SYNTAXSHAP: Syntax-aware Explainability Method for Text Generation

Anonymous ACL submission

Abstract

To harness the power of large language models in safety-critical domains we need to ensure the explainability of their predictions. However, despite the significant attention to model interpretability, there remains an unexplored domain in explaining sequence-to-sequence tasks using methods tailored for textual data. This paper introduces *SyntaxShap*, a local, model-agnostic explainability method for text generation that takes into consideration the syntax in the text data. The presented work extends Shapley values to account for parsing-based syntactic dependencies. Taking a game theoric approach, SyntaxShap only considers coalitions constraint by the dependency tree. We adopt a model-based evaluation to compare SyntaxShap and its weighted form to state-of-the-art explainability methods adapted to text generation tasks, using diverse metrics including faithfulness, complexity, coherency, and semantic alignment of the explanations to the model. We show that our syntax-aware method produces explanations that help build more faithful, coherent, and interpretable explanations for predictions by autoregressive models.[1]

1 Introduction

Language model (LM) interpretability has become very important with the popularity of generative AI. Despite the great results achieved by the most recent LMs, there is still a large range of tasks where the models fail, e.g., capturing negations (Truong et al., 2023). Therefore, it is crucial to get a better understanding of the LM reasoning and develop faithful explainability methods. As many LMs have little transparency and their use is restricted to API calls, model-agnostic explainability methods have become the most practical techniques for gaining better insights into LMs.

The SHapley Additive exPlanations (SHAP) framework is popular for generating local explanations thanks to its solid theoretical background and general applicability (Shapley et al., 1953). However, regarding sequence-to-sequence tasks such as next token generation, the usage of SHAP-based methods has not been explored in depth (Mosca et al., 2022). We address this gap and develop a coalition-based explainability method inspired by Shapley values for text generation explanation.

Our explainability method (in Figure 1) considers syntactic word dependencies (de Marneffe et al.). The syntax is important as next-word predictions in autoregressive LMs underlie implicit incremental syntactic inferences, i.e., LMs implicitly capture dependencies in text data (Eisape et al., 2022). In this paper, we investigate if dependency parsing trees can be used in the explainability process as syntactic relational graphs and help shed light on the influence of words on the model's prediction given their syntactic role in the sentence.

We evaluate the explanations on diverse metrics. First, we adapt fidelity, one of the most popular model-based evaluation metrics in xAI (eXplainable AI), to the text generation task and introduce two new metrics to test whether the generated explanations are faithful to the underlying model. Second, we introduce two qualitative evaluation metrics that capture the explanation quality with regard to human expectations, i.e., the coherency of explanations and their semantic alignment. Our evaluation procedure compares our method *SyntaxShap* to state-of-the-art explainability methods. Explanations produced by our method of the next token generation by two popular autoregressive models are more faithful, coherent, and semantically aligned compared to state-of-the-art SHAP-based methods that do not explicitly consider the word dependency for text generation tasks.

To summarize, our contributions are (1) SyntaxShap, a new SHAP-based explainability method that incorporates dependency tree information, (2) quantitative metrics that address LM's stochastic-

