and not just using the contexts for making predictions.

In Table 2, we show all-correct accuracies of the baseline and the student model on the test set. The baseline does better than chance but the student model provides significant gains over it. This indicates that the student model learns context-token interactions and is not relying on the context alone.

Text: in exactly 89 minutes , most of which passed as slowly as if i 'd been sitting naked on an igloo , formula 51 sank from quirky to jerky to utter turkey . (-ve) **Prediction:** -ve **Top 2 word-level scores:** quirky (0.26), formula (-0.22) **Top 2 phrase-level scores:** to utter turkey (-0.35), quirky (0.26)

Table 4: *Word and Phrase importance scores* on an example selected from the first 10 dev set examples.

A key advantage of MOXIE is providing a way to improve upon itself. We believe improvements in the all-correct accuracy of the student model would lead to improved performance when evaluated as in the subsequent sections. For completion, we provide the accuracies of the student model against gold labels in Appendix B.

4.3 Importance Scores

Table 3 capture the importance scores on the first three dev set examples. Table 4 shows an example selected from the first 10 dev set examples demonstrating how MOXIE can produce meaningful phrase-level scores.

As discussed before, it's hard to evaluate importance scores for trustworthiness. We evaluate the trustworthiness of MOXIE in subsequent sections. Here, we aim to contrast MOXIE, which doesn't learn its parameters using test examples, with methods which do. We aim to devise a test which would benefit the latter and see how well MOXIE performs. We choose LIME (Ribeiro et al., 2016) which directly incorporates the knowledge of teacher model predictions when words in the input text are modified. To test the same, we start with test examples where the teacher model makes an error, and successively mask words using importance scores, with an aim to correct the label prediction. With a masking budget, we compute the number of tokens that need masking. We report on: **Coverage**, the % of examples for which the model decision could be changed, and **Average length masked**, the average number of words that needed masking (see Appendix D for detailed steps). The test favors LIME as LIME learns using teacher model predictions on the test example and its neighborhood while MOXIE learns only on the train set.

We compare against LIME and a Random baseline where we assign random importance scores to the words in the input. From MOXIE, we obtain

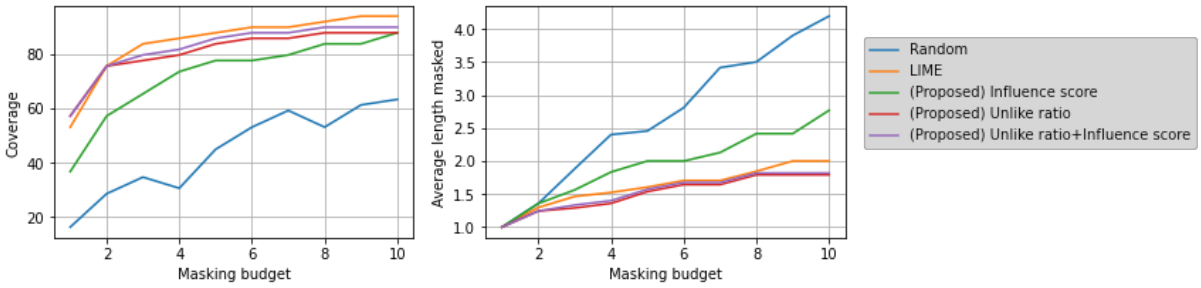

Figure 2: *Evaluation of importance scores* on examples where the teacher model makes an error. Tokens are successively masked using importance scores until the masking budget is met or the prediction of the teacher model changes. We report on the coverage and the average length that needed masking when the decision could be changed. We note that all methods perform better than the random baseline. MOXIE competes with LIME despite not seeing the test example and its neighborhood during training. Please see Section 4.3 for details.

influence scores and unlike ratios. We also derive a hybrid score (Unlike ratio+influence score) by using unlike ratios with influence scores as back-off when the former are non-informative (e.g., all scores are 0). Figure 2 captures the results of this test on the 49 dev set examples where the teacher model prediction was wrong.

