universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

An Evaluation of Explanation Methods for Detectors of Machine-Generated Text

verfasst von | submitted by

Loris Schoenegger B.Sc.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 935
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Medieninformatik
Betreut von | Supervisor:	Univ.-Prof. Dr. -Ing. Benjamin Roth B.Sc. M.Sc.

Acknowledgements

I want to thank Benjamin Roth and Yuxi Xia for supervising this thesis. I also wish to express my gratitude to all the participants in the user study and to everyone who helped organize it.

Abstract

The behavior of language-model-based detectors of machine-generated text and the features they might leverage for classification are not yet fully understood. To obtain insights into the behavior of these detectors, one can apply the explanation methods SHAP, LIME, and Anchor. These local explanation methods indicate which parts of an input might be used by a detector for prediction. However, the quality of the resulting explanations has not been assessed in detail before. This thesis conducts the first systematic evaluation of explanation quality for this task. The dimensions of *faithfulness* and *stability* are assessed with five automated metrics, and *usefulness* is evaluated in a user study. SHAP, LIME and Anchor explanations are generated for decisions of three existing language-model-based detectors on a dataset of ChatGPT-generated and human-written documents.

SHAP performs best in terms of faithfulness and stability. LIME, perceived as most useful by users, scores the worst in terms of user performance at predicting the detectors' behavior. Anchor, perceived as least useful by users, outranks LIME in terms of performance in the user study. LIME and Anchor each fail to surpass a random baseline on faithfulness, while SHAP surpasses the random baseline in both faithfulness experiments.

Kurzfassung

Das Verhalten sprachmodellbasierter Detektoren für maschinell generierten Text und die Merkmale, die sie zur Klassifizierung nutzen könnten, sind noch nicht vollständig erforscht. Um Erkenntnisse über das Verhalten dieser Detektoren zu gewinnen, können die Erklärungsmethoden SHAP, LIME und Anchor angewendet werden. Diese Methoden können darüber Aufschluss geben, welche Textstellen für die Entscheidung herangezogen wurden. Die Qualität der entstandenen Erklärungen wurde jedoch bisher nicht im Detail beurteilt. Diese Arbeit führt die erste systematische Bewertung solcher Erklärungen im Kontext dieser Problemstellung durch. Die Dimensionen *faithfulness* (originalgetreue Abbildung des Entscheidungsprozesses) und *stability* (ausreichende Sensitivität und ausreichend deterministisches Verhalten) werden mit fünf automatisierten Metriken bewertet. *Usefulness* (Nützlichkeit) wird mit Versuchspersonen evaluiert. SHAP-, LIME- und Anchor-Erklärungen werden für Entscheidungen dreier sprachmodellbasierter Detektoren erstellt. Hierfür wird ein Datensatz aus ChatGPT-generierten und von Menschen verfassten Dokumenten verwendet.

SHAP schneidet hinsichtlich faithfulness und stability am besten ab. LIME und Anchor verfehlen jeweils eine Baseline in den zwei durchgeführten Experimenten zur faithfulness. LIME wird von den Teilnehmenden als am nützlichsten empfunden, führt jedoch nicht zu messbar besserem Abschneiden in einem Versuch, in dem das Verhalten des Detektors vorhergesagt werden soll. Anchor, von den Versuchspersonen als am wenigsten nützlich empfunden, übertrifft LIME in dieser Hinsicht.

Contents

List of Tables

List of Figures

1. Introduction

Large language models, such as ChatGPT, produce output that is often virtually indistinguishable from human-written text. Their ability to generate human-like text at an unprecedented scale allows for new forms of phishing, disinformation campaigns, and academic fraud (Crothers et al., 2023). Recent work has proposed language-model-based detection methods for machine-generated text (Solaiman et al., 2019; Guo et al., 2023; Mitchell et al., 2023). These operate as black-box detectors: they provide no explanations for their decisions. This is insufficient for applications that demand additional evidence or when wrong decisions can result in major consequences, as would be the case for detecting machine-generated text in academia.

To address this, **explanation methods** have been applied to such detectors (Mosca et al., 2023; Liu et al., 2023c; Yu et al., 2023), with the majority of papers using SHAP (Lundberg and Lee, 2017) or LIME (Ribeiro et al., 2016). Anchor (Ribeiro et al., 2018), a method that produces rule-based explanations, is first applied to these detectors in this thesis. All three of these explanation methods produce local explanations that explain single predictions (machine-generated or not) by locating relevant elements in the input (words in the document that influenced the prediction).

This thesis aims to test three aspects that are particularly relevant in this context: Explanations should accurately depict the detector's behavior (**faithfulness:** Jacovi and Goldberg, 2020; Ribeiro et al., 2016; Alvarez Melis and Jaakkola, 2018). They should be sensitive enough and sufficiently deterministic (**stability**: Alvarez Melis and Jaakkola, 2018; Lakkaraju et al., 2020; Nauta et al., 2023), and be effective at communicating the model's decision process to users (**usefulness**: Hoffman et al., 2019; Doshi-Velez and Kim, 2017).

The main reason that warrants a dedicated evaluation for the task at hand relates to the common generation strategy of these methods: SHAP, LIME, and Anchor are **perturbation-based** methods: They track how removing, replacing, or masking tokens within a document influences a detector's decision. Previous work was able to leverage coherence metrics for classification (Fröhling and Zubiaga, 2021; Liu et al., 2023c; Crothers et al., 2022; Liu et al., 2023a). Language-model-based detectors that exploit features of this type might be sensitive to the removal or masking of tokens, which could impact the quality of explanations.

This thesis aims to systematically evaluate how suitable these explanation methods are for explaining the decisions of detectors of machine-generated text through automated metrics and a user study. The aspects introduced above are tested with two existing fine-tuned transformer-based detectors (Guo et al., 2023; Solaiman et al., 2019) and a zero-shot method (Mitchell et al., 2023).

To inform experimental design, Chapter 2 compiles current research efforts in three

areas: The detection of machine-generated text (Section 2.1), explanation methods and their application to the problem (2.2-2.3) and, finally, approaches to evaluate the quality of explanations (2.4). Suitable experiments are subsequently adapted for detectors of machine-generated text in Chapter 3. To enable a comparison for the property of faithfulness between all methods (3.1), a *token removal experiment* (Arras et al., 2016) and the *pointing game* in Poerner et al. (2018) are extended to rule-based explanations. Stability is assessed with *controlled synthetic data checks* (Nauta et al., 2023) constructed specifically for the task (3.2). Previous user studies have evaluated usefulness (3.3) with problems humans have an intuitive understanding of and perform well in themselves, such as sentiment analysis or income prediction (Ribeiro et al., 2016, 2018; Hase and Bansal, 2020). Identifying patterns in a detector's behavior is arguably more challenging for users when they have limited knowledge about the features that might be leveraged. To keep *forward simulation* experiments feasible for humans when using complex and unexplored classifiers, an existing user study design from Hase and Bansal (2020) is extended with a special document selection strategy. Rather than randomly selecting documents, users are presented with pairs of documents chosen based on explanation similarity.

Figure 1.1 gives an overview of the results provided in Chapter 5. SHAP performs best across all automated metrics. LIME explanations are perceived as most useful by the users but decrease users' performance at predicting the decisions of the detectors. Generating Anchor explanations is not computationally feasible with the original implementation here. With a setup, where the search for a good explanation is terminated early, Anchor is still able to outperform LIME in certain experiments[1].

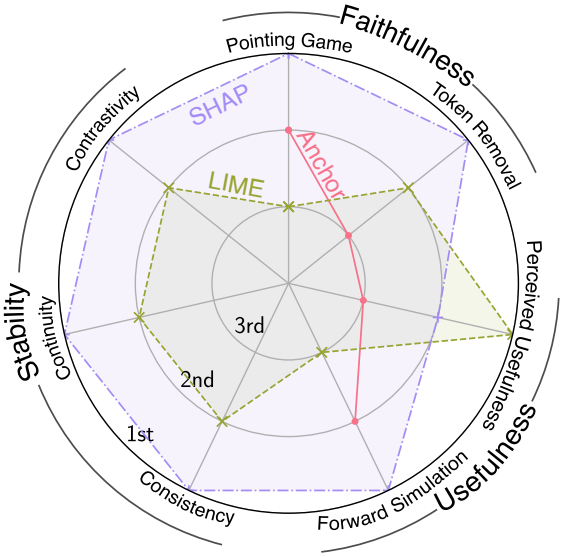


Figure 1.1.: Ranking in the experiments

[1] A repository with code for the experiments, a copy of the explanations, and the interface of the user study is made available at `https://github.com/loris3/evaluation_explanation_quality`

2. Literature Review

Designing experiments for evaluating explanation quality requires familiarity with the task and models. For this purpose, this chapter compiles past research efforts in three areas: The detection of machine-generated text (Section 2.1), explanation methods and their application to the task (2.2- 2.3), and approaches to evaluating the quality of explanations (2.4).

A survey of past attempts to explain the decisions of these detectors serves to identify common research objectives and types of analyses, based on which a set of properties to evaluate explanation quality with is selected. Lastly, it is discussed how experiments from related tasks can be adapted to the task.

2.1. Detection Methods

Thought experiments about the detection of machine-generated content predate the emergence of the term *computer science* (Turing, 1950). However, this thesis only considers detection efforts for text from current generative language models based on the Transformer architecture (Vaswani et al., 2017). The problem is framed as a binary classification task. This rules out interaction with the language model, mixtures of machine-generated and human-written text, and paraphrasing tasks. Recent detection approaches of this type are categorized into three groups:

2.1.1. Approaches Based on Handcrafted Features

Current generative language models are imperfect approximations of human language. For lack of a better definition of what constitutes *human text*, some evaluation- and development efforts focus on criteria like coherence, fluency, or repetitiveness (Pillutla et al., 2021; Gehrmann et al., 2023). Authors attempt to capture these and similar concepts with stylistic- and psycholinguistic features from related tasks (e.g., readability scores, word-frequency measures, sentence- or document-level statistics). Table 2.1 summarizes selected approaches that solely rely on handcrafted features.

The success of this type of detector depends strongly on the generator and text domain. Some authors select their features with the dataset in mind (e.g., news articles: Kumarage et al. 2023a, fraud detection in reviews: Kowalczyk et al. 2022) making a direct comparison of their results with detection attempts on more diverse domains unfeasible. The *gpt-2-output-dataset* (Radford et al., 2019), a dataset of websites and GPT-2 output used in various papers, enables a rough estimate of in-distribution performance throughout this chapter. On it, the simple logistic regression on TF-IDF uni- and bigrams

baseline in Solaiman et al. (2019) archives an accuracy of 0.74 on texts sampled from the untruncated distribution of the largest GPT-2 model (1.5B parameters)[1]. More sophisticated classifiers with handcrafted features, that were evaluated on this dataset, only accomplish marginal gains (+0.082: Fröhling and Zubiaga, 2021) or see no improvement at all (Crothers et al., 2022), whereas detectors based on fine-tuned language models (upcoming section) archive an accuracy of 0.95 on this data in the in-distribution setting (Solaiman et al., 2019).

Additionally, detection performance does not necessarily transfer well to new generators. Fröhling and Zubiaga (2021) observe that detectors trained on text from larger generators can detect that of smaller ones reasonably well if they share the same architecture. The opposite does not appear to be the case (Fröhling and Zubiaga, 2021; Crothers et al., 2022). Detection also breaks down when testing with text that is obtained with a different sampling strategy than the training data was (Fröhling and Zubiaga, 2021; Goldstein et al., 2023; Holtzman et al., 2020; Solaiman et al., 2019).

Handcrafted features have also been used as part of ensemble classifiers to improve the performance of other methods. These are omitted for brevity[2].

Table 2.1.: Approaches with handcrafted features. Only the results of the best-performing feature-based method are reported (highlighted in bold). Superscripts denote an overlap in datasets

		Generators						
Paper	**Approach**	**GPT-2**	**GPT-3**	**ChatGPT**	**GROVER**	**Others**	**Domain**	**Accuracy on**[1]
Solaiman et al. (2019)	Logistic regression on TF-IDF uni- and bi-grams	X					Web†	GPT-2 1.5B: 0.93 (Top-K=40) 0.74 (full)
Fröhling and Zubiaga (2021)	{Logistic regression, SVM, **NN**, Random forest} on stylistic features	X	X		X		Web$^{\dagger *}$ News^{G}	GPT-2 1.5B: 0.924 (Top-K=40) 0.782 (full) GPT3: 0.786 GROVER: 0.724
Crothers et al. (2022)	SVM on stylistic features	X	X				Web$^{\dagger *}$	GPT-2 1.5B: 0.712 GPT-3: 0.585
Kowalczyk et al. (2022)	{Random forest, **Extreme gradient boosting**} on stylistic features and LDA Topics	X					Reviews	GPT-2 124M: 0.887

Continued on next page

[1]GPT-2 text obtained with Top-K=40 sampling can be detected with an accuracy of over 0.9 (Fröhling and Zubiaga, 2021; Solaiman et al., 2019).

[2]See Kumarage et al. (2023a) for an approach that also applies an explanation method.

		Generators						
Paper	**Approach**	**GPT-2**	**GPT-3**	**ChatGPT**	**GROVER**	**Others**	**Domain**	**Accuracy on**[1]
Kumarage et al. (2023b)	Extreme gradient boosting on {**stylistic features**, BOW, word2vec}	X				X	Tweets	Multi-generator dataset: 0.771
Kumarage et al. (2023a)	Linear regression on {BoW, word2vec, ***journalism features***}	X	X	X	X	X	News	GROVER: 0.897 GPT-2[2]: 0.931 GPT-3: 0.912 ChatGPT: 0.883

[1] As reported in paper
[2] Model size not specified

2.1.2. Fine-Tuning Language Models

As with related NLP problems, pre-trained transformer-based language models have proven to be effective tools for detecting machine-generated text. Models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) are pre-trained on unlabeled text. They can be adapted to new classification tasks at low cost by adding an additional output layer (Devlin et al., 2019). Ippolito et al. (2020) attribute the superior performance of BERT-based detectors over many simple statistical models to their ability to capture dependencies on both the word- and sentence-level.

The RoBERTa-based classifier in Solaiman et al. (2019) archives an accuracy of 0.95 on the dataset with GPT-2 output introduced in the previous section (in-distribution setting). Several authors also report outstanding in-distribution performance for ChatGPT-generated text in specific domains: Liu et al. (2023b) archive an accuracy of over 0.99 on argumentative essays, Yu et al. (2023) a perfect AUROC score on academic abstracts, Mosca et al. (2023) perfect accuracy on academic papers and Guo et al. (2023) an F1 of over 0.99 on a multi-domain dataset. Moving out-of-domain, however, appears to have similar effects as it had for detectors based on handcrafted features. Yu et al. (2023)'s RoBERTa-based classifier archives a perfect AUROC on their dataset of human-written and ChatGPT-generated academic abstracts. However, the RoBERTa based one of Guo et al. (2023) trained on a Q&A dataset with ChatGPT-generated text (Section 4.1) only accomplishes an accuracy of 0.7554 on this data (AUROC 0.8191[3]). Crothers et al. (2022) perform a cross-generator experiment where they train their RoBERTa-based detector on a dataset with text from GPT-2 355M but evaluate on text from GPT-2 1.5B and GPT-3. They report an accuracy of 0.715 for GPT-2 1.5B text and 0.566 for GPT-3. One noteworthy study of Tu et al. (2023) illustrates a ChatGPT-specific problem and underlines the need for further research in the cross-generator setting. They collect ChatGPT responses to the same prompts from two time periods: before 18/01/23 and after

[3] Accuracy not provided for the in-distribution experiment.

