The Reverse Turing Test for Evaluating Interpretability Methods on Unknown Tasks

Gonzalez, Ana Valeria; Søgaard, Anders

Publication date: 2020

Document version
Early version, also known as pre-print

Document license: Other

Citation for published version (APA):
The Reverse Turing Test
for Evaluating Interpretability Methods on Unknown Tasks

Ana Valeria Gonzalez and Anders Søgaard
Department of Computer Science
University of Copenhagen
{ana,soegaard}@di.ku.dk

Abstract

The Turing Test evaluates a computer program’s ability to mimic human behaviour. The Reverse Turing Test, reversely, evaluates a human’s ability to mimic machine behaviour in a forward prediction task. We propose to use the Reverse Turing Test to evaluate the quality of interpretability methods. The Reverse Turing Test improves on previous experimental protocols for human evaluation of interpretability methods by a) including a training phase, and b) masking the task, which, combined, enables us to evaluate models independently of their quality, in a way that is unbiased by the participants’ previous exposure to the task. We present a human evaluation of LIME across five NLP tasks in a Latin Square design and analyze the effect of masking the task in forward prediction experiments. Additionally, we demonstrate a fundamental limitation of LIME and show how this limitation is detrimental for human forward prediction in some NLP tasks.

1 Introduction

Machine learning models have tremendous impact on our daily lives, from information storing and tracking (i.e. Google Search and Facebook News Feed), as well as on other scientific disciplines. Modern-day NLP models, for example, are complex neural networks with millions or billions of parameters trained with multiple objectives and often in multiple stages (Devlin et al., 2019; Raffel et al., 2019); they are often seen for that reason, as black boxes whose rationales cannot easily be queried. In other words, we are increasingly relying on models that we do not understand or cannot explain, in science, as well as in our daily lives. Model interpretability, however, is desired for several reasons: Humans often ask for the motivation behind advice, and in the same way, users are likely to trust model decisions more if they can ask for the rationale behind them. Model interpretability enables us to inspect whether models are fair and unbiased, and it enables engineers to detect when models rely on mere confounds. Combating this type of overfitting will lead to more robust (or less error-prone) decision making with better generalization to unseen data (and, hence, safer model employment).

Recent years has seen a surge in work on post-hoc interpretability methods for neural networks which aim to approximate complex decision boundaries with less complex models, for example, locally linear models. See §5 in Murdoch et al. (2019) for a brief survey. Unfortunately, there is little consensus on how to compare interpretability methods. Some benchmarks have been introduced (Rei and Søgaard, 2018; Poerner et al., 2018; DeYoung et al., 2020), but some of these are flawed, and they are all only applicable to some of the proposed interpretability methods. See §5 for discussion. In our view, a more promising approach to evaluating interpretability methods is by human forward prediction experiments. Nguyen (2018) presented the first evaluations of LIME (Ribeiro et al., 2016) for sentiment analysis using human subjects through a series of Mechanical Turk experiments. Their
study had two limitations: (a) They did not allow for a training phase for the human participants to learn model idiosyncrasies, and participants instead had to rely on the assumption that the model was near-perfect. (b) Since the participants thus had to rely on their own sentiment predictions, their evaluations are biased by their beliefs about the sentiment of the input documents. Hase and Bansal (2020) recently presented evaluations of LIME with human participants that involved a training phase, enabling them to predict poor model behavior, and thereby addressing limitation (a), but they still only included known tasks for which forward prediction is biased by the participants’ own beliefs. This paper aims to fill this gap.

Contributions This work presents a simple-yet-insightful method for evaluating interpretability methods through simple 15-minute experiments with human participants. Our experiments differ from previous work in a very important way: our proposed evaluation of interpretability involves conditions in which human subjects are less likely to rely on their cognitive biases. As our test case, we evaluate LIME (Ribeiro et al., 2016) – which has been a popular feature attribution method in the last few years – across five NLP tasks in a Latin Square design (Shah and Sinha, 1989), including three tasks which were kept secret to our participants. We argue that keeping the tasks secret to the participants makes the evaluation of interpretability methods more reliable and investigate the impact of this difference in experimental design. Additionally, we also point out a weakness of LIME – which is shared across many word attribution methods – namely, that its input/output dimensions are sometimes orthogonal to the relevant dimensions for interpretability. We include a task in which this happens and show how detrimental the interpretability method can be in such cases.