[1] Code and data are anonymously available here

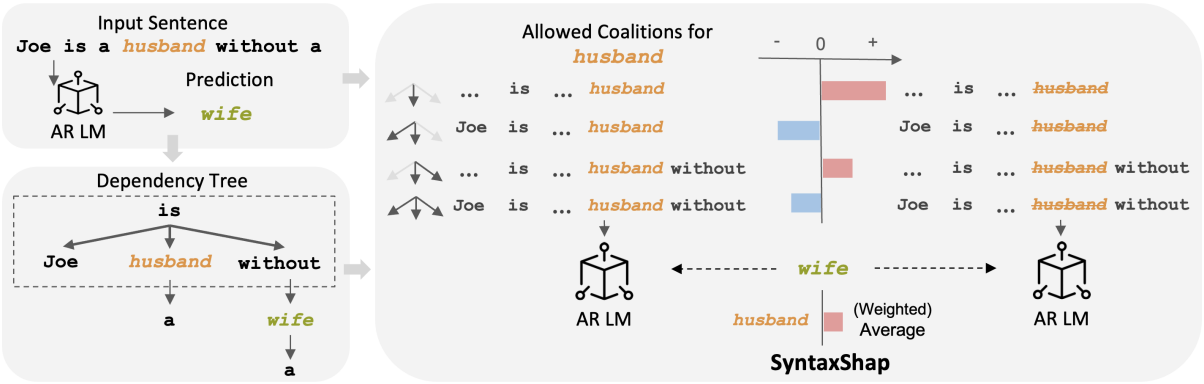


Figure 1: Given an input sentence, an autoregressive language model (AR LM) predicts the next token. The syntax of the sentence is extracted using dependency parsing (spaCy (Honnibal and Montani, 2017)). To measure the importance of the word ***husband*** for the model to predict the next token ***wife***, our method (1) extracts multiple coalitions of words following specific paths in the dependency tree, (2) analyze the contribution of adding ***husband*** to each coalition in the change of probability to predict the next token ***wife***, and (3) average those contributions to compute its final SyntaxShap value.

ity and qualitative metrics to account for human semantic expectations, and (3) an evaluation of the explanation quality on two autoregressive LMs. Our work opens multiple new research directions for future work.

2 Related Work

Explainability in Linguistics Syntax and semantics play an important role in explaining LM outcomes from a linguist perspective. Multiple attempts were made to explore the role of syntactic and semantic representations to enhance LM predictions. Ek et al. (2019) look at the role of syntactic and semantic tags for the specific task of human sentence acceptability judgment. They show that syntactic tags significantly influence the predictions of the LM. In recent years, there has been an increasing interest in methods that incorporate syntactic knowledge into Machine Translation (Ambati, 2008). In addition, Eisape et al. (2022) has shown that next-word predictions from autoregressive neural LMs show remarkable sensitivity to syntax. However, there has been no attempt to account for the syntax in explanations of those LMs for text generation tasks (Mosca et al., 2022). For this reason, we propose to incorporate syntax-based rules to explain AR LM text generation.

SHAP-based explainability in NLP One way to categorize model-agnostic post-hoc explainability methods is to separate them into perturbation-based and surrogate methods (Zhao et al., 2023). Among the most popular surrogate models are LIME and SHAP. The Shapley-value approach (Shapley et al., 1953) provides local explanations by attributing changes in predictions for individual data inputs to the model's features. Those changes can be combined to obtain a better global understanding of the model structure. For text data, available approaches seem mostly tailored to classification settings (Kokalj et al., 2021; Chen et al., 2020).

Shapley values and complex dependencies One underlying assumption of SHAP is feature independence. Confronted with more diverse types of data inputs, newer methods offer the possibility to account for more complex dependencies between features. Frye et al. (2020) propose Asymmetric Shapley values (ASV), which drop the symmetry assumption and enable the generation of model-agnostic explanations incorporating any causal dependency known to be present in the data. Following up with this work, Heskes et al. (2020) propose Causal Shapley values to account more specifically for causal structures behind feature interactions. Chen et al. (2019) construct coalitions based on a graph structure, grouping features with their neighbors or connected nodes. When it comes to text data, words present strong interactions, and their contribution heavily rely on the context. Therefore, feature attributions for textual data should be specifically tailored to account for those complex dependencies. HEDGE is one example of a SHAP-based method addressing the context dependencies specific to text data (Chen et al., 2020). It hierarchically builds clusters of words based on their interactions. While their objective is to cluster words to minimize the loss of faithfulness, i.e.,

prediction change, we propose a new strategy to create coalitions of words that respect the syntactic relationships dictated by the dependency tree. This way, we take into consideration the syntactic dependencies that are the basis of linguistics and which were proven essential for next-word predictions from autoregressive LMs (Eisape et al., 2022).

3 SyntaxShap Methodology

3.1 Objective

Given a sentence of n words $\mathbf{x} = (x_1, ..., x_n)$ and $\hat{\mathbf{y}} = (\hat{y}_1, ..., \hat{y}_m)$ the m generated words by an autoregressive LM f, the objective is to evaluate the importance of each input token for the prediction $\hat{\mathbf{y}}$. We focus on explaining the next token, i.e., $m = 1$. Let $f_y(x)$ be the model's predicted probability that the input data $\mathbf{x}$ has the next token y. Our method produces local explanations.

3.2 Shapley values approach

We adopt a game theory approach to measure the importance of each word x_i to the prediction. The Shapley value approach was first introduced in cooperative game theory (Shapley et al., 1953) and computes feature importance by evaluating how each feature i interacts with the other features in a coalition S. For each coalition of features, it computes the marginal contribution of feature i, i.e., the difference between the importance of all features in S, with and without i. It aggregates these marginal contributions over all subsets of features to get the final importance of feature i.