05/03/23. The performance of a RoBERTa-based model trained on output from the first period drops from 0.992 to 0.962 in this short timespan.

2.1.3. Zero-Shot Approaches

The approaches in this category do not require training. They do, however, require access to either the original generator or an appropriate surrogate language model.

DetectGPT

Solaiman et al. (2019) propose the arguably simplest zero-shot approach: They classify texts by thresholding on the total probability of a document under the language model (GPT-2). Mitchell et al. (2023) iterate on this approach and propose the perturbation based method DetectGPT: In experiments, they compare the log probability of the original document under the language model with the log probabilities of small rewrites of the document. They observe that the average drop in log probability is higher for machine-generated documents than human-written ones[4]. Their approach is as follows: An external model q[5] generates perturbations of the document x. The log probability of the perturbations and x are subsequently obtained from the original language model θ. Mitchell et al. (2023) define the perturbation discrepancy d as:

$$d(x, p_\theta, q) = \log p_\theta(x) - \mathbb{E}_{z\sim q} \log p_\theta(z) \tag{2.1}$$

A decision is obtained by thresholding on d. Comparisons with other approaches are sparse and hindered by the use of different evaluation metrics. On the ChatGPT detection task in Pegoraro et al. (2023), DetectGPT (balanced accuracy 0.49) is outperformed by the RoBERTa classifier of Guo et al. (2023) (0.73) operating out-of-distribution, a feature-based classifier by Fröhling and Zubiaga (2021) (0.585), and the out-of-domain RoBERTa classifier of Solaiman et al. (2019) (0.51).

The original formulation of DetectGPT assumes access to the exact source language model that generated the documents. Mireshghallah et al. (2023) successfully employ smaller surrogate models with this architecture. This enables the evaluation with ChatGPT-generated text in this thesis (Section 4.2), on which it archives an accuracy of 0.74.

DNA-GPT

Yang et al. (2023) find that conditioning generators with machine-generated text can lead to highly similar output across runs. They find that this phenomenon is less pronounced if the generator is conditioned on human-written text and therefore propose

[4] Mitchell et al. (2023) offer an alternative interpretation of their approach: When plotting the log probabilities of the perturbations and original document on a curve, the document is likely to be machine-generated if it lies in an area with clear negative curvature.

[5] The authors employ T5 (Raffel et al., 2023). Perturbations are syntactically valid natural text here (Mitchell et al., 2023).

to classify documents via a n-gram-based similarity metric. Specifically, this similarity metric reflects the average n-gram overlap of the machine-generated samples with the input document. In contrast to DetectGPT, this approach does not require access to model internal probabilities.

The only comparison to other approaches is the one provided in the paper, where this detector consistently outperforms DetectGPT across multiple datasets with text from *text-davinci-003*[6]. Note that Yang et al. (2023) also propose a metric for the white-box case, where access to the internals of the generator is granted. This metric is based on the log probability of tokens under the model and is equivalent to that used in DetectGPT (Equation 2.1). However, they employ the original language model as the perturbation generator q, and allow more substantial rewrites.

2.2. Explanation Methods

Guidotti et al. (2018) describe three approaches to explaining models: *Outcome explanations* describe the model's reasoning for a single instance (document) in isolation. *Model explanations* approximate the model's rationale globally so that its behavior is understood for any input. Explanations generated for purposes of *model inspection* explain single properties of a model globally (e.g., the influence of one specific feature on the decision process). This thesis evaluates explanation methods for outcome explanations, as all approaches identified in a survey of previous work (Section 2.3) rely on explanations of this type. The same applies to the output format: only *feature importance type* explanation methods have been applied to the task before. Anchor (Ribeiro et al., 2018), a method that produces *rule-based* outcome explanations, was found to be compatible with the task and is evaluated here as well.

2.2.1. Additive Feature Attribution Methods

Methods of this type attempt to explain a prediction $f(x)$ the original model f made for an instance x with a simpler model g (Lundberg and Lee, 2017). Instead of approximating the complete decision function globally, g is only expected to be faithful to f for x and similar documents (Ribeiro et al., 2016; Lundberg and Lee, 2017). While any type of model could be used for this purpose, both approaches described in this subsection can be categorized as *additive feature attribution methods* (Lundberg and Lee, 2017). These can be conceptualized as linear models

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i \cdot z'_i \tag{2.2}$$

that do not operate on the original input x but on a binary encoding $z' \in \{0,1\}^M$, where M is the number of words in the input document (Lundberg and Lee, 2017; Ribeiro et al.,

[6] A model from the GPT-3.5 family from OpenAI (Brown et al., 2020) that exposes the probabilities required for DetectGPT.

2016). For text, z' encodes the inclusion or omission of words in perturbed samples z (versions of the input document where certain tokens are removed, replaced or masked). The surrogate model g is trained on a dataset of decisions $f(z)$ for perturbed samples z (Ribeiro et al., 2016). By analyzing the parameters of g, one can subsequently map feature importance scores to individual tokens in x. They can be presented to the user by highlighting words in the input document (Figure 2.1). LIME and SHAP, outlined below, vary in their strategy for finding g (Lundberg and Lee, 2017).

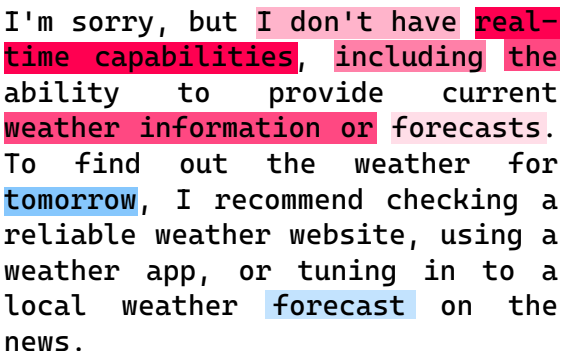


Figure 2.1.: Plot of a hypothetical feature importance explanation for the decision *machine* on text from ChatGPT. Red (all except *forecast* and *tomorrow*) represents positive feature importance scores

LIME

Ribeiro et al. (2016) name two properties of good explanations they aim to address with LIME: The generated explanation should be interpretable by humans (1), and locally faithful to f around x (2). Their approach to training g (as formulated in Equation 2.2) is as follows: Property 1 is accomplished by limiting the length of the explanation (or size of g) to K words. This is enforced through a complexity measure $\Omega(g)$ that introduces an infinite penalty if the explanation exceeds K tokens in length and is zero otherwise. Property 2 is captured by a fidelity function

$$\mathcal{L} = \sum_{z,z' \in \mathbb{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{2.3}$$

where samples z closer to x in the local neighborhood $\mathcal{D}$ around x are weighted stronger by a weighting function π_x[7]. The objective function for training g is

$$\arg\min_g \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.4}$$

[7]For text, Ribeiro et al. (2016) use a cosine distance based function $\pi_x(z) = exp(\frac{-(1-\frac{\mathbf{z}\cdot\mathbf{x}}{\|\mathbf{z}\|\|\mathbf{x}\|})^2}{\sigma^2})$

SHAP

Lundberg and Lee (2017) point out that an approach from cooperative game theory can be utilized to obtain suitable explanation models. They set up g so that it provides Shapley values (Shapley, 1953), which fulfill a set of properties that are desirable for explanations: The first, *local accuracy*, requires that the output of g matches that of f for the original input x. The second property, *missingness*, states that a feature marked absent in z' cannot be attributed importance ($z'_i = 0 \Rightarrow \phi_i = 0$). The last property, *consistency* states that the score of a feature must not decrease if the model is perturbed so that this single feature becomes more important for the decision (Lundberg and Lee, 2017).

While the exact calculation of Shapley values is intractable for most models, Lundberg and Lee propose both model-agnostic, and higher performing model specific ways of approximating them. One of these, Kernel SHAP, is an adaptation of LIME that is adapted so that g satisfies the properties outlined above. Lundberg and Lee remove the complexity measure Ω form Equation 2.4 and replace the weighting kernel $\pi_x(z)$ with a sparsity measure:

$$\pi_x(z') = \frac{M - 1}{\binom{M}{c(z')} \cdot c(z') \cdot (M - c(z'))}, \tag{2.5}$$

where $c(z')$ counts the number of non-zero entries in z' and M is the number of features in x (Lundberg and Lee, 2017). The implementation of SHAP used in this thesis calculates Owen values (Owen, 1977) as an approximation of Shapley values. Rather than to individual input features, feature importance is assigned to groups of correlated features to make computation more tractable (Lundberg, 2019).

2.2.2. Rule-Based Methods

Rule-based explanation methods attempt to explain a decision $f(x)$ by a set of rules. These rules only have to apply in the local neighborhood $\mathcal{D}$ around x. This restriction has the same advantage as it had for additive feature attribution methods: It makes expressing complex decision processes with simple rules feasible (Ribeiro et al., 2018).

Anchor

Anchor explanations provide *if-then rules* (Figure 2.2). Ribeiro et al. (2018) define an Anchor A_m for the document x as a set of tokens $A_m = \{t_{m,i}\}$ that, if present in the document, guarantee the same decision as for x with a probability greater than τ for perturbations in the local neighborhood of x. For a simplified outline of their approach when applied to text, let the only type of rules be the presence of tokens in a document, denoted by $t \in x$. Let A' be a rule that applies to the document to explain, e.g., $A' = \{(\text{This} \in x) \wedge (\text{is} \in x) \wedge (\text{example} \in x)\}$. By Ribeiro et al.'s definition, A' is an anchor A if it also applies to sufficiently many perturbations in the local neighborhood $\mathcal{D}$ around x that contain the tokens from A' and share the same prediction ($\{z \sim \mathcal{D}(z|A') | f(z) = f(x)\}$). Formally, the rule has to apply to these perturbations with a precision of τ to

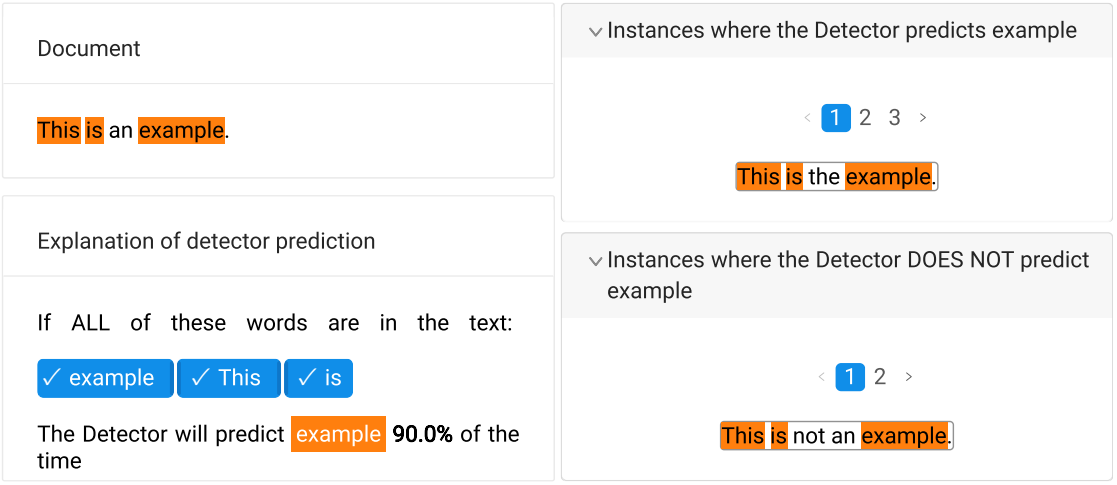


Figure 2.2.: Plot of a hypothetical Anchor explanation for a detector that detects examples

be considered an Anchor:

$$\underbrace{\mathbb{E}_{\mathcal{D}(z|A')}[\mathbf{1}_{f(z)=f(x)}]}_{:=prec(A')} \geq \tau \tag{2.6}$$

where

$$\mathbf{1}_{f(z)=f(x)} = \begin{cases} 1, \text{if} f(z) = f(x) \\ 0, \text{otherwise} \end{cases} \tag{2.7}$$

As there can exist multiple Anchors for a document that match this definition, Anchor aims to select those that maximize a coverage metric, defined as the probability that an Anchor applies to any given sample in $\mathcal{D}$, including those with $f(z) \neq f(x)$:

$$\max_A \underbrace{\mathbb{E}_{\mathcal{D}(z)}[A(z)]}_{:=cov(A)} \tag{2.8}$$

Longer Anchors will typically have high precision but low coverage (Ribeiro et al., 2018). These are considered less effective explanations, as they typically apply to fewer instances: Simply selecting all tokens from the input as an Anchor would in fact, as Ribeiro et al. point out, always fulfill the condition in Equation 2.6.

Anchors are constructed iteratively by beam search. In each iteration, a greedy algorithm attempts to extend each candidate in a set of candidate rules $\mathcal{A}$. A new token is added to a candidate $A' \in \mathcal{A}$ if the addition increases the precision metric in Equation 2.6. This requires evaluating f on $z \sim \mathcal{D}(z|A')$. Only the B best candidates by precision are kept and further iterated upon. The search will terminate and return the Anchor with the highest coverage, once a valid Anchor with $prec(A) \geq \tau$ is found.

2.3. Survey of Explanations for Detectors of Machine-Generated Text

This section compiles previous attempts to explain decisions of detectors of machine-generated text. Table 2.2 summarizes detectors, generators, and target domains for 8 papers. Only papers that utilize text from current generative language models based on the Transformer architecture are included here. All explain either fine-tuned language models or detectors based on handcrafted features. Except for one, all papers include SHAP as an explanation method. Two use LIME, and one employs a model-specific explanation method as well (not further discussed here).

2.3.1. Research Objectives

Across all papers, three motivations for analyzing explanations could be identified: Most use explanations to identify features and patterns their fine-tuned language models were able to leverage (Mitrović et al., 2023; Mosca et al., 2023; Silva and Frommholz, 2023; Liu et al., 2023c). One paper uses explanation methods to explore a novel dataset (Yu et al., 2023). Two assess the extent to which individual handcrafted features prove useful for detection (Kumarage et al., 2023a; Kowalczyk et al., 2022). And a final paper integrates SHAP into a user-facing system to explain an ensemble classifier (Weng et al., 2023). It is included here merely for completeness, as it does not present nor discuss any explanations.

2.3.2. Types of Analyses

Yu et al. (2023), Kowalczyk et al. (2022) and Kumarage et al. (2023a) perform a quantitative analysis of explanations: Yu et al. (2023) evaluate SHAP values on an aggregate level for different subsets of their dataset of abstracts for academic papers: It includes four classes of text with increasing level of ChatGPT involvement, ranging from fully human-written to fully machine-generated documents. For each subset, they extract the top-10 tokens by average SHAP value across all explanations. In an analysis of this ranking, they find that the fine-tuned PubMedBERT (Gu et al., 2022) classifier uses substantially different features between the different subsets.

Kumarage et al. (2023a) and Kowalczyk et al. (2022) apply SHAP to classifiers that are based on handcrafted features. Although they still generate explanations post-hoc, their approaches can be considered attempts of model inspection (Section 2.2): Instead of using words or sentences as input features, they configure SHAP to explain the decision in terms of the handcrafted features in their model. This enables the calculation of a mean SHAP value for each feature across all documents, which is then used to rank features by usefulness[8].