2 Human Biases in Forward Prediction

One thing sets our experiments in this paper apart from previous evaluations of interpretability methods by A/B testing with human forward prediction (Nguyen, 2018; Hase and Bansal, 2020): We will present participants with decisions by models trained on tasks that are unknown to the participants. In other words, humans are simply asked to predict $y$ from $x$, with no prior knowledge of the relation that may exist between them, beyond an initial training phase. Several different cognitive biases are particularly important for motivating and analyzing our experimental design:

**Belief bias** An effect where someone’s evaluation of the logical strength of an argument is biased by the plausibility of the conclusion (Klauer et al., 2000). In human forward prediction of model behavior, this happens when the plausibility of the conclusion, e.g., this review is positive, biases the subject’s evaluation of her own conclusions, e.g., the model will predict this review is negative, because it includes this or that term. We argue that it is particularly important to evaluate interpretability methods with human forward prediction on unknown tasks to avoid belief bias.

**Confirmation bias** This bias occurs when individuals seek information which supports their prior belief while disproportionately disregarding information that challenges this belief (Mynatt et al., 1977). In our context, such a bias could, for example, lead subjects that already classified a document in one way to disregard LIME mark-up. In the extreme, confirmation bias could cancel out any effect of interpretability methods in human forward prediction, but our results below show that in practice, LIME has a strong (positive or negative) effect on human forward prediction.

**Curse of knowledge** This is the phenomenon when better-informed people find it extremely difficult to think about problems from the perspective of lesser-informed people (Ackerman et al., 2003). In our case, the model plays the role of a lesser-informed agent. We believe the curse of knowledge amplifies belief bias and makes it very hard for participants to unlearn their prior knowledge of the underlying task relation. This bias is very evident in our experiments below and additional motivation for including a training phase in the Reverse Turing test (see our Pre-Experiment).

Our experimental design is motivated by a desire to reduce the above biases in our forward prediction experiments. Cognitive biases can interact with human forward prediction in a number of ways, e.g., making participants less confident about predictions that do not align with their prior beliefs, or leading them to ignore explanations that are inconsistent with their beliefs.
3  LIME – and its Limitations

The Local Model-agnostic Explanations (LIME) method (Ribeiro et al., 2016) has become one of the most widely used post-hoc model interpretability methods in NLP. LIME aims to interpret model predictions by locally approximating a model’s decision boundary around an individual prediction. This is done by training a linear classifier on perturbations of this example.

Several weaknesses of LIME have been identified in the literature: LIME is linear (Bramhall et al., 2020), unstable (Elshawi et al., 2019) and very sensitive to the width of the kernel used to assign weights to input example perturbations (Vlassopoulos, 2019, Kopper, 2019), an increasing number of features also increases weight instability (Gruber, 2019), and Vlassopoulos (2019) argues that with sparse data, sampling is insufficient. Laugel et al. (2018) argues the specific sampling technique is suboptimal.

We point to an additional, albeit perhaps obvious, weakness of LIME’s: It can only explain the decisions of a classifier in so far as the decision boundary of the classifier aligns with the feature dimensions of LIME. In most applications of feature attribution interpretability to NLP problems, the feature dimensions are the input words. That is to say, such methods can only explain the decisions of a classifier if the decision boundary aligns with the dimensions along which the occurrences of words are encoded. LIME can, for example, not explain the decisions of a classifier “1 if sentence length odd, else 0”. In our experiments, we include a task in which a classifier is trained to predict the length of the input sentence (from a low-rank representation), as a way of evaluating the effect of LIME on human forward prediction, on tasks that LIME is, for this reason, not able to explain.