3.3 Syntax-aware coalition game

Our work focuses on incorporating syntax knowledge into model-agnostic explainability. We adopt a coalition game approach that accounts for these syntactic rules. As illustrated in Figure 1, SyntaxShap computes the contribution of words only considering *allowed* coalitions $\mathfrak{S}$ constraint on the dependency tree structure. We define a coalition S as a set of words or features $\{x_i, i \in [1, n]\}$ from the input sentence x. Given a dependency tree with L levels, $l_i \in [1, L]$ corresponds to the level of word x_i in the tree and $n_l > 0$ the number of words at level l in the tree. To compute the contribution of the words in the tree, SyntaxShap only considers the allowed coalitions $\mathfrak{S} = \bigcup_{l=0}^{L} \mathfrak{S}_l$, where $\mathfrak{S}_l$ is the set of allowed coalitions at level l. We pose the default $\mathfrak{S}_0 = \{S_0\}$ and $S_0 = \{\}$ is the null coalition.

Notations Let X_l be the set that contains all the words at level l, $X_{<l}$ the one that contains all the words before level l in the tree, and $\mathcal{P}(X_l)$ the powerset, i.e. the set of all subsets of X_l.

Definition (Set of coalitions at level l) The set of coalitions $\mathfrak{S}_l$ at level l is defined as:

$$\mathfrak{S}_l = \bigcup_{\sigma \in \mathcal{P}(X_l)} X_{<l} \cup \sigma$$

Property At each level of the tree, each coalition $S \in \mathfrak{S}_l$ respects two properties:

$$\forall i \in [1, n] \text{ s.t. } l_i > l, x_i \notin S. \quad (1)$$

$$\forall i \in [1, n] \text{ s.t. } l_i < l, x_i \in S. \quad (2)$$

Given the tree-based coalitions, we can compute the contribution of each token in the input sentence to the model's prediction. The contribution of feature x_i at level l_i on the dependency tree to the model output $\hat{y}$ is defined as:

$$\phi_i = \frac{1}{N_i} \sum_{S \in \left(\bigcup_{p=0}^{l_i-1} \mathfrak{S}_p\right) \cup \mathfrak{S}_{l_i}^{\setminus i}} [f_{\hat{y}}(S \cup \{x_i\}) - f_{\hat{y}}(S)] \quad (3)$$

where N_i corresponds to the number of allowed coalitions at level l_i that do not contain feature x_i, and $\mathfrak{S}_l^{\setminus i}$ corresponds to the set of coalitions at level l that exclude word x_i, i.e.,

$$\mathfrak{S}_l^{\setminus i} = \bigcup_{\sigma \in \mathcal{P}(X_l)} X_{<l} \cup (\sigma \setminus \{x_i\}).$$

Property Given the number n_l of words (or nodes) at level l of the tree, each word at the same level shares the same number of updates, i.e., allowed coalitions, i.e., $\forall x_i$ s.t. $l_i = l$, $N_i = N^l$ and N_l can be expressed as:

$$N_l = \sum_{p=0}^{l-1} 2^{n_p} + 2^{n_l - 1} - l \quad (4)$$

Proof To compute N_l in equation 4, we proceed recursively starting from the root nodes. The dependency has L levels starting from level $l = 1$. We postulate a hypothetical level 0 where the null coalition $S_0 = \{\}$ can be formed. At level 1, there is the root node of the tree, i.e. $n_1 = 1$. The number of coalitions is $|\mathfrak{S}_1 = \{\{x_{\text{root}}\}\}| = 1$. Let n_l be the number of nodes at level l. The number of combinations of n_l features is 2^{n_l}. Since we already counted the null coalitions at the hypothetical level

0, we don't count it in the allowed coalitions $\mathfrak{S}_l$ at level l. We arrive at the final number of coalitions $|\mathfrak{S}_l| = 2^{n_l} - 1$. Now, let's say we have a word x at level l. This word can join all allowed coalitions at level $< l$ — there are $1 + \sum_{p=1}^{l-1}(2^{n_p} - 1)$ — and all the coalitions of the words at level l where x does not appear — there are $2^{n_l - 1} - 1$. In conclusion, we find that the number of allowed coalitions for word x at level l is:

$$N_l = 1 + \sum_{p=1}^{l-1}(2^{n_p} - 1) + 2^{n_l - 1} - 1$$
$$= \sum_{p=0}^{l-1} 2^{n_p} + 2^{n_l - 1} - l$$

We pose $n_0 = 0$, the number of nodes on the hypothetical level 0, to start the sum at $p = 0$ for simplification.