We note that all scores are better than the random baseline. Influence scores do worse than LIME but unlike ratios and the hybrid scores are competitive with LIME. This is despite never seeing the test example neighborhood during training, unlike LIME. The results support the hypothesis that a global learning objective can provide effective importance scores. However, this is not the main contribution of this paper. Our aim is to enable increased interaction with the model by providing testable predictions as discussed in subsequent sections.

4.4 Counterfactuals

As discussed in the Section 3.2.2, MOXIE allows predictions of counterfactuals using pre-computed token embeddings. We show examples of generated counterfactuals in Appendix E.1. We evaluate the reliability of the generated counterfactuals by computing the accuracy of the top-10 predictions using the teacher model. The student model takes a pre-computed POS-tagged dictionary of token embeddings (obtained using token processor A_T) and a context as input and predicts the top-10 candidate replacements (see Appendix E.2 for details).

Figure 3 captures the counterfactual accuracies obtained across contexts (with at least one counterfactual) in the dev set. Out of 872 examples, 580 examples had at least one context for which the student model made counterfactual predictions. In total, there were 1823 contexts with counterfactuals. The median counterfactual accuracy across contexts with at least one counterfactual was 90% which is significantly higher than chance.

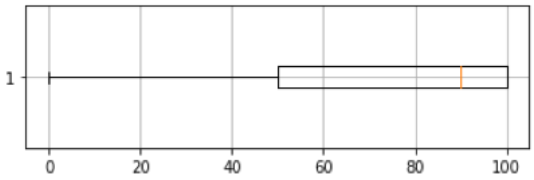

Figure 3: *Counterfactual prediction accuracies*: across contexts for which at least one counterfactual was found. The box indicates the range between quartiles 1 & 3. The median accuracy was 90.0% which is better than chance. This indicates that MOXIE is capable of reliably predicting counterfactuals. Please see Section 4.4 for details.

4.5 Biases

As discussed in the Section 3.2.3, MOXIE can quickly process a large number of contexts for a given word. As a case study, we look for potential biases against LGBTQ words in the teacher model.

We make pairwise queries to the student model, with a pair of words: a control word and a probe word, where we expect task specific meaning to not change between these words. We require the student model to find contexts from an input dataset where the control word leads to a positive sentiment prediction but the probe word leads to a negative sentiment prediction. We use the training dataset as the input dataset.

To avoid any negative influence from other parts of the context, we further require that the original context (as present in the input dataset) lead to a positive sentiment by the teacher model. Finally, we remove negative contexts, e.g., the context '*The*

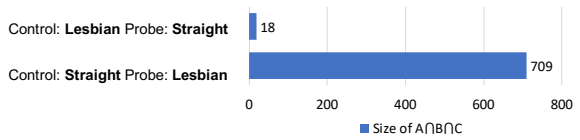

Figure 4: *Measuring potential biases*: using the student model. We show the relative sizes of the sets obtained with positive predictions with control word but negative predictions with the probe word. The results indicate a potential bias against the word '*lesbian*' compared to the word '*straight*' (see Section 4.5)

*movie is not **bad***' would be positive despite '*bad*' clearly having a negative influence. To ease the bias analysis by remove such contexts, we can remove all sentences with words which tend to be negative, e.g., *not*, *never* etc. For adjective contexts, we use the student model to filter out such contexts using a list of clearly positive/negative adjectives (see Appendix F for details on pre-processing contexts).

The output of the preceding steps can be pre-computed and stored. Next, we find the set of contexts satisfying the following criteria (e.g., with control word '*straight*' and probe word '*lesbian*'):

S_1 Teacher model predicts positive on the original context (pre-computed and stored), e.g., x:'*I have **cool** friends*', $\text{argmax}(f(x)) = +ve$.

S_2 Student model predicts positive when the marked token is replaced with the control word, e.g., x_{control}:'*I have **straight** friends*', $\text{argmax}(g(x_{\text{control}}, i)) = +ve$.