Examples of real tasks where this limitation is a problem, include, for example, all tasks where sentence length is predictive, including readability assessment (Kincaid et al., 1975), authorship attribution (Stamatatos, 2009), or sentence alignment (Brown et al., 1991). We note this limitation is not unique to LIME, but shared among most post-hoc interpretability methods which output word or span importance, e.g., hot flip (Ebrahimi et al., 2018), attention (Rei and Søgaard, 2018), and back-propagation (Rei and Søgaard, 2018). Other approaches to interpretability such as using influence functions (Koh and Liang, 2017) may have more explanatory power for such problems however we choose to focus our experiments on LIME, as a vast number of interpretability methods return explanations which are extractive similarly to this method.

4  Human Forward Prediction Experiments

The experiments we describe below are examples of the Reverse Turing Test. The test resembles the Turing Test (Turing, 1950; Horn, 1995) in that it focuses on the differences between the behavior of humans and computer programs. In the Reverse Turing Test, we quantify humans’ ability to simulate computer programs, however, rather than computer programs’ ability to simulate humans. Specifically, we quantify humans’ ability to predict the output of machine learning models given previously unseen examples. The test is defined (for classification models) as follows: The Reverse Turing test is an experimental protocol according to which participants are presented with $k$ examples of $(\hat{x}, y)$ pairs, with $y = f(\hat{x})$ the labeling of $x$ by some unknown machine learning model $f(\cdot)$, and $\hat{x}$ is a possibly empty interpretation function, which, in the case of post-hoc interpretability methods, highlights parts of the input, e.g., input words. The training phase is timed. Subsequently,
participants are presented with \( m \) unseen examples \( x_1 \ldots x_m \) and asked to predict \( f(x_1) \ldots f(x_m) \). The evaluation phase is also timed. The result of the Reverse Turing test is the accuracy or \( F_1 \) of the participants’ predictions compared to \( \hat{y}_1, \ldots, \hat{y}_m \), as well as the training and inference times. The test is meant to evaluate the quality of different interpretations, \( I(\cdot) \) and can be used for evaluation methods, like we do, or for evaluating models or interpretability methods during development (Lage et al., 2018). We believe our test is in some ways more critical than previous, as we are attempting to evaluate interpretability methods more reliably by reducing human belief bias.

4.1 Tasks and Data

Based on the efforts of 30 annotators, we collected a total of 3000 example annotations in human forward prediction experiments, distributed across five different tasks (two known; three unknown) and two experimental conditions (with and without explanations). The overall experimental protocol is shown in Figure 1. All code for preprocessing data, training the models, and the experimental set ups are publicly available at https://github.com/anavaleriagonzalez/reverse_turing_test.

**Known Tasks** For our known tasks, we focus on two very common text classification tasks: sentiment analysis and hate/offensive speech detection. For sentiment analysis we use the Stanford Sentiment Treebank (SST) (Socher et al., 2013). The SST dataset consists of 6920 documents for training, 872 documents for development and 1820 documents for testing. For hate speech detection, we use the HatEval dataset from SemEval 2019 (Basile et al., 2019). The dataset consists of several binary tasks, however we focus on the task of detecting presence of hate speech (disregarding which group is being targeted as this is considered a separate task). In total, there are 9000 tweets for training, 1000 for development and 3000 for testing.

**Unknown Tasks** As our unknown tasks, we use 3 of the 10 probing tasks introduced in Conneau et al. (2018). The probing tasks were originally designed to evaluate the linguistic properties of sentence embedding models. In this study we are mostly interested in the differences in performance between humans and machines, and are not looking to evaluate linguistic properties of representations in depth, therefore chose only a few of these tasks. The first task is *sentence length prediction* in which the sentences are grouped in 6 bins indicating length in terms of number of words. This task was chosen in order to examine the effect on LIME in a task where LIME offers poor explanations. The second probing task is *tense prediction*, which involves predicting whether the verb in the main clause is present or past tense. The third task is *subject number prediction*, which focuses on predicting whether the subject in the main clause is plural or singular. These last two are simple tasks where we expect LIME to offer good enouh explanations. The training data for each of the probing tasks consists of 100k sentences, 10k sentences for validation and 10k sentences for testing. The sentences are taken from the Toronto Book Corpus (Zhu et al., 2015). More details on data extraction can be found on Conneau et al. (2018).