Our strategy of building tree-based coalitions drops the efficiency assumption of Shapley values but preserves the symmetry axioms for the words at the same level of the dependency tree, as well as the nullity and additivity axioms. Appendix B.1 details the four shapley axioms and discusses which ones SyntaxShap respects or violates. Note that this does not undermine the quality of the explanations since the axioms were shown to work against the goals of feature selection in some cases (Fryer et al., 2021).

3.4 Weighted SyntaxShap

In the context of text data and syntactic dependencies, we assume that words at the top of the tree should be given more importance since they are the syntactic foundations of the sentence and usually correspond to the verb, subject, and verb qualifiers. Therefore, we propose SyntaxShap-W, a variant of our method that weighs words according to their position in the tree. The weights are tree-level-dependent and correspond to the inverse of the word level for which contribution is computed, i.e., $w_l = 1/l$. The contribution of a word x_i at level l_i can be expressed as:

$$\phi_i = \frac{w_{l_i}}{N_i} \sum_{S \in \left(\bigcup_{p=0}^{l_i - 1} \mathfrak{S}_p \right) \cup \mathfrak{S}_{l_i}^{\setminus i}} [f_{\hat{y}}(S \cup \{x_i\}) - f_{\hat{y}}(S)] \quad (5)$$

4 Evaluation

This section describes our model-based evaluation procedure that encompasses both quantitative and qualitative analysis of the explanations. While previous works only focus on the faithfulness of explanations to assess their quality, we also propose to consider human qualitative expectations.

4.1 Quantitative evaluation

To analyze if the explanations are faithful to the model, we adopt *fidelity* the most common model-based metric in xAI (Carvalho et al., 2019), which looks at the top-1 prediction and propose two new variants that balance the LM's probabilistic nature by considering the top-K predictions.

Fidelity Fidelity measures how much the explanation is faithful to the model's initial prediction for the next token. By keeping the top t% words in the input sentence, fidelity calculates the average change in the prediction probability on the predicted word over all test data as follows,

$$\text{Fid}(t) = \frac{1}{N} \sum_{i=1}^{N} (f_{\hat{y}}(x_i) - f_{\hat{y}}(\tilde{x}_i^{(t)})) \quad (6)$$

where $\tilde{x}_i^{(t)}$ is the masked input sentence constructed by keeping the $t\%$ top-scored words of x_i, $\hat{y}$ is the predicted token given input x_i, i.e. $\hat{y} = \text{argmax}_{y'} f_{y'}(x_i)$, and N is the number of examples. Usually, the missing words are replaced by the null token, but we also propose an alternative fidelity Fid_{rand} by replacing the missing words with random words from the tokenizer vocabulary.

Probability divergence@K The probability divergence at K corresponds to the average difference in the top K prediction probabilities on the predicted class over all test data. It can be expressed as follows,

$$\text{div@}K = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{K} (f_{\hat{y}_k}(x_i) - f_{\hat{y}_k}(\tilde{x_i}^{(t)})) \quad (7)$$

where $\hat{y}_k$ is the top k^{th} prediction given input x_i. We choose $K = 10$ because most of the sentences can be completed with multiple possible words that are synonyms or semantically consistent with the input sentence.

Accuracy@K The accuracy at K corresponds to the average ratio of common top K predictions between the full and masked sentences:

$$\text{acc@}K = \frac{1}{N} \sum_{i=1}^{N} \frac{\left| \{\hat{y}_k, k \leq K\} \cap \{\tilde{y}_k^{(t)}, k \leq K\} \right|}{K} \quad (8)$$

where $\tilde{y}_k^{(t)}$ is the top k^{th} prediction given input $\tilde{x}_i^{(t)}$.

4.2 Qualitative evaluation

Coherency Coherency describes how similar the explanation is w.r.t. similar next generated token. In other words, given a pair of input sentences with a slight variation in the syntax but a strong change in semantics (e.g., differing only by a negation), we expect similar explanations for similar model's predictions and dissimilar ones when the model is sensitive to the perturbation.