[8]Note that this approach is invalid if words are used as input features: Outcome explanations generated with SHAP are only required to be locally faithful (Section 2.2.1). Words can have different meanings in different contexts.

The remaining papers in Table 2.2 take a qualitative approach. Researchers generate outcome explanations and attempt to find patterns by inspecting feature importance visualizations of documents like the one in Figure 2.1. To convey the type of insights these analyses appear to be able to provide, three broad categories of findings are summarized here: Detectors were able to rely on **sampling artifacts** found in machine-generated text (Yu et al., 2023; Liu et al., 2023c; Mitrović et al., 2023; Mosca et al., 2023). Examples of this include the use of rare words in unfitting contexts (Mitrović et al., 2023; Mosca et al., 2023), a general lack of lexical diversity, or the presence of profanities (Mitrović et al., 2023).

The detectors further appear to capture **deviation from domain-specific norms**: The model of Liu et al. (2023c) exploits ChatGPT's tendency to insert summaries of results in inappropriate places like the abstract of academic papers[9]. Mitrović et al. (2023) observe that ChatGPT tends to provide general descriptions of restaurants in reviews, while humans tend to describe experiences.

Authors also observe cases where the detector appears to leverage **differences in syntax** between sources. Examples are the frequent use of certain stock phrases (Mosca et al., 2023; Mitrović et al., 2023; Liu et al., 2023c), a preference for certain tenses (Mitrović et al., 2023), or the aforementioned tendency of ChatGPT to frequently summarize information (Liu et al., 2023c).

Table 2.2.: Papers that explain decisions of detectors of machine-generated text

	Method						
Paper	**SHAP**	**LIME**	**Other**	**Domain**	**Detector**	**Analysis**	**Goals**
Mitrović et al. (2023)	X			Reviews	DistilBERT (Sanh et al., 2020) fine-tuned on ChatGPT	M	Identifying patterns useful for classification
Yu et al. (2023)	X			Academic abstracts	PubMedBERT (Gu et al., 2022) fine-tuned on ChatGPT	M A	Analyzing class differences in a mixed-task dataset (full generation vs. rewrites) Comparison of top-features between classes
Liu et al. (2023c)	X		IG	Wiki Essays News	LSTM and DNN on RoBERTa embeddings trained on ChatGPT	M	Identifying patterns useful for classification

Continued on next page

[9]They attribute this to the training objective of ChatGPT being QA (Liu et al., 2023c)

Table 2.2 *Continued from previous page*

	Method						
Paper	**SHAP**	**LIME**	**Other**	**Domain**	**Detector**	**Analysis**	**Goals**
Mosca et al. (2023)	X	X		Academic papers	{RoBERTa, Galactica (Taylor et al., 2022)} fine-tuned on a mixed-generator dataset of {ChatGPT, GPT-3, GPT-2, Galactica (Taylor et al., 2022), SCIgen (Stribling et al., 2005)}	M	Identifying patterns useful for classification
Silva and Frommholz (2023)		X		Academic papers	Ensemble of BERT based *multimodal transformer* (Gu and Budhkar, 2021) and stylistic features on ChatGPT	M	Identifying patterns useful for classification
Kumarage et al. (2023a)	X			News	Logistic Regressors on stylistic features trained on datasets of {ChatGPT, GROVER (Zellers et al., 2019), CTRL (Keskar et al., 2019), PPLM_gpt2 (Dathathri et al., 2020), GPT-2, GPT-3}	A	White-box model inspection: explanation in terms of model-internal features
Kowalczyk et al. (2022)	X			Reviews	Extreme Gradient Boosting on stylistic features and LDA-topics trained on GPT-2	A	White-box model inspection: explanation in terms of model-internal features
Weng et al. (2023)	X			Academic papers	Human-in-the-loop ensemble classifier for {ChatGPT, GPT-3, GPT-2}	-	White-box model inspection: Proposes the integration of SHAP into a user-facing system

[M] Manual qualitative analysis of outcome explanations
[A] Aggregate quantitative analysis of outcome explanations
[IG] Integrated gradients (model-specific method)

2.4. Evaluating Explanation Quality

This thesis focuses on three aspects that are particularly relevant for the type of manual analysis of explanations described in Section 2.3: Explanations should accurately depict the detector's behavior (**faithfulness:** Jacovi and Goldberg, 2020; Ribeiro et al., 2016; Alvarez Melis and Jaakkola, 2018). They should be sensitive enough and sufficiently deterministic (**stability**: Alvarez Melis and Jaakkola, 2018; Lakkaraju et al., 2020; Nauta et al., 2023), and be effective at communicating the model's decision process to users (**usefulness**: Hoffman et al., 2019; Doshi-Velez and Kim, 2017). This section aims to compile a set of experiments from the literature and to discuss their compatibility with the chosen detectors (two fine-tuned language models and DetectGPT; Section 4.2).

2.4.1. Evaluating Faithfulness

Previous work (Ribeiro et al., 2016; Lakkaraju et al., 2020) has evaluated faithfulness with simple interpretable models (e.g., decision trees or linear regression). With these, one can easily identify which input features (words) were relevant to the decision. The degree to which the explanation aligns with this known set of relevant features is taken as a measure of faithfulness (Ribeiro et al., 2016). However, one cannot expect observations from such models to transfer to language-model-based detectors with a vastly more complex architecture. Similarly, evaluation methods that perturb the internals of the model (Adebayo et al., 2018; La Rosa et al., 2020) are not applicable here, as this thesis includes a zero-shot method.

Two evaluation methods are compatible with all detectors: Arras et al. (2016) evaluate faithfulness with **token removal experiments**, where they track the effect that removing tokens from the input document has on the detector's decision. This can reveal whether the features selected by the explanation method were truly relevant to the detector's decision. Note that this setup raises the same concern perturbation-based explanation methods do when applied to this kind of classifier: It assumes detectors behave predictably for partial input. **Controlled synthetic data checks** are an alternative type of experiment that foregoes this issue (Nauta et al., 2023). With their *pointing game*, Poerner et al. (2018) propose a design of this type for evaluating feature importance type explanations. The token removal experiment in Arras et al. (2016) and this pointing game are further described and extended to Anchor explanations in Section 3.1.

2.4.2. Evaluating Stability

For the choice of explanation methods and detectors in this thesis, assessing stability in perturbation-based experiments (Alvarez Melis and Jaakkola, 2018; Lakkaraju et al., 2020) is most feasible. Other setups require retraining the model (Adebayo et al., 2018; Tseng et al., 2020; Nauta et al., 2023), which is not practical with these detectors and not applicable to the zero-shot method. Quantifying the stability of explanations across runs (*consistency*: Nauta et al., 2023) with agreement measures is only useful to establish a lower-bound here: For these language-model-based classifiers, one is forced to trade

convergence for computation time. The properties of *continuity* (similar explanations for similar documents with the same prediction) (Alvarez Melis and Jaakkola, 2018) and *contrastivity* (sufficiently different explanations for similar documents but different predictions) described by Nauta et al. (2023) are alternative notions of stability that rely on convergence to a lesser extent. A setup that tests for these with perturbations from a language model is proposed in Section 3.2.

2.4.3. Evaluating Usefulness

Users' performance in *forward simulation experiments* (Doshi-Velez and Kim, 2017) can serve as a measure of usefulness. In these, participants attempt to predict the behavior of the classifier with the help of explanations. Experimental setups that provide explanations for all examples shown to users (Chen et al., 2020; Nguyen, 2018; Bang et al., 2021) allow for a comparison of different methods, but cannot verify whether showing explanations led to an improvement in user performance. Multiphase designs (Hase and Bansal, 2020; Ribeiro et al., 2018) can assess this, as they also measure performance before explanations are introduced. The user study in Hase and Bansal (2020) utilizes such a design, but evaluates with two comparatively simple binary classifiers and tasks (sentiment analysis and income prediction). Given that the explanation methods to be analyzed here produce outcome- and not model explanations (Guidotti et al., 2018), it is unlikely that users will obtain a comprehensive understanding of the detector's behavior or relevant features and successfully apply that to new instances with a real-world dataset of human and machine-generated text. A strategy to increase the feasibility of this kind of experiment is proposed in Section 3.3.

The users' ability to predict model behavior may present an incomplete assessment of the adequacy of an explanation method. Other researchers include *rating tasks* in their evaluations (Hoffman et al., 2019; Pruthi et al., 2020; Liu et al., 2019a; Atanasova et al., 2020; Ribeiro et al., 2016; Hase and Bansal, 2020). Through questionnaires, they attempt to measure concepts like *usefulness* (Hase and Bansal, 2020; Pruthi et al., 2020), *satisfaction* (Hoffman et al., 2023), *helpfulness* (Zhou et al., 2020), *trust* (Ribeiro et al., 2016) or other subjective task-specific dimensions of explanation quality. These can provide further insights into the suitability of explanation methods and inform future applications. In this thesis, *perceived usefulness* is measured with questionnaire items adapted from Hoffman et al. (2019).

2.5. Summary

This chapter discussed detection- and explanation methods, as well as evaluation methods for assessing explanation quality. Current high-performing detectors are black boxes that benefit from the addition of explanations. Only outcome-explanation methods were applied to the task before, with the majority of papers using SHAP or LIME for this purpose. To gain insights into the behavior of their classifiers, researchers commonly examine individual explanations through manual qualitative analysis. With this application

in mind, explanation quality is evaluated along the dimensions of faithfulness, stability, and usefulness in this thesis. Existing evaluation methods for these properties require adaptation to the task, which will be addressed in the next chapter.

3. Evaluation of Explanation Quality

This chapter explains how the evaluation methods introduced in Chapter 2 will be used to evaluate explanation quality for detectors of machine-generated text. It discusses how the experiments are adapted to make them compatible with all explanation methods and feasible to perform with the detectors selected for this thesis (two fine-tuned language models and DetectGPT). Table 3.1 provides an overview of the individual experiments by property.

Throughout this chapter, $f(d_i)$ refers to the decision of a detector f for a document d_i. D is the base dataset of human-written and machine-generated documents $\{d_i\}$. SHAP and LIME provide *feature importance scores* for tokens from the input document d_i as explanations for decisions made by f (Lundberg and Lee, 2017). Those tokens that had the strongest impact on the detector's decision should be attributed the highest scores. An Anchor $A_{i,m}$ for the document d_i is defined as set of tokens $A_{i,m} = \{t_{i,j}\}$ that, if present in the document, guarantees the same decision as for d_i with a probability greater than τ for perturbations in the local neighborhood of d_i.

	Property	**Experiment**	**Evaluation**
Automated Metrics	Faithfulness	Pointing Game (Poerner et al., 2018)	Pointing Game Accuracy
		Token Removal (Arras et al., 2016)	Rate of decline/incline in accuracy when masking tokens
	Stability	Consistency (Nauta et al., 2023)	Agreement with Krippendorff's α across 5 explanations generated with different seeds
		Continuity (Alvarez Melis and Jaakkola, 2018)	Krippendorff's α between the explanation of the original document and explanations of 4 perturbations
		Contrastivity (Nauta et al., 2023)	Krippendorff's α on the explanations of the common part; Accuracy in two controlled synthetic data checks
User Study	Usefulness	Forward Simulation (Hase and Bansal, 2020)	Change in user accuracy after being shown explanations
		Perceived Usefulness (Hoffman et al., 2019)	User ratings measured with three Likert scale items

Table 3.1.: Overview of the experiments

3.1. Evaluation of Faithfulness

A faithful explanation method should accurately depict the detector's behavior (Jacovi and Goldberg, 2020; Ribeiro et al., 2016; Alvarez Melis and Jaakkola, 2018). The first test for this is the **pointing game** in Poerner et al. (2018). Random sentences from the base dataset D are concatenated to form a synthetic dataset of *hybrid documents* $D^h = \{d_i^h\}$. The length of these hybrid documents is set to match the mean document length by the number of sentences in the original dataset for this thesis. It is assumed that the detector's decision $f(d_i^h)$ on such a document can be traced to segments that were originally part of documents with ground truth y_i equal to $f(d_i^h)$. If the detector behaves this way, a faithful explanation method should find these segments to be more important for the decision than those originating from opposite-class documents. In the pointing game, feature importance type explanation methods are therefore awarded hits for a document d_i^h if the token with the highest feature importance score $t_{i,max} \in d_i^h$ was in fact originally part of a document with ground truth y_i equal to $f(d_i^h)$ (Poerner et al., 2018):

$$S_x = \bigcup_{\forall d_i \in D} \{t \in d_i \mid y_i = x\}$$

$$hit(d_i^h, t) = \begin{cases} 1, & \text{if } t \in S_{f(d_i^h)} \\ 0, & \text{otherwise} \end{cases}$$

The *pointing game accuracy* is the fraction of documents in the dataset D^h that get awarded hits. For feature importance type explanations, it is given as:

$$\text{Acc}_{\text{pg}} = [\sum_{\forall d_i^h \in D^h} hit(d_i^h, t_{i,max})]/|D^h|$$

Poerner et al. (2018) only evaluate feature importance type explanation methods. Note that there is no distinction by importance between tokens within an Anchor. Furthermore, a single Anchor can span multiple sentences. Hits are therefore attributed proportionally for this explanation method in this thesis: For documents with multiple valid Anchors, only the one that applies to the highest fraction of perturbations (τ) is considered in this experiment. The hit function is replaced with one that tests all tokens $t_{i,j}$ specified by the Anchor A_i individually and returns the average number of hits instead of a binary value:

$$hit^R(d_i^h) = \frac{\sum_{\forall t_{i,j} \in A_i} hit(d_i^h, t_{i,j})}{|A_i|}$$

A **token removal experiment** as in Arras et al. (2016) is performed as the second test for faithfulness. Let d_i be the original document and d_i^k a version with the k top-tokens by feature importance towards $f(d_i)$ removed. Arras et al. plot the accuracy of the detector at different k with respect to the ground truth y_i of the original documents.

To enable a comparison with Anchor explanations, tokens from the Anchor that applies

to the highest proportion of documents in the local neighborhood (the one with the highest τ) are removed first and in random order in this thesis.

3.2. Evaluation of Stability

Three different notions of stability are evaluated here with purpose-built experiments. The individual experiments aim to measure the properties of consistency, continuity, and contrastivity described in Nauta et al. (2023).

As will be discussed in Section 4.3, the search for good rules is terminated early to make the computation of Anchors feasible with these detectors. Anchors are constructed with a greedy algorithm. As set up here, the search space is not sufficiently explored to expect consistent explanations from this process. Anchor is excluded from these experiments as it would add an excessive amount of computation time[1].

Consistency The consistency across five explanations for the same document is measured with the agreement metric Krippendorff's α (Krippendorff, 1970), calculated on the explanations' feature importance vectors.

Continuity The continuity between the explanation for the original document and explanations for a set of 5 perturbations $\{d_i^t\}$ is also measured with Krippendorff's α. For each document d_i from the original dataset, a single token is randomly selected and replaced with an arbitrary number of tokens using T5 (Raffel et al., 2020). The maximum output length was set to 150 tokens. It is verified that $f(d_i^t) = f(d_i)$ (Nauta et al., 2023). In some instances, T5 fails to generate five unique replacements. In these cases, the token is replaced with a random token from the vocabulary.