4.2 Classification Model

For training sentiment and hate speech classifiers, we pass as our input pretrained BERT representations (Devlin et al., 2019) through an LSTM layer (Hochreiter and Schmidhuber, 1997) \( (d = 100) \) followed by a multi-layered perceptron with a single hidden layer \( (d = 100) \). We use a learning rate of 0.001 and Adam optimizer. The hyper-parameters were not tuned for optimal performance. We use the same architecture for all tasks, except for sentence length prediction. For the sentence length prediction task, we use BERT token representations and pass them through a mean pooling layer followed by a multi-layered perceptron with a single hidden layer \( (d = 100) \). Both models are trained for 20 epochs. Note also that we do not fine-tune the BERT representations. This, together with our hyper-parameters, gives us suboptimal performance, especially on the known tasks, but this was done on purpose to make our predictions different from the gold labels for the known tasks, in order to make it possible to quantify participants’ belief bias: If results are too close to human performance, it would not be possible to distinguish human forward prediction performance with respect to model predictions from human performance with respect to predicting the true class. Our performance on the unknown probing tasks is comparable to the results in Conneau et al. (2018).
We include 3 unknown tasks, meaning that no information about the tasks was provided to the participants in advance of the experiment. Instead, subjects had to try to infer patterns from the data sample, possibly augmented with LIME explanations. For the known tasks, we follow [Nguyen (2018) and Hase and Bansal (2020)] and provide subjects with a brief explanation of the task, but emphasize the fact that the participants should predict model decisions, not the true labels; and hence, they should avoid being influenced by their own beliefs of whether a text is positive or an instance of hate speech. As in [Hase and Bansal (2020)], we make sure that true positives, false positives, true negatives, and false negatives are balanced across the training and test data. In total we have 30 participants, all with at least undergraduate education and some knowledge of computer science and machine learning. We collect 3000 human forward predictions: 1800 from training sessions and 1200 from the evaluation sessions. For each condition and item in the evaluation set, we have at least two human forward predictions. Some of the participants gave us optional feedback on strategies they used. This, together with the fact that the participants should predict model decisions and present participants with the original LIME output images, after turning them into grayscale in order to avoid revealing the class label. We rely on 500 perturbations of each data sample in order to obtain the top 3 most informative input tokens. See Figure 2 for an example of the visual stimuli under this condition. The training sessions are interactive, simulating the test interface, but providing the true answer whenever the participant has provided an initial guess. We shuffle the training sessions at random. The evaluation sets for each task consist of 45 samples in total, split into chunks of 15: \( t_{e_1}, t_{e_2}, t_{e_3} \). In the evaluation session, subjects are not provided with the true model responses, to avoid biases from additional training. We divide our participants in three groups, and for each task, the groups are assigned task subsamples in the following Latin Square design:

<table>
<thead>
<tr>
<th>Subjects</th>
<th>( x )</th>
<th>LIME(( x ))</th>
<th>Control(( x ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects(_\text{1})</td>
<td>( t_{1_1}, t_{1_2}, t_{1_3}, t_{3_1}, t_{3_2}, t_{3_3} )</td>
<td>( t_{2_1}, t_{2_2}, t_{2_3} )</td>
<td>( t_{3_1}, t_{3_2}, t_{3_3} )</td>
</tr>
<tr>
<td>Subjects(_\text{2})</td>
<td>( t_{1_1}, t_{1_2}, t_{1_3}, t_{3_1}, t_{3_2}, t_{3_3} )</td>
<td>( t_{2_1}, t_{2_2}, t_{2_3} )</td>
<td>( t_{3_1}, t_{3_2}, t_{3_3} )</td>
</tr>
<tr>
<td>Subjects(_\text{3})</td>
<td>( t_{1_1}, t_{1_2}, t_{1_3}, t_{3_1}, t_{3_2}, t_{3_3} )</td>
<td>( t_{2_1}, t_{2_2}, t_{2_3} )</td>
<td>( t_{3_1}, t_{3_2}, t_{3_3} )</td>
</tr>
</tbody>
</table>