Semantic alignment An important criterion to evaluate a textual explanation is whether it is aligned with human expectations. As humans, we intuitively expect the language model to draw little attention to tokens in the input sentence which semantic substance is not reflected in the prediction. This semantic alignment can be measured for some semantically rich tokens that are decisive for text generation, e.g., the negation. Given a decisive token in input sentences and a model's prediction that does not semantically account for it, we compare methods on the importance rank attributed to this token. An explainability method is semantically aligned if this rank is high, i.e., the decisive token is not important for the model's prediction.

5 Experiments

We evaluate SyntaxShap and SyntaxShap-W on various criteria such as faithfulness and their computational complexity in section 5.2, the coherency in section 5.3, and the semantic alignment of their explanations in section 5.4.

5.1 Experimental setting

For the evaluation, we use three datasets, i.e., the *Generics KB*[2] (*Generics*) (hug, 2020), *ROCStories Winter2017*[3] (*ROCStories*) (Mostafazadeh et al., 2017), and *Inconsistent Dataset Negation*[2] (*Negation*) (Kalouli et al., 2022). They have the following characteristics: (1) The *Generics* dataset contains high-quality, semantically complete statements; (2) The *ROCStories* dataset contains a collection of five-sentence everyday life stories; (3) The *Negation* dataset contains disjoint sentence pairs, i.e., a sentence and its negated version. For evaluation purposes, we first separate the stories of the *ROCStories* dataset into single sentences and remove the last token from sentences in the three datasets. We use the TextDescriptives component in `spaCy` to measure the dependency distance of the analyzed sentences following the universal dependency relations established by de Marneffe et al. and compute the average number of tokens per sentence as well as the number of unique tokens in the three datasets. As shown in Table 1, sentences in the *Generics* and *ROCStories* datasets have more complex syntactic structures, and the sentences are longer than in the *Negation* dataset. We decide not to include the *Negation* dataset in the quantitative analysis because it becomes difficult to compare explainability approaches when a small number of tokens – less than six – are removed from short phrases without disrupting the sentence's overall meaning. Nevertheless, it is the most suited dataset to compare xAI methods on coherency and semantic alignment since it contains sentences with little syntactic variations but great semantic ones, enabling fine-grained qualitative analysis.

	Generics	ROCStories	Negation
Depd. Dist. μ	1.96	2.12	1.4
Depd. Dist. σ	0.46	0.47	0.30
# Tokens Mean	9.80	9.83	5.54
# Unique Tokens	3548	2082	99

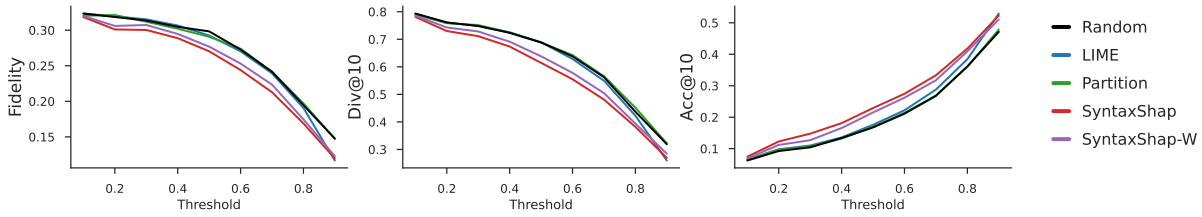
Table 1: Dataset descriptives.

To assess the performance of our method, we use two autoregressive LMs: GPT-2 model (Radford et al., 2019) consisting of 117M parameters and Mistral 7B (Jiang et al., 2023) with 7B parameters. We reproduce our experiments on four different seeds and convey mean and variance of our results. Our methods SyntaxShap and SyntaxShap-W are compared against the *Random* baseline, and two other explainability baselines *LIME* (Ribeiro et al., 2016) and the *NaiveShap*, a naive SHAP-based approach that computes all coalitions, adapted for the problem of next token generation and text data. We also compare them against *Partition*, a faster version of KernelSHAP that hierarchically clusters features. Its implementation is based on HEDGE (Chen et al., 2020), a SHAP-based method that builds hierarchical explanations via divisive generation, respecting some pre-computed word clustering, and is particularly suited for text data.