S_3 Student model predicts negative when the marked token is replaced with the probe word, e.g., x_{probe}: '*I have **lesbian** friends*', $\text{argmax}(g(x_{\text{probe}}, i)) = -ve$.

S_2 and S_3 can be computed efficiently by pre-computing the output of the context processor A_C for all contexts in the input dataset. If E_C denotes the matrix of output embeddings from the context processor, S_2 and S_3 for word w can be computed by first obtaining the token processor representation $z_t = A_T(w)$ and then using the combine module $y_C = C(A_M(E_C, z_t))$.

The relative size of the set $S_1 \cap S_2 \cap S_3$ is indicative of a potential bias against the probe word. Figure 4 shows the size of the set $S_1 \cap S_2 \cap S_3$ with '*straight*' and '*lesbian*' interchangeably as control and probe words. Note that the relative size with probe word as '*lesbian*' is much larger than the relative size with probe word as '*straight*'. This is indicative of a potential bias against the word '*lesbian*'. Table 5 shows some examples of biased sentences obtained through this procedure.

almost every **lesbian** facet of production
gay to its animatronic roots
the **bisexual** lives of the characters in his film
the most **transsexual** thing about this film

Table 5: *Examples of biased contexts (negative prediction).* If the highlighted word were to be swapped by the word '*straight*', the prediction would be positive. See Section 4.5 for details.

Size of $S_1 \cap S_2 \cap S_3$ (top-100)	Control word	Acc	Probe word	Acc
100	straight	67.0	lesbian	90.0
100	straight	61.0	gay	93.0
100	straight	68.0	bisexual	82.0
100	straight	69.0	transsexual	85.0
0	straight	-	queer	-

Table 6: *Evaluating student model claims of biases*: Up to 100 confident contexts are selected using student model predictions where the student model claims a +ve prediction using the control word and -ve prediction using the probe word. The predictions are tested using the teacher model and the accuracy reported. Note that except for '*queer*' where the set size is zero, the prediction accuracy of the student model is better than chance. This indicates the ability of the student model to predict biases. See Section 4.5 for details.

Next, we aim to evaluate the claim of the student model using the teacher model. For this, we consider the set $S_1 \cap S_2 \cap S_3$ with probe word as '*lesbian*' and evaluate the contexts with both '*straight*' and '*lesbian*'. The student model claims the model prediction to be positive for the former and negative for the latter. We process the examples with the corresponding replacements using the teacher model to measure the accuracy of this claim (i.e., teacher model's outputs serve as the reference label). The accuracy of the student model claim with '*straight*' is 65.16% while with '*lesbian*', it is 75.88%. We also evaluate the 100 most confident predictions from the student model (using softmax scores). The accuracies with '*straight*' and '*lesbian*' then increase to 67.0% and 90.0% respectively. In Table 6, we show the results on the 100 most confident predictions for more LGBTQ words. Note that we don't claim this to be an exhaustive set of words reflecting the LGBTQ community, but as only roughly representative. The results indicate a similar pattern as with '*lesbian*', except for the

word '*queer*' where the student model doesn't predict any biased contexts. This is presumably due to the word '*queer*' carrying additional meanings, unlike the other LGBTQ words.

Finally, the student model provides ~450 speedup when compared to using the teacher model to probe for biases. It takes less than $1s$ to test a control word against a probe word on a single NVIDIA V100 GPU using the student model, thus enabling an interactive interface. Unlike using the teacher model directly, MOXIE allows precomputing large sets of context/token representations and thus obtain the aforementioned gains.

In summary, the results indicate bias against LGBTQ words. The evaluation indicates that the student model can make reliable bias predictions.