Contrastivity This experiment verifies that documents that are similar in content, but are assigned different labels by the detector, get sufficiently different explanations (Nauta et al., 2023). To obtain pairs of documents (d_i, d_i^Ω) where d_i and d_i^Ω are similar but get assigned different labels, documents from the dataset are edited with a language model. In each iteration k, one token is deleted from the end of the document. The remainder d_i^* is used as the prompt for facebook/opt-350m (Zhang et al., 2022), set up to generate a perturbation d_i^Ω. Additional tokens are removed until $f(d_i^\Omega) \neq f(d_i)$.

To encourage smaller edits, five attempts are made at every k. This strategy can produce very dissimilar documents if the first tokens in the document are highly influential for the decision of the detector[2]. To retain a certain level of similarity between documents, perturbations that edit more than 50% of the original document are discarded. The resulting synthetic datasets of pairs (one set per detector) are further described in Figure B.2 in Appendix B.

[1] A minimum of 900 hours based on the average generation time with the original documents on an NVIDIA RTX 4060 GPU.

[2] This can be observed for documents that start with greetings like "Hi!".

Three scores are subsequently calculated on the explanations for d_i and d_i^Ω. These verify whether the explanations are consistent with the generation strategy: Given that the left part of d_i and d_i^Ω are identical, but $f(d_i^\Omega) \neq f(d_i)$, one expects the new part in d_i^Ω to have had a strong influence on the detector when labeling d_i^Ω. A sufficiently sensitive explanation method should be able to detect this.

The first score (c_α) is Krippendorff's α for the feature importance scores on the matching part of d_i and d_i^Ω. This enables a comparison with the results from the consistency and continuity experiments.

A second and third score are formulated as synthetic data checks, similar to the pointing game introduced in Section 3.1. The second score ($c_{\boldsymbol{inter}}$) is calculated on the parts that differ across the two documents, d_i^- and $d_i^{\Omega-}$. It tests whether the mean feature importance score towards $f(d_i^\Omega)$ (denoted $\mu_{\vec{v}\ \Omega}(\cdot)$) is higher for $d_i^{\Omega-}$ than it is for d_i^-. A hit function is given as:

$$hit_{inter} = \begin{cases} 1, & \text{if } \mu_{\vec{v}\ \Omega}(d_i^{\Omega-}) > \mu_{\vec{v}\ \Omega}(d_i^-) \\ 0, & \text{otherwise} \end{cases}$$

The third score ($c_{\boldsymbol{intra}}$) shares the same intuition but is defined only on d_i^Ω. As the left parts of d_i and d_i^Ω are identical, but $f(d^\Omega) \neq f(d_i)$, one expects the generated part $d_i^{\Omega-}$ to have a higher average feature importance score towards $f(d_i^\Omega)$ than the shared part:

$$hit_{intra} = \begin{cases} 1, & \text{if } \mu_{\vec{v}\ \Omega}(d_i^{\Omega-}) > \mu_{\vec{v}\ \Omega}(d_i^{\Omega*}) \\ 0, & \text{otherwise} \end{cases}$$

Both scores are again given as the fraction of documents in the dataset that score a hit:

$$c_{\{intra,inter\}} = [\sum_{\forall(d_i,d_i^\Omega)} hit_{\{intra,inter\}}(d_i, d_i^\Omega)]/|D|$$

3.3. Evaluation of Usefulness

Usefulness is defined here as the explanation method's ability to improve users' understanding of the detector's behavior (Hoffman et al., 2019; Nauta et al., 2023). To quantify this, the design for a forward simulation experiment in Hase and Bansal (2020) is modified here (Figure 3.2). In phase 1 of the experiment, users inspect decisions of the detector on a set of documents $\{a_j\}$ without explanations. In phase 2 they are then instructed to predict the detector's decision (not to guess the true document class) on a second set $\{b_j\}$. Phase 3 provides explanations for set $\{a_j\}$ (Figure 3.5). Users conclude the experiment by labeling the documents from set $\{b_j\}$ again in phase 4. The change in user accuracy from phase 2 to 4 is reported as a measure of performance (Hase and Bansal, 2020). In this thesis, three additional questions from Hoffman et al. (2019) are asked for every explanation shown in phase 3 to measure *perceived usefulness*. These are provided in Figure 3.1.

For each document, please also rate to what extent you agree with these statements:

Q1 From the explanation, I **understand why** the detector decided the way it did for this document.

Select *agree* or *strongly agree* if you think the visualization presents sufficient evidence for why the detector decided the way it did in this specific case.

Select *disagree* or *strongly disagree* if you can't figure out why the detector decided the way it did.

Q2 From the explanation, I now better **understand how** the detector works.

Select *agree* or *strongly agree* if you think this explanation increased your understanding of how the detector reasons.

Select *disagree* or *strongly disagree* if you don't.

Q3 The information from this explanation **will help me** predict the detector's behaviour.

Select *agree* or *strongly agree* if you think you could apply this information to the documents you labelled in the previous phase. You will do so in the next phase.

Select *disagree* or *strongly disagree* if you don't.

Figure 3.1.: Instructions for the Likert scale items shown in phase 3 adapted from Hoffman et al. (2019)

Document Selection Users can only apply observations about detector behavior in the annotation phases (2 and 4) if they have seen similar cases in the teaching phases (1 and 3). Rather than using random documents, two sets of documents $\{a_j\}$ and $\{b_j\}$ are constructed here. Each document a_j, to be shown in phases 1 and 3, has a corresponding document b_j shown in phases 2 and 4. The explanations for the documents in a pair (a_j, b_j) should be *sufficiently similar* so that the task is feasible for human annotators. In the first step, all possible combinations of documents in the dataset are sorted by their cosine similarity in a bag-of-words encoding of the most salient features (as in Ribeiro et al., 2016). A second step aims to maintain sufficient diversity across pairs: Among the top-k most similar pairs, the n pairs that maximize coverage are chosen, defined here as the number of features with a non-zero importance score in the global encoding.

For rule-based explanations, this strategy cannot be applied as one lacks an appropriate similarity metric for this type of rule. Pairs are found by testing for set-equality: For a given document a_j, all other documents in the dataset that share an Anchor with a_j, and have $f(d_i) = f(a_j)$ are collected. Anchors can be as short as one token long, and documents therefore might only overlap in one word with this strategy. A second step

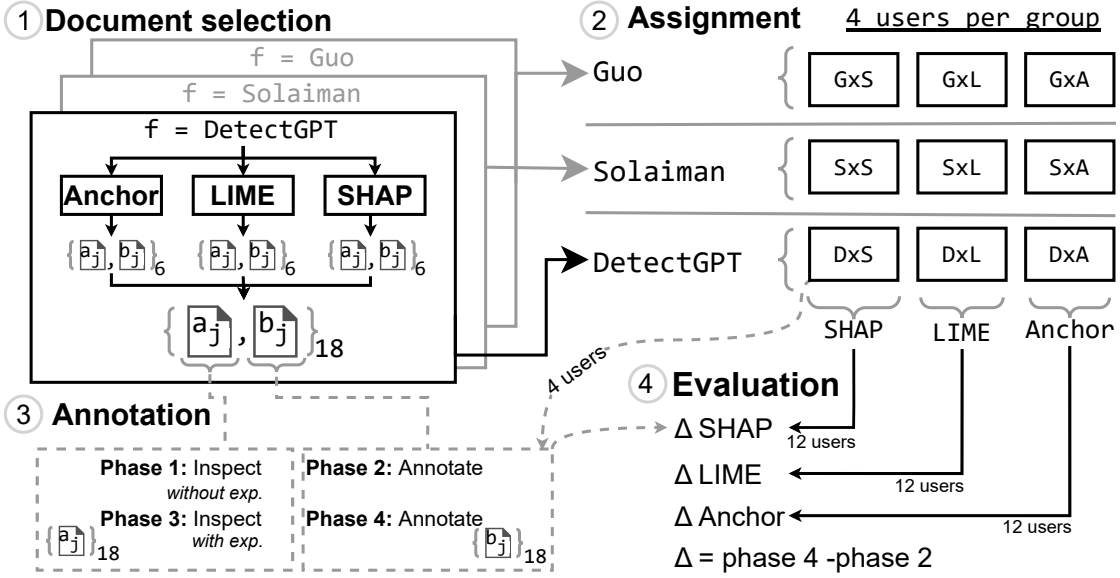


Figure 3.2.: Setup of the user study

is an attempt to address this: If multiple documents share an Anchor with a_j, only the document with the highest Jaccard index with a_j is kept. This is done so that the two documents share as similar of a context as possible. Finally, n pairs are selected at random from these candidates.

Note that these selection strategies will yield different sets for different pairings of detectors and explanation methods. The experiment will be conducted with three sets of pairs, one per detector (Figure 3.2). Of the 18 pairs in a set, 6 will be determined by each explanation method. Document- and explanation similarity for the three sets are reported in Section 5.3. The datasets are balanced in respect to the detectors' predictions, to eliminate counting examples in phases 1 or 3 to infer the number of machine- and human-documents in phases 2 or 4 as a strategy (Hase and Bansal, 2020).

Assignment In total, there are 9 different explanation method-detector combinations. One user only sees explanations from one explanation method for one detector. Users who are assigned the same detector, but different explanation methods, see the same set of documents. Those who are assigned the same explanation method, but different detectors do not.

Evaluation The average change in user accuracy is reported per explanation method. Whether showing explanations leads to a significant increase in performance is assessed with McNemar's test (McNemar, 1947).

3.4. Summary

This chapter described the automated metrics and the user study used to assess the quality of explanations. Existing experiments for evaluating faithfulness expect that the individual features selected by the explanation method can be ranked by importance. This is not possible for Anchor explanations: For the pointing game, hits are attributed proportionally to tokens specified by Anchor explanations. In the token removal experiment, where tokens are removed in order of feature importance for feature importance type explanations, they are removed in random order for Anchor.

Stability is assessed in three separate experiments: Consistency across re-runs and continuity (across explanations for small perturbations) are evaluated with an agreement metric. Contrastivity is assessed with controlled synthetic data checks, verifying whether the explanations align with the data generation process (Nauta et al., 2023).

For assessing usefulness, the user study design in Hase and Bansal (2020) is extended by a novel document selection strategy that aims to compensate for the complexity of the task for human annotators. Perceived usefulness is measured with Lickert scale items.

The upcoming chapter describes the setup for generating explanations.