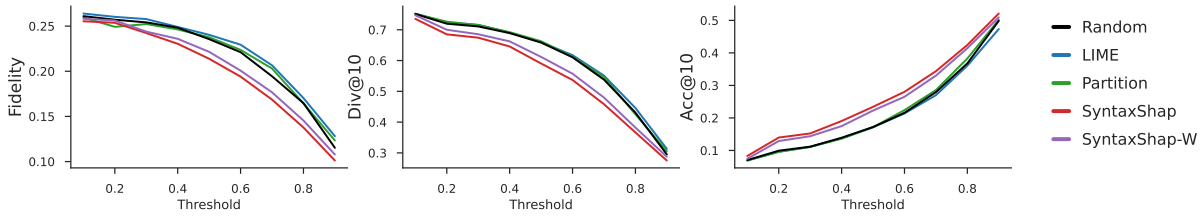
5.2 Faithfulness

In this section, we evaluate the faithfulness of our explanations to Random, LIME, the NaiveShap, and Partition on the full datasets with sentence lengths between 5 and 20 tokens. NaiveShap is

[2]published at the ACL Anthology, CC BY 4.0 License
[3]publicly available, no license



(a) Mistral 7B, Generics dataset.



(b) Mistral 7B, ROCStories dataset.

Figure 2: Faithfulness of the explanations of Mistral 7B predictions by the methods Random, LIME, Partition, and our methods SyntaxShap and SyntaxShap-W. The scores were averaged on 858 instances for Generics dataset and 1046 instances for ROCStories. NaiveShap scores were not displayed here as they are limited to sentences with less than 10 tokens (See Appendix C).

	Generics	**ROCStories**
Random	0.687±0.005	0.657±0.002
LIME	0.686±0.003	0.663±0.004
Partition	0.687±0.005	0.661±0.007
SyntaxShap	**0.615**±0.007	**0.590**±0.005
SyntaxShap-W	0.638±0.002	0.609±0.005

(a) Mistral 7B model.

	Generics	**ROCStories**
Random	0.578±0.003	0.576±0.002
LIME	0.579±0.007	0.587±0.002
NaiveShap	0.518±0.002	0.517±0.002
Partition	0.550±0.004	0.542±0.006
SyntaxShap	**0.512**±0.002	**0.497**±0.002
SyntaxShap-W	0.556±0.007	0.536±0.003

(b) GPT-2 model.

Table 2: The div@10 scores of explainability methods for the Mistral 7B model. Explanations are sparse at threshold $t = 0.5$, i.e. we keep 50% of the top words. We report the mean and variance after running experiments on four different random seeds. The methods introduced in this paper are shaded.

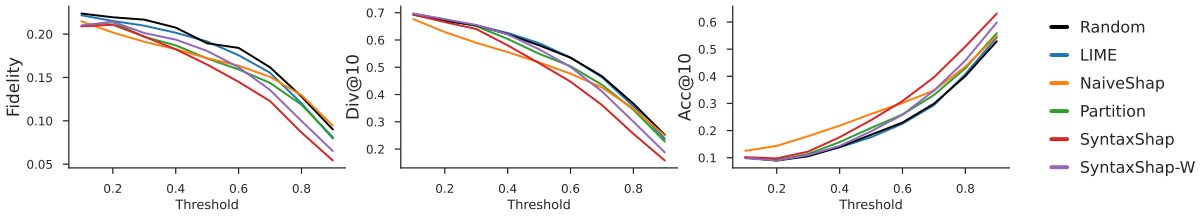
omitted for Mistral 7B since computations become intractable for sentences with $>$ 10 tokens. See Appendix D.1 for the comparison with NaiveShap on the filtered datasets.

For both models, Mistral 7B and GPT-2 in Figure 2 and Figure 3, our methods SyntaxShap and SyntaxShap-W produce more faithful explanations than the trivial random algorithm, the LIME method adapted to NLP tasks, and Partition, the state-of-the-art shapley-based local explainability method for text data. Therefore, building coalitions based on syntactic rules gives more faithful explanations than when minimizing a cohesion score, preserving the strongest word interactions. For GPT-2 model in Figure 3, NaiveShap generates explanations as faithful as SyntaxShap. However, SyntaxShap has the advantage of being much faster, with a computational complexity of $\mathcal{O}(nL2^{n/L})$ against $\mathcal{O}(n2^n)$ for NaiveShap, where n is the number of words in the input sentence and L the tree depth. This is a huge advantage when explaining long sentences with more than 10 tokens or when the LM has a high inference time. We refer to Appendix B.2 for the comparison of NaiveShap and SyntaxShap complexities and to Appendix D.2 where we show how the number of tokens affects NaiveShap computations.