We include 3 unknown tasks, meaning that no information about the tasks was provided to the participants in advance of the experiment. Instead, subjects had to try to infer patterns from the data sample, possibly augmented with LIME explanations. For the known tasks, we follow [Nguyen (2018) and Hase and Bansal (2020)] and provide subjects with a brief explanation of the task, but emphasize the fact that the participants should predict model decisions, not the true labels; and hence, they should avoid being influenced by their own beliefs of whether a text is positive or an instance of hate speech. As in [Hase and Bansal (2020)], we make sure that true positives, false positives, true negatives, and false negatives are balanced across the training and test data. In total we have 30 participants, all with at least undergraduate education and some knowledge of computer science and machine learning. We collect 3000 human forward predictions: 1800 from training sessions and 1200 from the evaluation sessions. For each condition and item in the evaluation set, we have at least two human forward predictions. Some of the participants gave us optional feedback on strategies they used. This, as well as some examples of our interface can be found in the Appendix.

4.4 Pre-Experiment: The Effect of Training on Forward Prediction

In addition to our main experiment with 30 participants, we also performed a human forward prediction pre-experiment with a single participant. In the pre-experiment we compare human forward prediction with and without training; we do so to motivate our experimental design, in which we follow [Hase and Bansal (2020)], but depart from [Nguyen (2018)], in including a training phase in which humans can learn the idiosyncracies of the machine learning model. In the pre-experiment, we only explore the effects of the training phase for the known tasks. We first ran the experiment without training; then ran the experiment with training. To clearly be able to quantify the effect of our interactive training phase, we only use examples with false model predictions in the pre-experiment. For each of the two tasks, sentiment analysis and hate speech detection, we use: (a) 20 distinct examples for evaluation for each of the two conditions; and (b) 25 distinct examples for training for the second experimental condition. Note that since we only use a single human participant in the

...
Table 1: RESULTS FROM MAIN EXPERIMENT. Columns 1–2: accuracy of human forward prediction results on plain input \((x)\) or augmented with LIME interpretations \(\text{LIME}(x)\). \(^*\): Significance of \(\alpha < .05\) computed with Mann-Whitney \(U\) test. Columns 3–4: average duration of evaluation sessions \(\text{human inference time}\). Column 5 lists the model accuracies with respect to human gold annotation; which we compare with human accuracies with respect to human gold annotation.

<table>
<thead>
<tr>
<th>Task</th>
<th>(\text{HUMAN ACC.}(\hat{p}))</th>
<th>(\text{HUMAN TIME}(\hat{p}))</th>
<th>(\text{MODEL ACC.}(p))</th>
<th>(\text{HUMAN ACC.}(p))</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST</td>
<td>0.557</td>
<td>(0.694)</td>
<td>03:00 (\text{01:50})</td>
<td>0.822 (0.767)</td>
</tr>
<tr>
<td>HatEval2019</td>
<td>0.562</td>
<td>(0.715)</td>
<td>02:18 (0\text{1:10})</td>
<td>0.573 (0.706)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNKNOW Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent Len</td>
</tr>
<tr>
<td>Subj Number</td>
</tr>
<tr>
<td>Tense</td>
</tr>
</tbody>
</table>

The effect of training is positive. On the SST dataset, accuracy with respect to model predictions \((\hat{p})\) increases from \(0.400\) to \(0.550\); on the HatEval2019 dataset, performance increases from \(0.3690\) to \(0.526\). We see this as a very strong motivation for including a training phase. A training phase also makes it possible to perform human forward prediction experiments on tasks that are unknown to the participants, removing any belief bias that may otherwise affect results. We note that a training phase does not necessarily lead to faster inference times. On HatEval, average inference time was reduced from \(08:56\) to \(07:21\), but on STS, it increased from \(06:24\) to \(08:53\). This suggests that untrained annotators (after a few instances) learn superficial heuristics that enable them to draw fast, yet inaccurate, inferences.