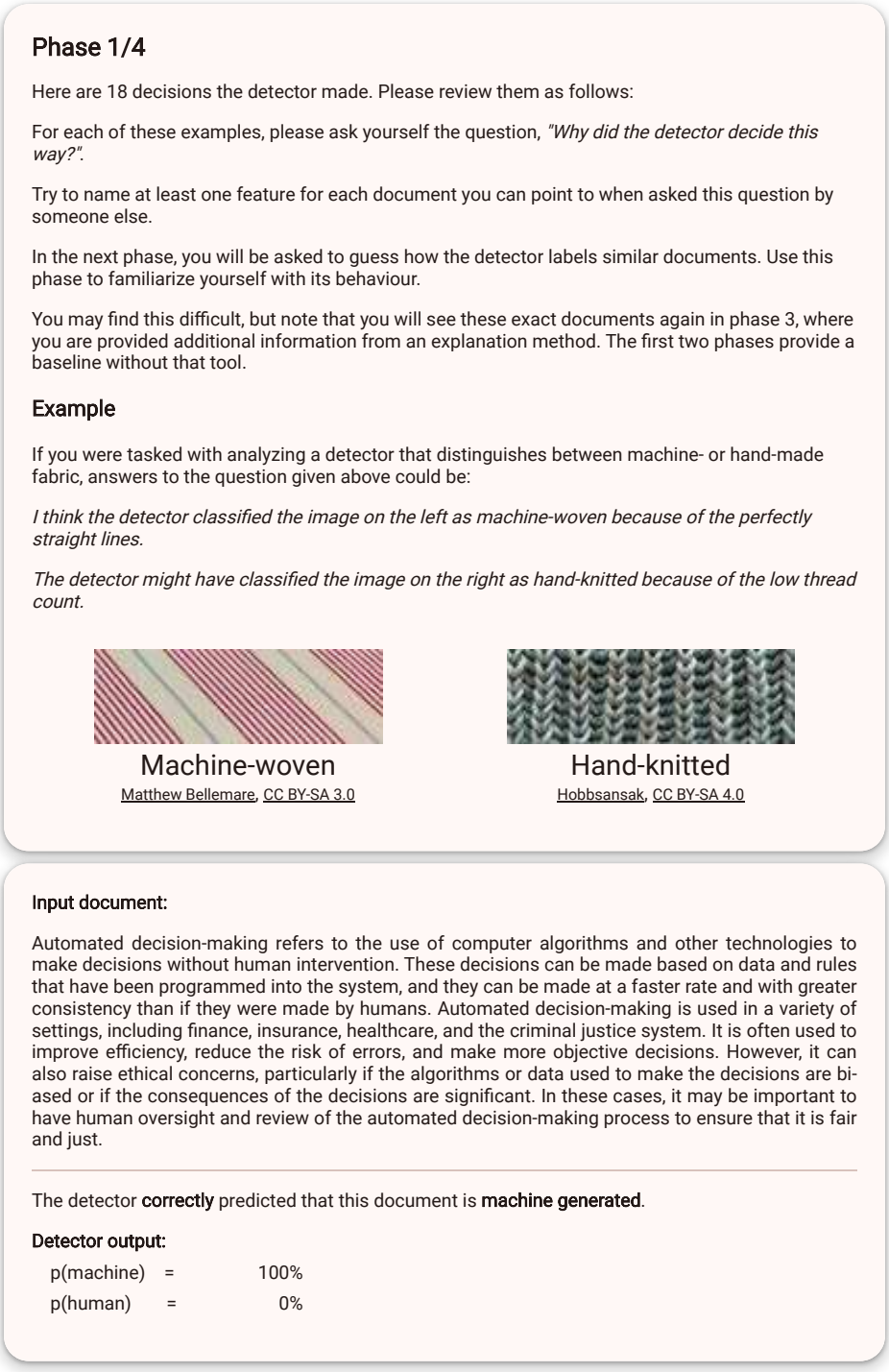


Figure 3.3.: Instructions and interface for phase 1

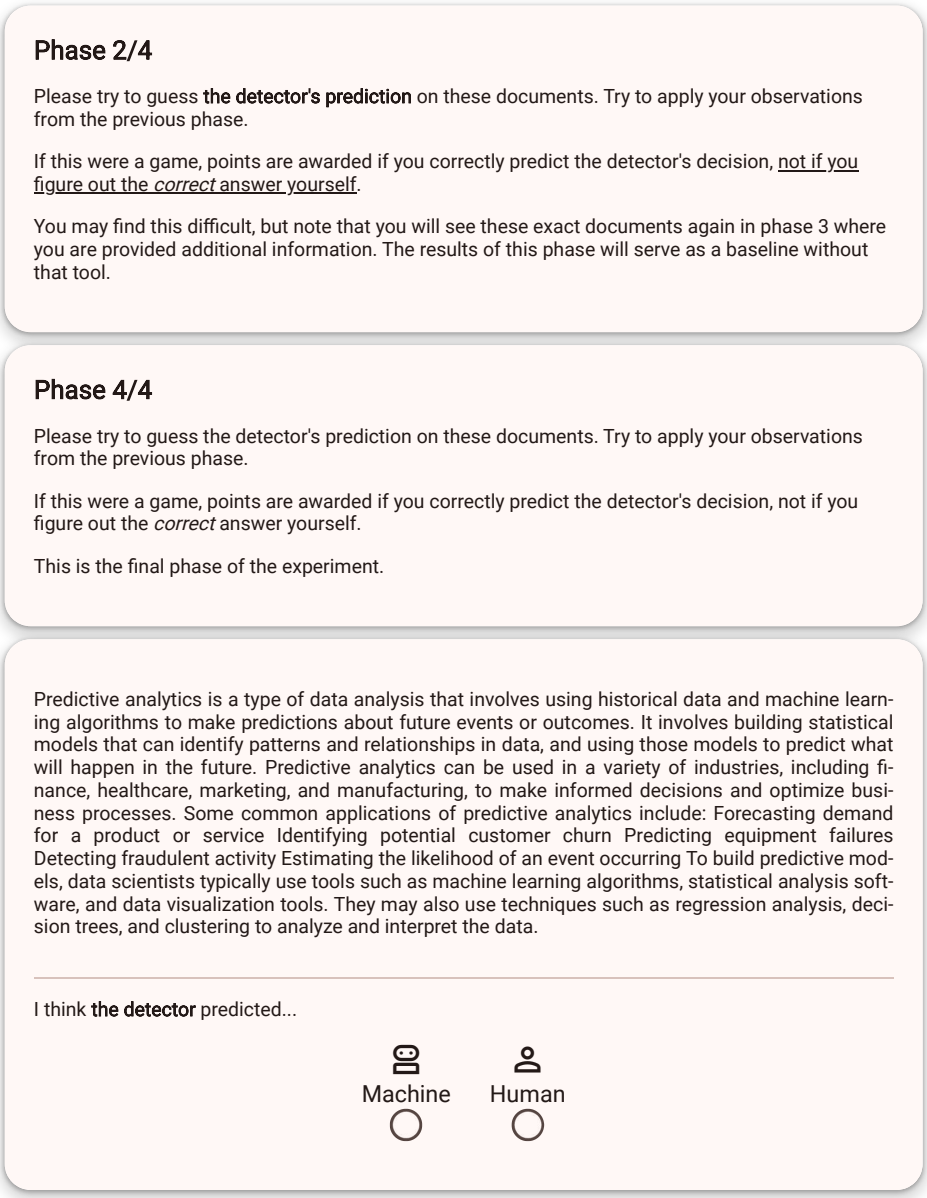


Figure 3.4.: Instructions and interface for phases 2 and 4

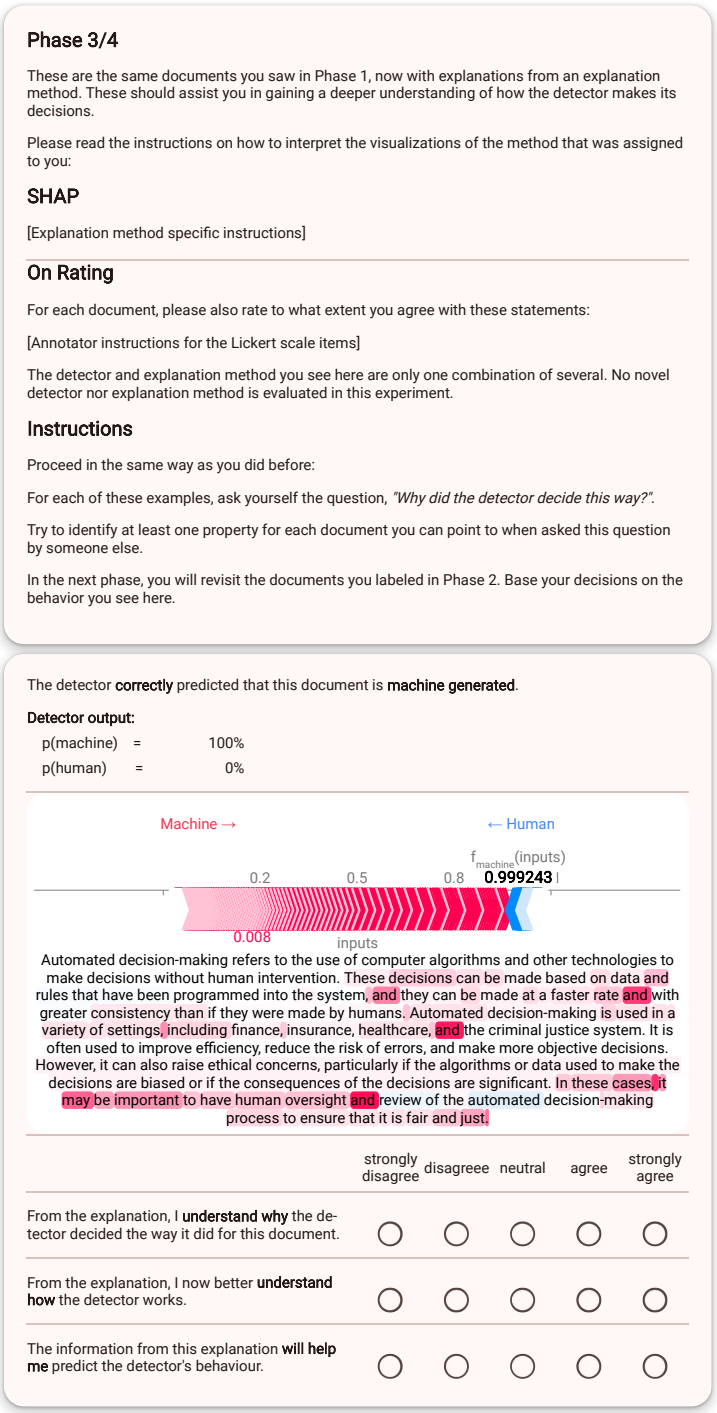


Figure 3.5.: Instructions and interface for phase 3. Instructions on the explanation method and Lickert scale items are omitted from this figure but provided in Appendix A and Figure 3.1.

4. Technical Details and Experimental Setup

The methods described in Chapter 3 are used to evaluate explanations from three explanation methods for decisions from three detectors. This chapter describes how these detectors and explanation methods are set up and provides details on the dataset and user study.

4.1. Dataset

A subset of the H3 dataset by Guo et al. (2023) is used for the experiments. They add ChatGPT written answers to question answering datasets from various domains[1]. All questions from WikiQA and ELI5 are removed, as the source datasets contain crawling artifacts that make identifying human texts trivial[2]. Additionally, only documents that are between 50 and 150 words long are kept. This is to ensure a sufficient input length for the zero-shot detector. 1016 documents remain. Due to the high computation time involved with generating explanations, only 30% of documents (N=305) are used. As discussed in Section 3, additional synthetic datasets are derived from this set. The split has an equal number of machine-generated and human-written documents.

4.2. Detectors

Three open-source black-box detectors are used for evaluation here. Forks of these methods, that interface with the experiments, are made available with this thesis.

Guo (Guo et al., 2023)

With their dataset, Guo et al. also ship a fine-tuned RoBERTa model (Liu et al., 2019b). The accuracy of this detector on the sub-set for which explanations were generated is 0.99.

[1] ELI5 (Fan et al., 2019), WikiQA (Yang et al., 2015), FiQA (Maia et al., 2018), Medical Dialog (He et al., 2020) and a novel dataset obtained by crawling articles on computer science from Wikipedia (Guo et al., 2023).

[2] E.g., extra whitespace characters at the end of sentences; markdown-style formatted URLs in ELI5, which consists of documents from a Q&A Reddit forum where users frequently link additional resources.

Solaiman (Solaiman et al., 2019)

This detector was fine-tuned on the Webtext and GPT-2 output datasets (Solaiman et al., 2019, Section 2.1.1), but uses the same base model as Guo et al. (2023). It is therefore treated as an out-of-distribution detector. Its accuracy is 0.92.

DetectGPT (Mitchell et al., 2023)

This is a zero-shot method. It is set up here with a smaller language model (pythia-70m: Biderman et al., 2023) as suggested by Mireshghallah et al. (2023) and accomplishes an accuracy of 0.74. This, and reducing the number of perturbations per evaluation from 100 to 5 is done to reduce inference time from roughly 6.3 (15.8 with GPT-2) to 0.9 seconds per document. This is required to make obtaining explanations feasible[3]. A comparison with the original implementation is provided in Appendix B.

4.3. Explanation Methods

The individual explanation methods were set up as follows:

LIME

LIME trains a local surrogate model on a dataset of detector output on perturbations (Section 2.2.1). The number of samples to use for fitting this model, and its size, have to be chosen manually. The number of samples was set to 1k (default 5k) for the RoBERTa-based detectors and 500 for the zero-shot method to match SHAP's runtime[4]. The default number of features of 10 appears appropriate for the document length used here (50-150 tokens).

SHAP

The implementation of SHAP used in this paper (Partition Explainer: Lundberg and Lee, 2017) computes Owen values (Owen, 1977) as a measure of feature importance. Explanations are generated with default parameters.

Anchor

Among valid rules as outlined in Section 2.2.2, Anchor's search algorithm attempts to select those that apply to the highest proportion of perturbations in the local neighborhood. In its default implementation, Anchor is often unable to terminate for a single document within an hour of runtime with these detectors. The following strategies are

[3] With a perturbation-based explanation method that is allowed 1,000 samples, the default parameters for this detector would require 10^5 generations with GPT-2 for a single explanation.

[4] Resulting in an average of ≈30s for the RoBERTa-based detectors and ≈415s for DetectGPT per explanation on the original documents.

employed here to make the computation of Anchors feasible: A low target level of precision is chosen ($\tau = 0.75$) and a limit on the number of samples used during construction is imposed (200 samples per candidate Anchor). This effectively terminates the search for good Anchors early. The fallback search for the *best anchor of each size* when failing to reach the target level of precision is not limited in the number of samples so that the value for τ reported by the explanation is truthful. Additionally, DistillBERT is replaced with DistillRoBERTa (Sanh et al., 2020), and at most 20% of tokens are edited per perturbation to increase their coherence. Combined, this reduces the computation time to an average of roughly 5 minutes per explanation for the RoBERTa-based detectors and 15 minutes for DetectGPT on the original documents. A version of Anchor that implements these changes is made available with this thesis.

Perturbation Strategy

An experiment that tests different perturbation strategies (Figure 4.1) did not single out a method that works equally well for all detectors. In the interest of consistency, perturbations are generated by replacing tokens with the mask token of the detector's tokenizer for both LIME and SHAP throughout all experiments. Also note that token deletion, or replacement with whitespace characters, can lead to perturbations that are too short for the zero-shot method[5].

Anchor offers masking with a specified token or to perturb with a language model. The latter strategy was used, as it was found to terminate considerably faster with these detectors.

4.4. User Study

36 participants (B.Sc., M.Sc., and PhD students) with a background in computer science and English reading proficiency at the C1 CEFR level or higher were recruited for the user study. Of them, 27 stated that they had never worked with explanation methods before. Copies of the instructions on the individual explanation methods shown to users are provided in Appendix A. Participants were offered a compensation of €10 for their participation. The experiment was conducted online through a purpose-built form (Figures 3.3, 3.4 and 3.5).

4.5. Summary

The H3 dataset from Guo et al. (2023) is used to evaluate the RoBERTa based detectors of Guo et al. (2023) and Solaiman et al. (2019), and DetectGPT (Mitchell et al., 2023, Section 2.1.3). For all experiments discussed in the next Chapter, SHAP and LIME ex-

[5]For short or incoherent text, DetectGPT will often fail to generate valid perturbations with T5 and fall back to replacement with random tokens from the vocabulary after three attempts in the implementation used in this thesis. This considerably slows down inference.

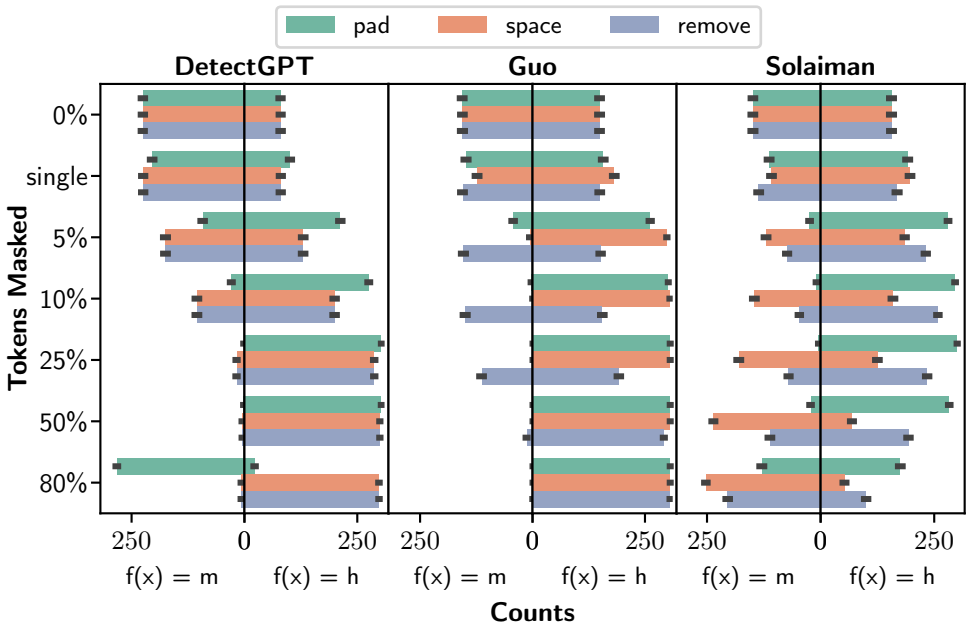


Figure 4.1.: Perturbation strategy: Shift in detector output at different percentages of tokens randomly removed or replaced. The original dataset is balanced, the counts at 0% of tokens removed serve as a baseline. Mean of 10 runs

planations are generated by perturbation with the mask token of the detector's tokenizer. Anchor is set up to perturb with a language model.

5. Results

This chapter summarizes the results from the experiments introduced in Chapter 3. Only aggregate results per explanation method are discussed here. Additional results for all 9 detector-explanation method pairings are provided in Appendix B for reference.

	Pointing Game	**Token Removal**		**Consistency**	**Continuity**	**Contrastivity**		
	Acc_{pg}	$\Delta_{\text{right},k=10}$	$\Delta_{\text{wrong},k=10}$	α	α	c_α	c_{inter}	c_{intra}
Random	0.565	51.2%	75.1%	-0.167	-0.137	-0.362	0.486	0.498
LIME	0.546	46.9%	57.3%	0.192	0.459	0.079	0.598	0.587
Anchor	0.589	28.3%	**100.0**%	-	-	-	-	-
SHAP	**0.692**	**50.4**%	62.3%	**1.000**	**0.854**	0.360	**0.799**	**0.774**

Table 5.1.: Results for the faithfulness and stability experiments

5.1. Faithfulness

In the **pointing game**, LIME fails to outperform the baseline of random feature importance scores (Table 5.1). A series of binominal tests (H_0 : No difference between one method and the next best in the ranking; Sydorova et al., 2019) suggests that the difference between SHAP and Anchor, as well as the one between Anchor and the random baseline are significant ($p < 0.05$). LIME is not significantly worse than the random baseline ($p=0.143$).