5 Conclusion

In summary, we have shown that MOXIE provides a novel framework for interpreting text classifiers and a method to draw quick insights about the model on large datasets. MOXIE can make efficient, testable and reliable predictions beyond importance score, such as counterfactuals and potential biases. Further, with a global learning objective, it provides a clear path for improving itself using the standard ML pipeline. Finally, the principles and the evaluation methodology should help the interpretability research overcome the problem of testing the faithfulness of interpretation methods.

As future work, we identify improving the accuracy of the student model. Further analysis of the nature of counterfactuals selected by the student model could lead to useful insights towards improving the interpretation method. Finally, identifying other learning objectives which enable testable predictions would be useful and challenging.

6 Broader Impact

In this work, we aim to improve interpretability of existing text classification systems. More interpretable systems are likely to reveal biases and help towards a fairer deployment of production systems built using these systems.

To demonstrate our work, we choose to study potential biases against words associated with the LGBTQ community. In particular, we probe for bias in a learned sentiment classification systems against the words that make up the acronym LGBTQ - Lesbian, Gay, Bisexual, Transsexual and Queer. Note that we don't use identity information of any individual for this. Instead, we probe whether, in arbitrary contexts, the learned sentiment classification model is likely to find these qualifiers more negative when compared to adjectives in general or adjectives usually associated with the hegemony. Our work doesn't aim to discriminate but instead provides a way to measure if there are intended or unintended biases in a learned system.

Acknowledgements

We would like to thank the reviewers for their valuable feedback.

References

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Amirata Ghorbani, Abubakar Abid, and James Zou. 2019. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions

and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online. Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the ai: informing design practices for explainable ai user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Zachary C Lipton. 2018. The mythos of model interpretability. *Queue*, 16(3):31–57.

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

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

W James Murdoch, Peter J Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Eric Wallace, Shi Feng, and Jordan Boyd-Graber. 2018. Interpreting neural networks with nearest neighbors. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 136–144, Brussels, Belgium. Association for Computational Linguistics.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP interpret: A framework for explaining predictions of NLP models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 7–12, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

A Training Details

A.1 Data

The SST-2 dataset (Socher et al., 2013; Wang et al., 2018) contains English language movie reviews from the "Rotten Tomatoes" website. The training data consists of 67349 examples and is roughly label-balanced with 56% positive label and 44% negative label data. The dev and test sets contain 872 and 1821 examples respectively.

A.2 Other Training Details

For the teacher models, we train the models for 3 epochs. For optimization, we use an initial learning rate of 2e-5, adam epsilon of 1e-8, max gradient norm of 1.0 and a batch size of 64. The maximum token length for a text example was set to 128.

	Teacher Model (accuracy)	Student Model (All-correct accuracy)
bert-base-cased	91.97	85.09
roberta-base	94.38	86.93
xlmr-base	92.55	83.37
roberta-large	95.64	88.88

Table 7: *Accuracy against gold labels* on the dev set. The student model does significantly better than chance with scope for improvement.

POS tag	Size of extracted vocabulary
NOUN	7534
VERB	2749
ADJ	3394
ADV	809
.	15
DET	26
ADP	79
CONJ	12
PRT	19
PRON	28

Table 8: *Size of extracted lists of POS-tagged words*.

For student models, we train the models for 10 epochs. For optimization, we use an initial learning rate of 2e-5, adam epsilon of 1e-8, max gradient norm of 1.0 and a batch size of 64. The maximum token length for a text example was set to 128. The maximum token length of the masked span input to the token processor was set to 50. When trained on Nvidia's GeForce GTX 1080 Ti GPUs, each run took approximately 6 hours to complete.

B Evaluating Student Model against Gold Labels

In Table 7, we provide the accuracies of the teacher and student models against gold labels. In this work, we care about accuracies of the student model against teacher model predictions and we show accuracies against gold labels here only for completion.

C Pre-computing POS-tagged Dictionary of Token Embeddings

For evaluating importance scores and counterfactual predictions, we use a POS-tagged dictionary of token embeddings. The token embeddings are obtained by processing the tokens through the token processor A_T. This is done only once for a given student model and used for all subsequent experiments.