4.5 Main Experiment: The Effect of LIME on Forward Prediction

We report the results of our main experiment in Table 1. Results show that LIME helps, both in terms of accuracy and time, on known and unknown target tasks, except when the decisions boundary does not align with LIME dimensions \(\text{Sent Len}\) (columns 1–4); and that while humans are biased by their beliefs and knowledge of the known tasks, they are not biased during unknown tasks, which can be seen by their decrease in accuracy with respect to human annotation. We make the following observations:

The Effect of LIME on Known Tasks This is the standard set-up considered also in previous work \([\text{Nguyen} \ 2018, \ Hase \ and \ Bansal \ 2020]\); see columns 1–2 and rows 1–2 in Table 1. We see that LIME leads to significantly better human forward prediction performance on both tasks. It also leads to (statistically) significantly faster inference times, approximately halving the time participants spend on classifying the test examples. This shows that LIME, in spite of its limitations \((\S 3)\), is a very useful tool in some cases.

The Effect of LIME on Unknown Tasks The effect of LIME on human forward prediction accuracy on 2 of the unknown tasks is not significant. On the two tasks, where LIME provides meaningful explanations (subject number and tense prediction), LIME does lead to smaller reductions in inference time which are not statistically significant. The effect on the participants’ accuracy is mixed and insignificant. In addition, LIME is significantly detrimental on human forward prediction accuracy for the task of sentence length prediction; it also leads to longer inference times, although this difference was not statistically significant. This shows that while LIME is useful in some cases, this is not always the case. We speculate that since LIME explanations are partial, they are only

Note that our human participant, without training had lower-than-random accuracy in both tasks. This is not surprising, since we have selected data points on which our model was wrong. Under the influence of belief bias, humans are likely to also classify these wrongly.
effective when supplemented by (approximately correct) belief bias. If true, this suggests that LIME, even for the tasks that can be explained in terms of input words, is, nevertheless, only applicable to tasks that humans have experience with, and when the underlying models perform reasonably well.

**Known and Unknown Tasks** In general, our participants are much slower at classifying examples when the task is unknown. This shows the efficiency of the belief biases our participants have in sentiment analysis and hate speech classification. The effectiveness of these biases is also demonstrated by the performance gaps between humans and models when comparing their predictions to ground truth labels. To see this, consider columns 5–7 in Table 1. Participants, while instructed to predict model output (\( \hat{p} \)), actually significantly outperform our classifier in predicting the true labels (0.706 vs. 0.573)! In contrast, participants perform subject number and tense prediction at chance levels (0.491 and 0.500), while a simple classifier achieves accuracy greater than 0.9 on both tasks. This clearly demonstrates belief bias in human forward prediction experiments.

**Human Inference Time** In addition to considering performance, we also recorded the time our participants spent on completing the forward prediction tasks. We present the average times of each condition in Table 1 with shorter times bolded. We used the Mann-Whitney U test to determine significance for these, which is also shown in the same table. We plot the total averages in Figure 3. All the results shown in the plot are significant with \( \alpha < 0.001 \).

**5 Related Works**

**Interpretability methods** Interpretability methods come in different flavors: (a) post-hoc analysis methods that estimate input feature importance for decisions, including LIME, (b) post-hoc analysis methods that estimate the influence of training instances on decisions, e.g., influence functions [Koh and Liang (2017)] and (c) strategies for making complex models interpretable by learning to generate explanations [Narang et al. (2020)] or uptraining simpler models [Agarwal et al. (2020)]. In this paper we have focused on post-hoc interpretability methods, but it is equally important, we argue, to evaluate other types of interpretability methods on unknown tasks, when running human forward prediction experiments, to avoid participants’ cognitive biases.