For the **token removal experiment**, results for initially correct predictions ($f(d_i) = y_i$) and initially wrong predictions ($f(d_i) \neq y_i$) are plotted separately in Figure 5.1 to allow for consistent interpretation of accuracy scores (Arras et al., 2016). Table 5.1 reports the drop in accuracy at $k = 10$ tokens removed ($\Delta_{k=10}$), corresponding to the maximum number of tokens Anchor and LIME include in their explanations. The mean score for five explanations with random feature importance vectors is provided as a baseline. For $f(d_i) = y_i$, the most important features are removed first. A faithful (feature importance type) explanation method should have a steep drop in accuracy. For $k < 10$ tokens removed, the accuracy of LIME drops slightly faster than that of SHAP. The average accuracy drops below 50% for SHAP at roughly 10 tokens masked. With Anchor, the accuracy drops slower than for all other methods.

For $f(d_i) \neq y_i$, one expects the accuracy to increase. Note that few initially wrong examples are available (105 vs. 810 initially correct cases) given the high accuracy of two detectors. The average accuracy of LIME and SHAP does not increase faster than the random baseline here. Anchor flips the label in all instances and thus archives a perfect score at $k = 10$ for initially wrong examples.

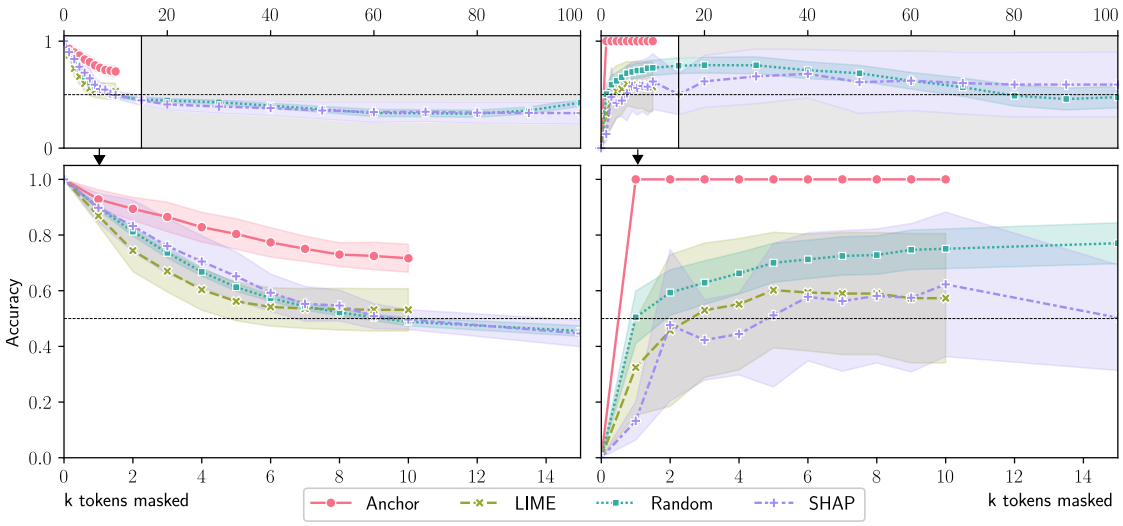


Figure 5.1.: Accuracy at k tokens masked. A faithful explanation method should feature a steep decline (left) or steep incline (right). Only SHAP explanations cover more than 10 tokens (top). Mean across all detectors, error bars at ± 1 standard error

5.2. Stability

Consistency For Krippendorff's α, a score of 0 reflects an agreement by chance, 1 perfect agreement, and negative values systematic disagreement (Krippendorff, 1970). Scoring 0.192, LIME's consistency is far below what could be considered reliable. SHAP (Partition Explainer) is deterministic.

Continuity SHAP's explanations remain stable under small perturbations ($\alpha > 0.8$). Those from LIME do not, but agree with each other better than explanations from the random baseline.

Contrastivity The reliability score of SHAP explanations on matching parts of d_i and d_i^{Ω} (c_α) is higher than that of LIME, but considerably lower than in the consistency and continuity experiments. SHAP ranks higher in both synthetic data checks c_{inter} and c_{intra}.

5.3. Usefulness

Document Selection Table 5.2 reports the document- and explanation similarity of pairs obtained with the proposed selection strategy. Pairs of explanations are significantly

more similar than random selections, while not featuring overly similar documents. This should increase the feasibility of the experiment for users.

(a) Document Similarity

	Set	Method	Random	Increase
Jaccard	Solaiman	0.17	0.12	**0.05**
	Guo	0.14	0.12	**0.02**
	DetectGPT	0.14	0.11	**0.03**
Cosine TF-IDF	Solaiman	0.12	0.08	**0.05**
	Guo	0.12	0.09	**0.04**
	DetectGPT	0.12	0.08	**0.04**

(b) Explanation Similarity

	Set	Method	Random	Gain
Cosine Similarity of FI-Features	Solaiman	0.31	0.17	**0.15**
	Guo	0.38	0.22	**0.16**
	DetectGPT	0.24	0.11	**0.14**
# Matching Anchors	Solaiman	0.33	0.09	**0.24**
	Guo	0.89	0.42	**0.47**
	DetectGPT	0.28	0.06	**0.22**

Table 5.2.: Similarity between pairs (a_j, b_j) in the datasets for the user study against the mean similarity of 10 random selections ($p < 0.05$ bold). Cosine similarity is given as the mean of SHAP and LIME

	Forward Simulation				Perceived Usefulness		
	Without	With	Change	p	Q1: *Why*	Q2: *How*	Q3: *Helpful*
LIME	0.741	0.644	-13.12%	0.006	**3.60**	**3.37**	**3.31**
Anchor	0.694	0.699	0.67%	1.000	2.57	2.48	2.51
SHAP	0.755	0.778	**3.07%**	0.551	3.06	2.86	2.84

Table 5.3.: Results from the user study. 5-point Lickert scale (3 = neutral, 5 = strongly agree)

Forward Simulation Table 5.3 shows the change in user accuracy from phase 2 to phase 4 per explanation method and the results from McNemar's tests. SHAP is the best-performing method, followed by Anchor and LIME. The increase in user accuracy for both SHAP and Anchor is not significant at $p < 0.05$. Participants who have seen LIME explanations perform 13.12% worse after being shown explanations than they did before (p=0.006).

Perceived Usefulness Conversely, users rated LIME best, SHAP second, and Anchor third across all three questions assessing perceived usefulness. For SHAP, users only tended to agree with the first item, where they were asked whether they could *understand why* the detector decided the way it did from an explanation (Figure 3.1). They disagreed with the two other items, asking whether they better *understood how* the detector works

and whether they thought the information from the explanation *would help them* perform better in the second round of annotation. For LIME, they tended to agree, for Anchor explanations to disagree with all three statements on average.

5.4. Summary

SHAP is the best-performing method in the experiments for faithfulness and stability. As set up here, LIME produces considerably less stable explanations than SHAP. LIME, perceived as most useful by users, scores the worst in terms of user performance at predicting the detectors' behavior. Anchor, perceived as least useful by users, outranks LIME in terms of performance in the user study. LIME fails to surpass the random baseline in the pointing game, and Anchor does so for initially right predictions in the token removal experiment.

6. Discussion

This chapter provides an interpretation of the results, discusses implementation details that explain some of the observed behavior, and offers suggestions for future evaluations of explanation quality.

6.1. Faithfulness

Only the explanations from SHAP and Anchor are in line with the data generation strategy more often than the random baseline in the pointing game. The token removal experiment does not produce the same ranking: One would expect masking tokens selected by Anchor to change the prediction more frequently than random masking, as the definition states that altering tokens not part of an Anchor should not affect the detector's prediction. While a drop or increase in accuracy can therefore inform about faithfulness for Anchor, one should, however, expect the change in accuracy to be less pronounced as for the other methods: Anchor does not attempt to identify a set of tokens that affect the decision most. It merely aims to provide a set that is *important enough* to guarantee a certain outcome. It can be argued that this setup is consequently not well suited for comparing Anchor explanations with feature importance ones. Rather than randomly removing tokens when constructing d_i^k for rule-based explanations, future experiments could create all possible d_i^k and use the one with the highest change in accuracy. Note that this setup would have been prohibitively expensive with the zero-shot method included in this thesis.

Anchor outperforms the random baseline and the other methods for initially wrong predictions. The detectors do not produce enough initially wrong predictions on this dataset to derive further insights from this.

6.2. Stability

The partition tree used by SHAP's Partition Explainer is obtained with a deterministic algorithm. SHAP is therefore able to accomplish a perfect score in the **consistency** experiment. To increase consistency for LIME, one should increase the number of samples so that the linear model is more likely to converge. Its parameters were chosen to match SHAP's runtime here. LIME scores higher in the **continuity** experiment than in the one assessing consistency. This may be partially due to an implementation detail: To ensure reproducibility, the random number generator is re-initialized to the same state for every explanation in the continuity experiment. LIME's perturbation method relies

on random sampling and will consequently produce similar output, if two documents are of the same length and differ only by one token.

Lower similarity scores c_α in the **contrastivity** experiment are desirable but cannot be arbitrarily low as the explanation method should remain faithful. That LIME's score is lower than that of SHAP for this metric is in line with expectations: having a limited number of features at its disposal, one would expect it to focus on the new part $d_i^{\Omega-}$ to remain faithful. SHAP can attribute one feature importance score per token, which should allow it to be more consistent on the common part than LIME. SHAP's explanations align with the data generation process more often than those of LIME (c_{inter}). It also appears considerably better than LIME at identifying that the new part is responsible for flipping the label (c_{intra}). Based on the results from these three experiments, SHAP seems to be more sensitive and to produce more stable explanations than LIME.

6.3. Usefulness

Users' perceived usefulness of a method did not align with their performance. This is in line with the observations of Hase and Bansal (2020) for the tasks of sentiment analysis and income prediction. Potential causes for the difference in user ratings between SHAP and LIME are either the shorter length of LIME explanations or the supplementary plots presented by the explanation methods: SHAP visualizes feature importance values in *force plots* (Figure 3.5) while LIME provides them as a bar chart, which users are likely more familiar with.

While it is plausible that showing LIME explanations leads to a decrease in performance given the results from the other experiments, identifying the exact cause for this behavior would require an additional user study, which is left for future work.

6.4. Summary

The setup for a token removal experiment can inform about faithfulness for Anchor explanations but is not well-suited for comparing them to feature importance type ones. The implementation of SHAP used in this thesis is deterministic for identical inputs. When LIME's parameters are adjusted to match SHAP's runtime, LIME does not produce sufficiently consistent explanations. More experiments are required to determine what caused the difference in user ratings between the two feature importance type explanation methods SHAP and LIME.

7. Conclusion

This thesis provides the first dedicated evaluation of explanation methods for detectors of machine-generated text. It evaluates SHAP, LIME, and Anchor explanations with automated metrics and a user study. To enable this analysis, existing detection methods were surveyed, illustrating that current high-performing methods operate as black-box models. Subsequently, past attempts to explain decisions from these detectors were categorized by their research objectives and types of analyses. It was found that most researchers use SHAP or LIME and examine individual outcome explanations in manual qualitative analyses. This thesis focused on evaluating the dimensions of faithfulness, stability, and usefulness, which are particularly relevant for this type of analysis.

Existing evaluation methods for assessing these aspects were identified and adapted to allow for a comparison of all analyzed explanation methods. Specifically, two experiments for testing faithfulness, originally formulated for feature importance type explanations, were extended to Anchor explanations. Stability was assessed with perturbations from a language model in experiments specially constructed for the task. Usefulness, defined here as the users' ability to predict the behavior of a detector with the help of explanations, was evaluated in a user study. Given the difficulty of the task for humans, it would have been unlikely for users to obtain a comprehensive understanding of the detector's behavior or relevant features and successfully apply this knowledge to new instances with a real-world dataset of human and machine-generated text. To keep the task feasible for human annotators, users were presented with pairs of documents based on explanation similarity rather than randomly selected documents. Additionally, perceived usefulness was measured with Likert scale items.

The experiments were conducted with three black-box detectors (two RoBERTa-based models and DetectGPT) and a dataset of human-written and ChatGPT-generated documents. It was found that SHAP fulfills the theoretically motivated properties of good explanations best. However, no explanation method led to a significant increase in performance in the user study. For LIME, users' perceived usefulness of explanations did not align with the measured decrease in performance, nor with the results from the faithfulness and stability experiments.

The results from the user study suggest that combinations of these methods and detectors should not be implemented into systems that face untrained users in their current form. Note, however, that this experimental setup was primarily designed to determine whether showing explanations leads to an increase in user performance. Users did not have knowledge about the model architecture, were shown a relatively small set of explanations, and spent a relatively short time interpreting them. This is not representative of past attempts at model- or dataset inspection, where researchers have access to explanations for all documents, prior knowledge about the model and task, as well as access

to an unlimited number of examples. Based on the results from the faithfulness and stability experiments, SHAP can be recommended for this application.

Accommodating the comparatively high runtime of LIME and Anchor was a limiting factor for experimental design, and controlling for it arguably affected the quality of their output. Especially for Anchor explanations, future work should assess explanation quality without imposing this constraint.

Bibliography

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

David Alvarez Melis and Tommi Jaakkola. 2018. Towards Robust Interpretability With Self-Explaining Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining Predictions of Non-Linear Classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany. Association for Computational Linguistics.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. Generating Fact Checking Explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364. Association for Computational Linguistics.

Seojin Bang, Pengtao Xie, Heewook Lee, Wei Wu, and Eric Xing. 2021. Explaining A Black-box By Using A Deep Variational Information Bottleneck Approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11396–11404.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165v1 [cs]*.

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. Association for Computational Linguistics.

Evan Crothers, Nathalie Japkowicz, Herna Viktor, and Paula Branco. 2022. Adversarial Robustness of Neural-Statistical Features in Detection of Generative Transformers. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. ISSN: 2161-4407.

Evan N. Crothers, Nathalie Japkowicz, and Herna L. Viktor. 2023. Machine-Generated Text: A Comprehensive Survey of Threat Models and Detection Methods. *IEEE Access*, 11:70977–71002.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. *arXiv:1912.02164v1 [cs]*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805v1 [cs]*.

Finale Doshi-Velez and Been Kim. 2017. Towards a Rigorous Science of Interpretable Machine Learning. *arXiv:1702.08608v2 [stat.ML]*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Leon Fröhling and Arkaitz Zubiaga. 2021. Feature-based detection of automated language models: Tackling GPT-2, GPT-3 and Grover. *PeerJ Computer Science*, 7:e443.

Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2023. Repairing the Cracked Foundation: A Survey of Obstacles in Evaluation Practices for Generated Text. *Journal of Artificial Intelligence Research*, 77:103–166.

Josh A. Goldstein, Girish Sastry, Micah Musser, Renee DiResta, Matthew Gentzel, and Katerina Sedova. 2023. Generative Language Models and Automated Influence Operations: Emerging Threats and Potential Mitigations. *arXiv:2301.04246v1 [cs]*.

Ken Gu and Akshay Budhkar. 2021. A Package for Learning on Tabular and Text Data with Transformers. In *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*, pages 69–73, Mexico City, Mexico. Association for Computational Linguistics.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2022. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.*, 51(5).

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. *arXiv:2301.07597v1 [cs.CL].*

Peter Hase and Mohit Bansal. 2020. Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552. Association for Computational Linguistics.