We use the training dataset for extracting the list of open class words. We use nltk's averaged_perceptron_tagger for obtaining POS tags, and use the universal_tagset[5]. The open class words correspond to the tags — *NOUN*, *VERB*, *ADJ*, *ADV*. We assign each word to the POS tag with which it occurs most commonly in the training dataset.

For closed class words, we use the Penn Treebank corpus included in the ntlk toolkit (*treebank*). Again, we use the universal_tagset from nltk toolkit. We ignore the *NUM* and *X* as well as open class tags. For the punctuation tag, we remove any token containing alphanumeric characters.

In Table 8, we show the size of the extracted lists for each POS tag.

Input: A text sequence x: $x_1x_2...x_n$
Input: Importance scores s: $s_1s_2...s_n$
Input: A masking budget m
Output: The number of words that need masking
Initialize: count $\leftarrow -1$; $a \leftarrow x$
Compute teacher prediction:
  prediction $\leftarrow \text{argmax}(f(a))$
Sort importance scores:
  ImportanceOrder $\leftarrow \text{argsort}(s)$
for $k \leftarrow 1$ ***to*** m **do**
  $i \leftarrow \text{ImportanceOrder}(k)$
  Mask the next important word: $a \leftarrow a^i_{\text{mask\_t}}$
  Compute teacher prediction: $l = \text{argmax}(f(a))$
  if $l \neq$ prediction **then**
    Set count if criterion met: count $\leftarrow k$
    return count
  end
end
return count

Algorithm 1: MASKLENGTH Computes the number of words that need masking to change the model prediction

D Importance Scores

D.1 Evaluating Importance Scores

In Algorithm 1, we provide the detailed steps for computing the mask length as used in the evaluation of importance scores.

Unlike ratios are computed using the pre-computed POS-tagged dictionary of token embeddings obtained as in Section C.

In Table 9, we show the top-3 importance scores supporting the prediction from the model being interpreted, obtained from LIME and MOXIE on the first 4 dev set examples where the model being interpreted makes an error (wrong label pre-

[5]The meaning and examples of the tags in the universal tagset can be found in the nltk book https://www.nltk.org/book/ch05.html

Text: the iditarod lasts for days - this just felt like it did . **Gold label:**-ve **Prediction:**+ve **LIME:** did (0.24), lasts (0.19), it (0.13) **MOXIE influence scores:** days (0.88), like (0.82), the (0.77) **MOXIE unlike ratios:** for (78.48), did (63.26), like (41.77)
Text: holden caulfield did it better . **Gold label:**-ve **Prediction:**+ve **LIME:** better (0.03), it (0.02), holden (0.01) **MOXIE influence scores:** better (0.97), . (0.93), did (0.93) **MOXIE unlike ratios:** holden (22.54), caulfield (17.61), better (12.36)
Text: you wo n't like roger , but you will quickly recognize him . **Gold label:**-ve **Prediction:**+ve **LIME:** recognize (0.20), but (0.19), will (0.14) **MOXIE influence scores:** quickly (0.98), but (0.93), n't (0.68) **MOXIE unlike ratios:** recognize (12.62), quickly (3.58), will (0.54)
Text: if steven soderbergh 's ' solaris ' is a failure^a it is a glorious failure^b . **Gold label:**+ve **Prediction:**-ve **LIME:** failure^a (-0.91), failure^b (-0.91), if (-0.03) **MOXIE influence scores:** failure^b (-1.00), a (-0.56), if (-0.36) **MOXIE unlike ratios:** failure^b (30.33)

Table 9: *Word importance scores when the model to be interpreted makes a wrong prediction* The top three scores supporting the model prediction obtained using LIME and MOXIE are shown for the first 4 dev set examples where the model being interpreted makes an error. For MOXIE, we show scores obtained using influence scores as well as unlike ratios. Superscripts are used to distinguish word positions if required.

diction). For MOXIE, we show importance scores obtained using both influence scores and unlike ratios. MOXIE scores are position independent and we assign the same scores to all occurrences of a word.