**Intrinsic evaluation of interpretability methods** One standard approach to evaluating explanations is to remove the parts of the input detected by the interpretability method and see whether classifier performance degrades [ Samek et al. (2017)]. One drawback of this method is that the corrupted examples are now out-of-distribution, and classifiers will generally perform worse on such examples. Hooker et al. (2019) improve on this by evaluating classifiers retrained on the corrupted examples. This approach, however, now suffers from another drawback: If classifiers perform well on the corrupted examples, that does not mean the interpretability methods were wrong. Jain and Wallace (2019) evaluate attention functions as explanations and argue that they do not provide useful explanations, in part because they do not correlate with gradient-based approaches to determining feature importance; Wiegreffe and Pinter (2019), in return, show this test is not sufficient to show attention functions do not provide useful explanations.

To see this, consider a sparsity-promoting classifier relying on a single feature \( f \) in the context of feature swamping [Sutton et al. (2006)], i.e., frequent features may lead to undertraining of covariate features in discriminative learning. If \( f \) is removed, but the classifier retains its original performance by now relying on covariate features, that does not mean the classifier did not solely rely on \( f \) when trained on the original data.
Extrinsic benchmarks for interpretability methods. Rei and Søgaard (2018) show how token-level annotated corpora can be converted to benchmarks for evaluating post-hoc interpretability methods. They train sentence classifiers to predict whether sentences contain labels or not, and use interpretability methods to predict what input words were important, and use the $F_1$ score of those predictions to evaluate the interpretability methods. Their method, however, only works as an evaluation of interpretability methods under the assumption that the classifier is near-perfect (since otherwise the token-level annotations cannot be assumed to be explanations of model decisions); furthermore, it is only applicable to tasks for which we have token-level annotations. Poerner et al. (2018) adopt a slightly different approach, augmenting real documents with random text passages to see whether interpretability methods focus on the original text passages. This method suffers from the same drawback, that it assumes near-perfect performance. It is also only designed to capture false positives; it cannot distinguish between true or false negatives. Finally, DeYoung et al. (2020) recently introduced ERASER, a suite of NLP datasets augmented with rationales, including reading comprehension, natural language inference, and fact checking. ERASER also assumes near-perfect performance, and can be seen as extending the set of tasks for which the method proposed in Rei and Søgaard (2018), is applicable. Our method, in contrast, is independent of model quality.

Human evaluation of explanations. The idea of evaluating explanations by testing human participants’ ability to predict model decisions with and without explanations is not novel. Nguyen (2018), Lage et al. (2018) and Hase and Bansal (2020), as already discussed, present such experiments. Schmidt and Biessmann (2019) is another example of human forward prediction experiments in a crowdsourcing platform. They perform experiments on the effect of LIME and COV AR on human forward prediction for a sentiment task that is known to be participants, in advance. Our criticism of Nguyen (2018) also applies to their study. Narayanan et al. (2018) also present evaluations of interpretability methods with humans; they design simple tasks in which humans verify whether an output is consistent with an input and an explanation. The human participants are provided with explanations of what the tasks are, and they only consider a handful of input features.

The Reverse Turing Test that we propose here is different from previous proposals to use human forward prediction to evaluate interpretability methods, in that it a) includes a training phase which is important for subjects to learn model nuances and which in turn, allows us to b) include human forward prediction on unknown tasks, i.e., tasks about which they have no prior beliefs. We are, to the best of our knowledge, the first to propose such a protocol. In the above experiments, designed to motivate the design of the Reverse Turing Test, we see the limitations of a widely used interpretability method, LIME. On some tasks, i.e., tasks which cannot be explained by the occurrence of input words, the effect of LIME is detrimental; and on unknown tasks, for which LIME interpretations are not supported by participants’ cognitive biases, its effect on human forward prediction is insignificant. Overall, our experiments show that our proposed design offers interesting insights into the role that cognitive biases play in the evaluation of interpretability, and propose that such a set up be used in further research to explore the effect of cognitive biases for other interpretability methods which provide final rationales similar to those provided by LIME.