Xuehai He, Shu Chen, Zeqian Ju, Xiangyu Dong, Hongchao Fang, Sicheng Wang, Yue Yang, Jiaqi Zeng, Ruisi Zhang, Ruoyu Zhang, Meng Zhou, Penghui Zhu, and Pengtao Xie. 2020. MedDialog: Two Large-scale Medical Dialogue Datasets. *arXiv:2004.03329v2 [cs.LG].*

Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. 2019. Metrics for Explainable AI: Challenges and Prospects. *arXiv:1812.04608v2 [cs.AI].*

Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. 2023. Measures for explainable AI: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-AI performance. *Frontiers in Computer Science*, 5:1096257.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. *arXiv:2301.04246v1 [cs].*

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic Detection of Generated Text is Easiest when Humans are Fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822. 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. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *arXiv:1909.05858v1 [cs].*

Peter Kowalczyk, Marco Röder, Alexander Dürr, and Frédéric Thiesse. 2022. Detecting and Understanding Textual Deepfakes in Online Reviews. In *Proceedings of the 55th Hawaii International Conference on System Sciences.*

Klaus Krippendorff. 1970. Bivariate Agreement Coefficients for Reliability of Data. *Sociological Methodology*, 2:139–150.

Tharindu Kumarage, Amrita Bhattacharjee, Djordje Padejski, Kristy Roschke, Dan Gillmor, Scott Ruston, Huan Liu, and Joshua Garland. 2023a. J-Guard: Journalism Guided Adversarially Robust Detection of AI-generated News. *arXiv:2309.03164v1 [cs].*

Tharindu Kumarage, Joshua Garland, Amrita Bhattacharjee, Kirill Trapeznikov, Scott Ruston, and Huan Liu. 2023b. Stylometric Detection of AI-Generated Text in Twitter Timelines. *arXiv:2303.03697v1 [cs.CL].*

Biagio La Rosa, Roberto Capobianco, and Daniele Nardi. 2020. Explainable Inference on Sequential Data via Memory-Tracking. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2006–2013. International Joint Conferences on Artificial Intelligence Organization. Main track.

Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. 2020. Robust and Stable Black Box Explanations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5628–5638. PMLR.

Hui Liu, Qingyu Yin, and William Yang Wang. 2019a. Towards Explainable NLP: A Generative Explanation Framework for Text Classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5570–5581, Florence, Italy. Association for Computational Linguistics.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2023a. CoCo: Coherence-Enhanced Machine-Generated Text Detection Under Data Limitation With Contrastive Learning. *arXiv:2212.10341v2 [cs.CL].*

Yikang Liu, Ziyin Zhang, Wanyang Zhang, Shisen Yue, Xiaojing Zhao, Xinyuan Cheng, Yiwen Zhang, and Hai Hu. 2023b. ArguGPT: evaluating, understanding and identifying argumentative essays generated by GPT models. *arXiv:2304.07666v2 [cs].*

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692v1 [cs.CL].*

Zeyan Liu, Zijun Yao, Fengjun Li, and Bo Luo. 2023c. Check Me If You Can: Detecting ChatGPT-Generated Academic Writing using CheckGPT. *arXiv:2306.05524v1 [cs.CL].*

Scott Lundberg. 2019. shap.PartitionExplainer. [Accessed 07-05-2024].

Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 634–642. Curran Associates, Inc.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion

Mining and Question Answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1941–1942, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Quinn McNemar. 1947. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157.

Fatemehsadat Mireshghallah, Justus Mattern, Sicun Gao, Reza Shokri, and Taylor Berg-Kirkpatrick. 2023. Smaller Language Models are Better Black-box Machine-Generated Text Detectors. *arXiv:2305.09859v4 [cs.CL]*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection Using Probability Curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24950–24962. PMLR.

Sandra Mitrović, Davide Andreoletti, and Omran Ayoub. 2023. ChatGPT or Human? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text. *arXiv:2301.13852v1 [cs]*.

Edoardo Mosca, Mohamed Hesham Ibrahim Abdalla, Paolo Basso, Margherita Musumeci, and Georg Groh. 2023. Distinguishing Fact from Fiction: A Benchmark Dataset for Identifying Machine-Generated Scientific Papers in the LLM Era. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 190–207, Toronto, Canada. Association for Computational Linguistics.

Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice Van Keulen, and Christin Seifert. 2023. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. *ACM Computing Surveys*, 55(13s):1–42.

Dong Nguyen. 2018. Comparing Automatic and Human Evaluation of Local Explanations for Text Classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

Guilliermo Owen. 1977. Values of Games With A Priori Unions. In *Mathematical Economics and Game Theory*, pages 76–88, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alessandro Pegoraro, Kavita Kumari, Hossein Fereidooni, and Ahmad-Reza Sadeghi. 2023. To ChatGPT, or not to ChatGPT: That is the question! *arXiv:2304.01487v1 [cs]*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. MAUVE: Measuring the Gap Between Neural

Text and Human Text using Divergence Frontiers. In *Advances in Neural Information Processing Systems*, volume 34, pages 4816–4828. Curran Associates, Inc.

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating Neural Network Explanation Methods Using Hybrid Documents and Morphosyntactic Agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Melbourne, Australia. Association for Computational Linguistics.

Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. 2020. Learning to Deceive with Attention-Based Explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv:1910.10683v1 [cs, stat]*.

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*, KDD '16, pages 1135–1144, New York, NY, USA. Association for Computing Machinery.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv:1910.01108v4 [cs.CL]*.

L. S. Shapley. 1953. 17. A Value for n-Person Games. In Harold William Kuhn and Albert William Tucker, editors, *Contributions to the Theory of Games (AM-28), Volume II*, pages 307–318. Princeton University Press.

Kanishka Silva and Ingo Frommholz. 2023. What if ChatGPT Wrote the Abstract? - Explainable Multi-Authorship Attribution With a Data Augmentation Strategy. In *Proceedings of the IACT'23 Workshop*, volume 3477, Taipei, Taiwan.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release Strategies and the Social Impacts of Language Models. *arXiv:1908.09203v2 [cs.CL]*.

Jeremy Stribling, Max Krohn, and Dan Aguayo. 2005. SCIgen - An Automatic CS Paper Generator.

Alona Sydorova, Nina Poerner, and Benjamin Roth. 2019. Interpretable Question Answering on Knowledge Bases and Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4943–4951. Association for Computational Linguistics.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A Large Language Model for Science. *arXiv:2211.09085v1 [cs, stat]*.

Alex Tseng, Avanti Shrikumar, and Anshul Kundaje. 2020. Fourier-transform-based attribution priors improve the interpretability and stability of deep learning models for genomics. In *Advances in Neural Information Processing Systems*, volume 33, pages 1913–1923. Curran Associates, Inc.

Shangqing Tu, Chunyang Li, Jifan Yu, Xiaozhi Wang, Lei Hou, and Juanzi Li. 2023. ChatLog: Recording and Analyzing ChatGPT Across Time. *arXiv:2304.14106v1 [cs]*.

A. M. Turing. 1950. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Luoxuan Weng, Minfeng Zhu, Kam Kwai Wong, Shi Liu, Jiashun Sun, Hang Zhu, Dongming Han, and Wei Chen. 2023. Towards an Understanding and Explanation for Mixed-Initiative Artificial Scientific Text Detection. *arXiv:2304.05011v1 [cs]*.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Petzold, William Yang Wang, and Haifeng Chen. 2023. DNA-GPT: Divergent N-Gram Analysis for Training-Free Detection of GPT-Generated Text. *arXiv:2305.17359v1 [cs]*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Peipeng Yu, Jiahan Chen, Xuan Feng, and Zhihua Xia. 2023. CHEAT: A Large-scale Dataset for Detecting ChatGPT-writtEn AbsTracts. *arXiv:2304.12008v2 [cs.CL]*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending Against Neural Fake news. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 9054–9065, Red Hook, NY, USA. Curran Associates Inc.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv:2205.01068v4 [cs.CL]*.

Wangchunshu Zhou, Jinyi Hu, Hanlin Zhang, Xiaodan Liang, Maosong Sun, Chenyan Xiong, and Jian Tang. 2020. Towards Interpretable Natural Language Understanding with Explanations as Latent Variables. In *Advances in Neural Information Processing Systems*, volume 33, pages 6803–6814. Curran Associates, Inc.

A. Annotator Guidelines and Examples

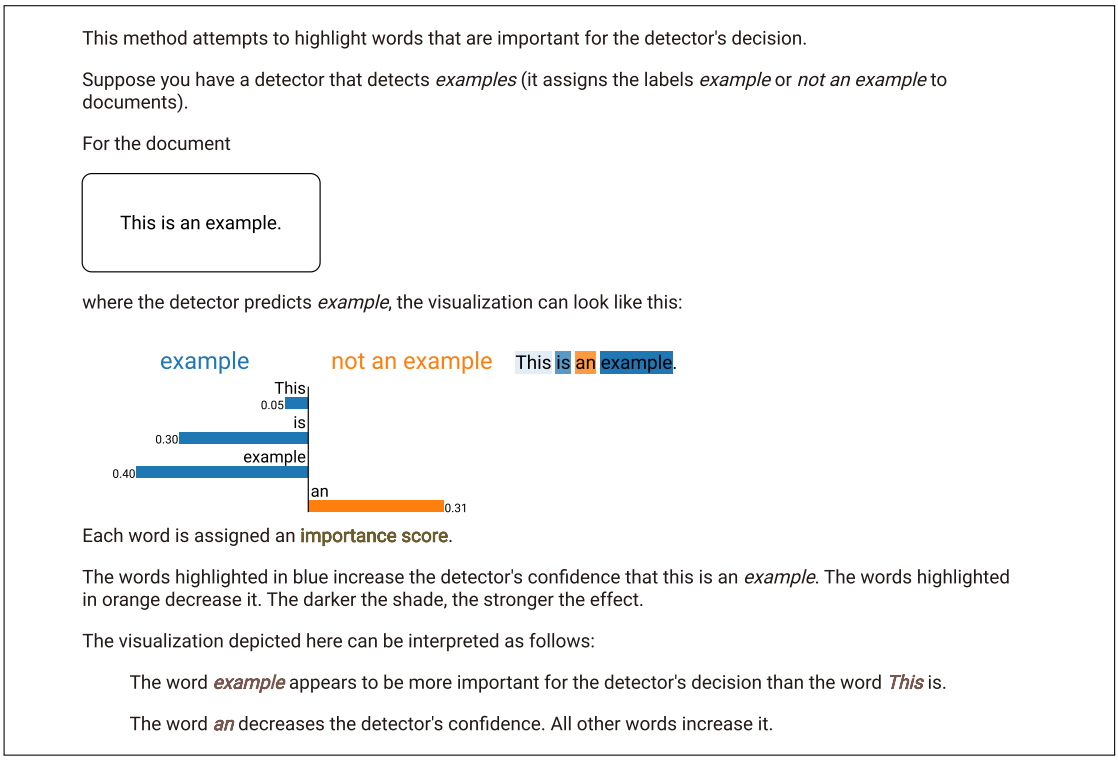


Figure A.1.: Instructions on LIME shown in phase 3

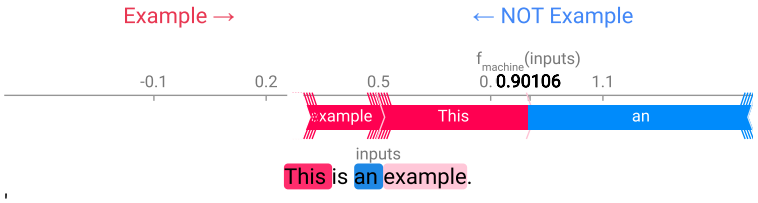
This method attempts to highlight words that are important for the detector's decision.

Suppose you have a detector that detects *examples* (it assigns the labels *example* or *not an example*).

For the document

> This is an example.

where the detector predicts *example*, the visualization can look like this:



Each word is assigned an **importance score**.

The words highlighted in pink increase the detector's confidence that this is an *example*. The words highlighted in blue decrease it. The darker the shade, the stronger the effect.

The visualization depicted here can be interpreted as follows:

> The word ***This*** appears to be more important for the detector's decision than the word ***example*** is.
>
> The word ***an*** decreases the detector's confidence. All other words increase it.

The upper part of the plot provides the same information: the word in blue lowers the confidence that this is an *example*, the words in pink increase it.