E Counterfactuals

E.1 Example Counterfactual Predictions

In Table 10, we show selected examples of counterfactual predictions. The examples have been picked from the first 10 dev set examples.

Text (Prediction)	Replacement Prediction
unflinchingly **bleak** and desperate (-ve)	sensual
it 's **slow** – very , very slow . (-ve)	enjoyable
a sometimes **tedious** film (-ve)	heart-breaking

Table 10: *Example counterfactual predictions* selected from the first 10 examples of the dev set. The highlighted words in the left column indicate the words which are replaced with the words in the right column.

E.2 Computing Counterfactual Accuracy

In Algorithm 2, we provide the detailed steps for computing counterfactual accuracy for a context as used in evaluating counterfactual predictions. Pre-computed POS-tagged dictionary of token embeddings are obtained as in Section C.

The median size and median accuracy when selecting top-10 tokens (as done in Algorithm 2) are 90.0 and 10.0 respectively. If we don't do any selection, the median size and median accuracy are 72.0 and 63.41 respectively.

F Biases

F.1 Filtering Contexts for Analyzing Biases

Here, we detail the steps used to filter the contexts from the input dataset below when probing with adjectives as control/probe words:

1. Get teacher model predictions on each example.
2. Tokenize and get a POS tag for each example in the input dataset.
3. Select contexts (an example with a marked token position) with adjective POS tag. This could lead to none, one or more contexts per example.
4. Select contexts for which teacher model predictions (on the corresponding example) are positive.
5. Remove contexts for which the student model predicts negative for at least one replacement from the set {*immense*, *marvelous*, *wonderful*, *glorious*, *divine*, *terrific*, *sensational*, *magnificent*, *tremendous*, *colossal*} and positive for at least one replacement from the set {*dreadful*, *terrible*, *awful*, *hideous*, *horrid*, *horrible*}.
6. Additionally, remove contexts for which the student model predictions never change when the marked token is replaced by another word with the same POS tag.

Again, we use nltk's aver-

Input: A text sequence x: $x_1 x_2 ... x_n$
Input: Location in the sequence i
Input: Precomputed token embeddings E_V with words of the same POS tag as x_i
Output: Size and accuracy of generated counterfactuals
Compute teacher prediction: prediction $\leftarrow \text{argmax}(f(x))$
Compute context embedding: $z_c = A_C(x^i_{\text{mask\_t}})$
Compute predictions for each token in the vocabulary: $y_V = C(A_M(z_c, E_V))$
Sort according to the probability of differing from teacher prediction, i.e., using $(1 - y^j_V[\text{prediction}])$, to get the list V_{sorted}
Select up to 10 tokens from the top of the list that differ from teacher prediction: $\text{argmax}(y^j_V) \neq \text{prediction}$, to get the list V_{selected}
Initialize: size $\leftarrow 0$; correct $\leftarrow 0$
for $k \leftarrow 1$ ***to*** $|V_{\text{selected}}|$ **do**
  size $\leftarrow$ size $+ 1$
  $w \leftarrow V_{\text{selected}}[k]$
  $a \leftarrow x$ Replace the i-th token with word w: $a[i] \leftarrow w$
  Compute teacher prediction: $l \leftarrow \text{argmax}(f(a))$
  if $l \neq$ prediction **then**
    correct $=$ correct $+ 1$;
  end
end
if count $= 0$ **then**
  return $0, 0$
end
acc $\leftarrow 100.0 *$ correct/count
return count, acc

Algorithm 2: COUNTERFACTUAL_ACC Computes the accuracy of generated counterfactuals

aged_perceptron_tagger for obtaining POS tags, and use the universal_tagset. For Step 6, we used the pre-computed POS-tagged dictionary of token embeddings as obtained in Section C.

There were a total of 81435 adjective contexts in the training dataset. The size of the filtered set was 29885.