6 Conclusion

We presented an evaluation protocol for interpretability methods, which differs from previous work by including a training phase and by including unknown tasks. This makes our protocol work independently of model quality, and controls for belief bias. Using LIME as our test case, we find that on known tasks, LIME leads to statistically significant improvements in human forward prediction, both in accuracy and inference time. However, when tasks are unknown, differences are no longer significant. We see this as evidence of bias in the standard protocols, and argue that making tasks unknown, leads to more reliable evaluations. We also identify tasks, where model decisions cannot be explained in terms of input word occurrences, and for which the effect of LIME is detrimental for human forward prediction performance.

3[http://www.eraserbenchmark.com/]

8
References

Mark Ackerman, Volkmar Pipek, and Volker Wulf, editors. 2003. Sharing expertise beyond knowledge management. MIT Press.


A  Presentation of stimuli

We created a web application using Flask\footnote{https://flask.palletsprojects.com/en/1.1.x/} in order to collect participant data. Participants would get assigned a known or unknown task and LIME explanations or no explanations. For all tasks we provide the same general instructions. See top of Figure 4 for a screenshot of our general instructions. In addition, we had task specific instructions. For known tasks we provided short descriptions of the task, while emphasizing the fact that subjects should imitate the model rather than follow their own opinions about the true labels. For unknown tasks, we provided instructions as seen in Figure 4.

Figure 4: Example of the instructions presented to the participants. The participants could get a secret task or one of the known tasks, as well as LIME explanations or no explanations.

The training and evaluation sessions were almost the same, with the only difference being that during training, subjects could check the model’s answer after making an initial guess. See Figure 5 for an example of what the items looked like. The example here is for the task of sentence length prediction using LIME explanations.

Figure 5: (a) Example of item in the training session for sentence length prediction. Note that the participants are able to check the model answer (b) Example of item in the evaluation session for sentence length prediction. Here the participants are no longer able to check the model answer.

B  Subject Feedback

As an optional part of our tests, subjects provided some insight into the strategies they came up with or troubles they had when solving a task. We only had this feedback from some of the participants, which can be found in Table\ref{table:feedback}.
<table>
<thead>
<tr>
<th>Explanation</th>
<th>Task</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Binary</td>
<td>Tried to identify what kinds of data the ML model fails</td>
</tr>
<tr>
<td>lime</td>
<td>Binary</td>
<td>1. Read the sentence. 2. Paid attention to the shaded words: if the overall sentiment of these words was clear I assumed the model would classify them accordingly. Otherwise I tried to consider how easy it would be for the model to understand the compositional meaning of the sentence assuming it will make mistakes at phenomena involving ironies or comparisons to proper names etc.</td>
</tr>
<tr>
<td>none</td>
<td>Binary</td>
<td>Keywords, the sentimental polarity of the sentence</td>
</tr>
<tr>
<td>lime</td>
<td>Hateval</td>
<td>only look at highlighted words</td>
</tr>
<tr>
<td>none</td>
<td>Hateval</td>
<td>logical</td>
</tr>
<tr>
<td>lime</td>
<td>Sent Len</td>
<td>Haven’t got the faintest idea.</td>
</tr>
<tr>
<td>none</td>
<td>Sent Len</td>
<td>I was very lost in this task. I could not find topics in the sentences so I tried to focus whether sentences contained similar words guessing that these would be mapped to the same class...</td>
</tr>
<tr>
<td>lime</td>
<td>Tense</td>
<td>no clue</td>
</tr>
<tr>
<td>none</td>
<td>Tense</td>
<td>1st Person 1 Person vs 2nd Person Multiple participants</td>
</tr>
</tbody>
</table>

Table 2: Feedback on strategies found by participants. Writing a strategy was not mandatory therefore we do not have written feedback from every participant.