Figure A.2.: Instructions on SHAP shown in phase 3

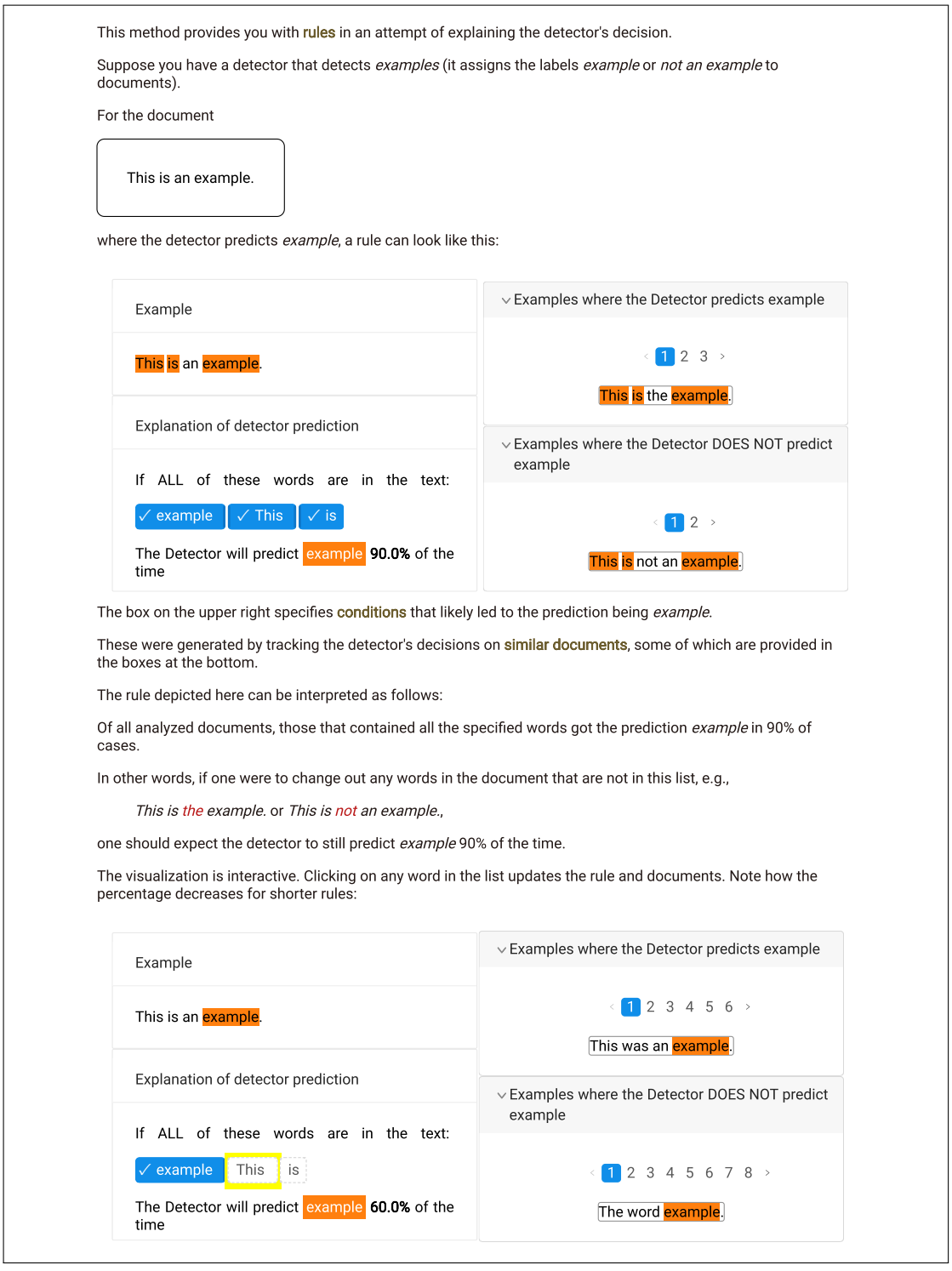
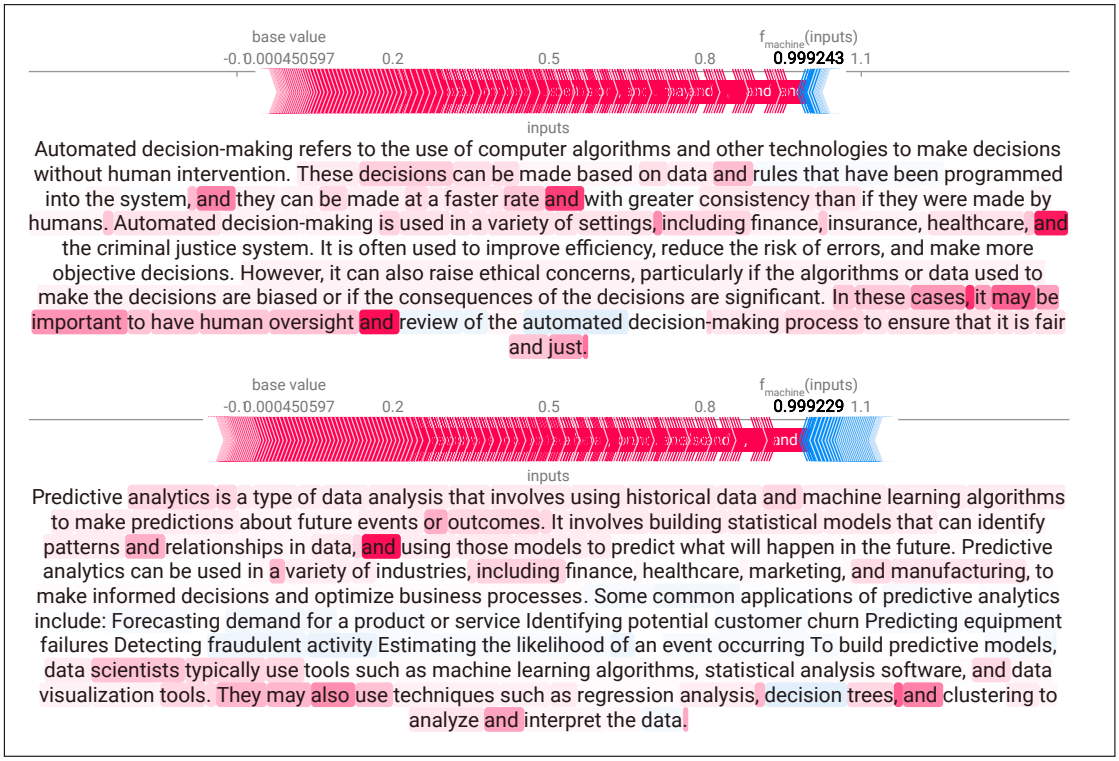
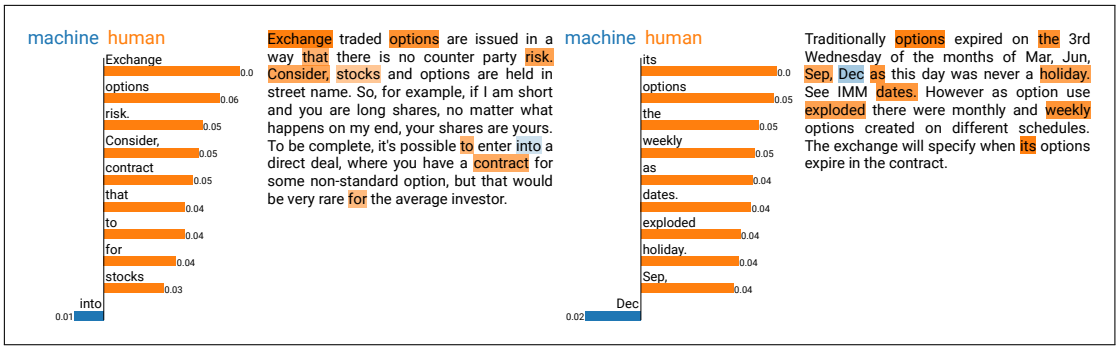


Figure A.3.: Instructions on Anchor shown in phase 3

(a) Set Guo, selected by SHAP; $y_i = f(d_i) =$ machine



(b) Set DetectGPT, selected by LIME; $y_i = f(d_i) =$ human



(c) Set Solaiman, selected by Anchor; $y_i = f(d_i) =$ machine

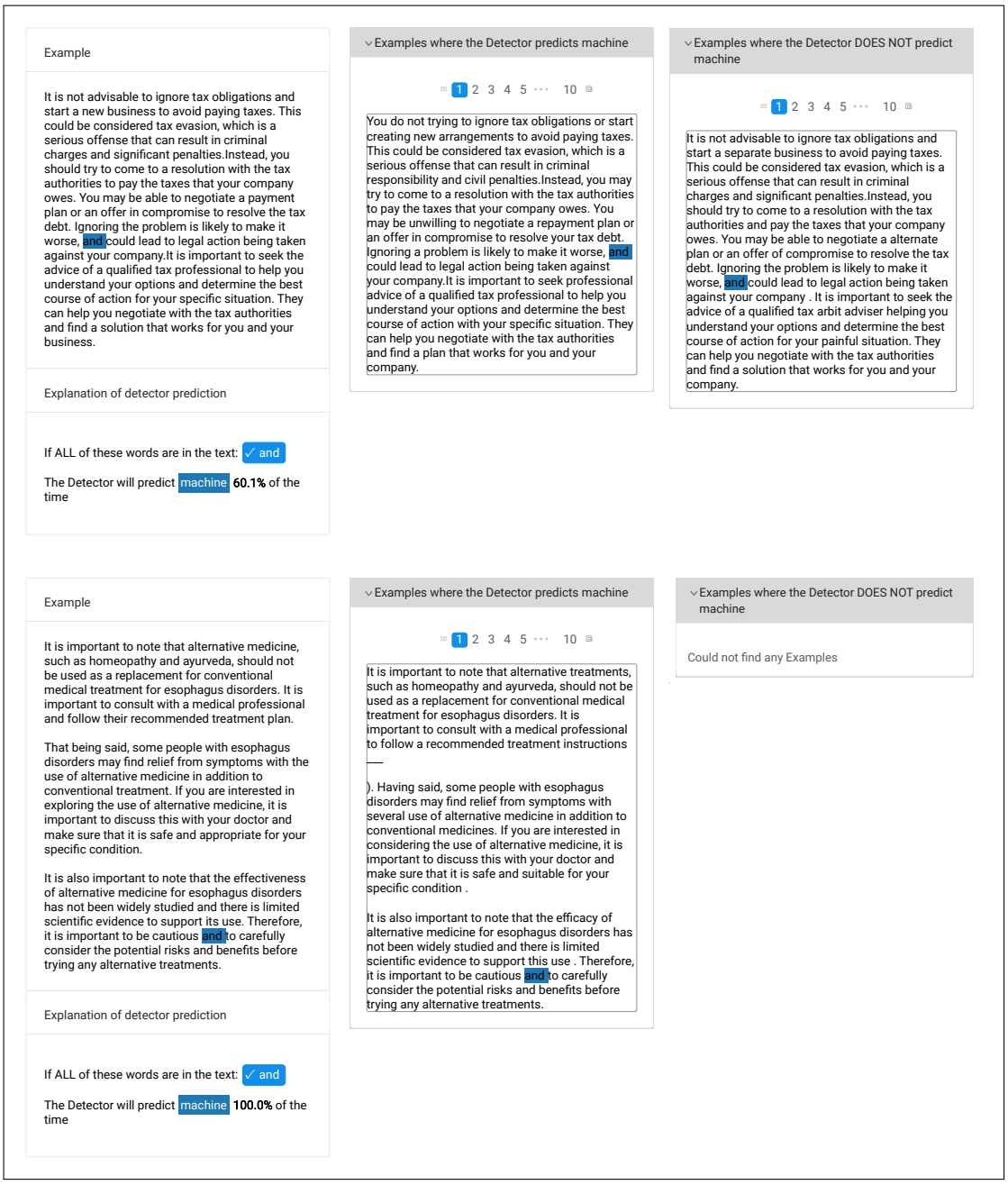


Figure A.4.: Pairs of documents shown to users. Users only see explanations for the first document of each pair and annotate the second one.

B. Complementary Results

	Acc	F1	AUC	TN	FP	FN	TP	ms/evaluation
DetectGPT GPT-2 @100 samples	0.502	0.000	0.500	153	0	152	0	15808
DetectGPT pythia-70m @100 samples	0.705	0.579	0.704	153	0	90	62	6391
DetectGPT pythia-70m (this paper)	0.744	0.664	0.743	150	3	75	77	898
Solaiman	0.921	0.922	0.921	139	14	10	142	19
Guo	0.990	0.990	0.990	153	0	3	149	18

Table B.1.: Performance of the detectors. Note that one could obtain better results for DetectGPT when using GPT-2 by adjusting the classification threshold

(a) Guo

Explainer	Score	p
LIME	0.605	0.168
Random	0.635	0.000
Anchor	0.681	0.000
SHAP	0.812	

(b) Solaiman

Explainer	Score	p
LIME	0.402	0.004
Random	0.484	0.003
Anchor	0.492	0.000
SHAP	0.631	

(c) DetectGPT

Explainer	Score	p
Random	0.577	0.000
Anchor	0.596	0.116
LIME	0.631	0.473
SHAP	0.635	

Table B.2.: Pointing game: Per detector results. P-values from row-wise binomial tests

(a) Consistency

Explainer	Detector	α
SHAP	Solaiman	**1.000**
	Guo	1.000
LIME	Solaiman	0.204
	Guo	0.179
Random	Guo	-0.167
	Solaiman	-0.167

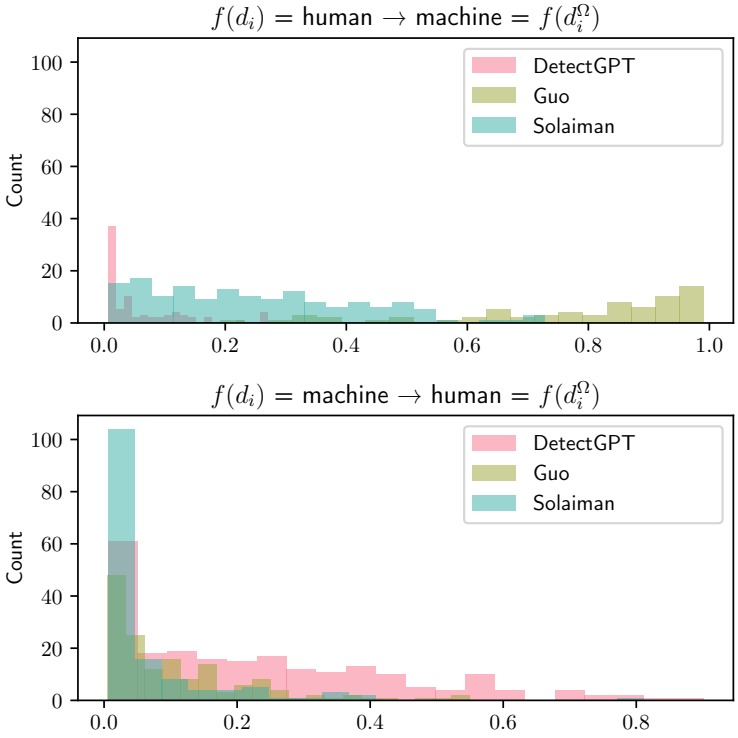
(b) Continuity

Explainer	Detector	α
SHAP	Guo	**0.896**
	Solaiman	0.812
LIME	Guo	0.478
	Solaiman	0.439
Random	Solaiman	-0.137
	Guo	-0.137

Figure B.1.: Consistency and continuity: Per detector results

		$f(d_i) \rightarrow f(d_i^\Omega)$	**n**	α	c_{intra}	c_{inter}
DetectGPT	Random	h → m	800	-0.33	0.48	0.49
		m → h	1990	-0.37	0.50	0.49
	LIME	h → m	80	0.08	0.39	0.74
		m → h	199	0.05	0.62	0.28
	SHAP	h → m	80	0.09	0.59	0.59
		m → h	199	0.05	0.59	0.63
Guo	Random	h → m	90	-0.31	0.46	0.34
		m → h	1530	-0.35	0.49	0.50
	LIME	h → m	9	-0.03	0.78	0.44
		m → h	153	-0.02	0.52	0.73
	SHAP	h → m	9	0.14	1.00	1.00
		m → h	153	0.49	0.93	0.71
Solaiman	Random	h → m	1420	-0.37	0.48	0.51
		m → h	1480	-0.37	0.48	0.51
	LIME	h → m	142	0.24	0.79	0.90
		m → h	148	0.07	0.56	0.48
	SHAP	h → m	142	0.66	0.97	0.94
		m → h	148	0.53	0.88	0.97

(a) Per detector results



(b) Fraction of tokens edited until $f(d_i^\Omega) \neq f(d_i)$

Figure B.2.: Contrastivity: Per detector results and data on the synthetic datasets

		User Acc without	User Acc with	Change	p
LIME	Solaiman	0.681	0.569	-16.33%	0.057
LIME	DetectGPT	0.792	0.681	-14.04%	0.134
LIME	Guo	0.750	0.681	-9.26%	0.359
Anchor	DetectGPT	0.806	0.750	-6.90%	0.289
SHAP	DetectGPT	0.819	0.806	-1.69%	1.000
SHAP	Solaiman	0.667	0.681	2.08%	1.000
Anchor	Solaiman	0.569	0.583	2.44%	1.000
Anchor	Guo	0.708	0.764	7.84%	0.344
SHAP	Guo	0.778	0.847	8.93%	0.227

DetectGPT	Anchor	Q1	2.15
		Q2	2.04
		Q3	2.11
	LIME	Q1	3.62
		Q2	2.97
		Q3	3.08
	SHAP	Q1	2.54
		Q2	2.56
		Q3	2.57

Solaiman	Anchor	Q1	3.17
		Q2	3.18
		Q3	3.15
	LIME	Q1	4.00
		Q2	3.81
		Q3	3.65
	SHAP	Q1	3.33
		Q2	3.39
		Q3	3.29

Guo	Anchor	Q1	2.39
		Q2	2.22
		Q3	2.26
	LIME	Q1	3.17
		Q2	3.32
		Q3	3.19
	SHAP	Q1	3.29
		Q2	2.64
		Q3	2.65

Table B.3.: Forward simulation and perceived usefulness: Per group results. (3 = neutral, 5 = strongly agree)