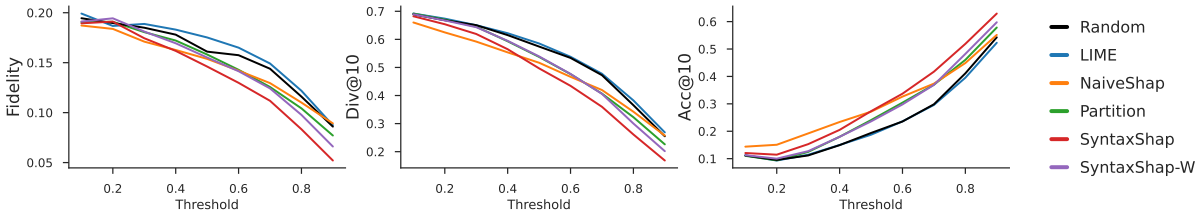
SyntaxShap(-W) generates more faithful explanations than the random baseline, LIME, and Partition. Although it does not beat the NaiveShap method, it can scale to longer sentences and its computation is faster.

5.3 Coherency

In this section, we explore whether SyntaxShap produces coherent explanations with the model understanding. For this evaluation, we use Mistral 7B and run a perturbation analysis using sentence pairs from the *Negation* dataset. We use a sample of 72 sentence pairs (with and without the negation *not* and with varying usage of *with* and *without*) whereby for 20 pairs, the model predicts the same



(a) GPT-2, Generics dataset



(b) GPT-2, ROCStories dataset

Figure 3: Faithfulness of the explanations of GPT-2 predictions by the methods Random, LIME, NaiveShap, Partition, and our methods SyntaxShap and SyntaxShap-W. The scores were averaged on 1434 instances for Generics dataset and 1318 instances for ROCStories.

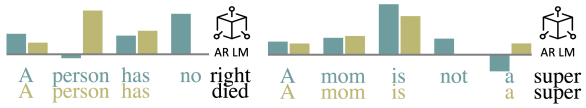


Figure 4: An example of attribution values of the SyntaxShap-W method for two sentence pairs with different and similar next token predictions.

next token. An example of two sentence pairs is shown in Figure 4. For pairs with equal predictions (e.g., *A mom is* ***not*** *a* and *A mom is a* with an equal next token prediction **super**), we expect more similar attribution ranks than for pairs with different predictions (e.g., *A person has* ***no*** **right** and *A person has* **died**). To evaluate the coherency, we

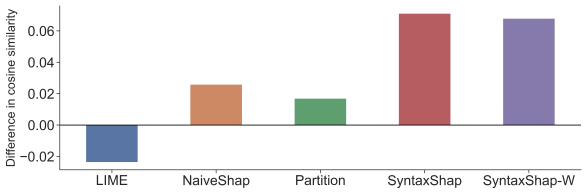


Figure 5: Coherency of explainability methods for the Mistral 7B model on sentence pairs varying by the used negation. SyntaxShap and SyntaxShap-W produce more similar attribution scores for sentence pairs where the model predicts the same next token compared to sentence pairs with different next token predictions.

first represent the attribution scores as rank vectors. We then separate pairs with equal predictions and different predictions into two distinct groups and measure the cosine similarity between rank vectors of each pair within each group, whereby negation words are excluded to get equal-length vectors. The average difference in cosine similarity between the two groups for each explainability method is displayed in Figure 5. It shows that SyntaxShap and SyntaxShap-W produce more similar attributions for sentence pairs that predict the same next token and more diverse attributions for sentence pairs with different next token predictions.

Given a pair of sentences with and without a negation, which theoretically have two disjoint semantic meanings, the similarity of SyntaxShap(-W)'s token attribution values for each sentence better reflect the degree of similarity of the next token predictions than LIME, NaiveShap, and Partition.

5.4 Semantic alignment

We explore here if the generated explanations are aligned with human semantic expectations. To be able to answer this question, we analyze cases where there is a negation in a sentence, but the model's prediction does not reflect it, e.g., *A father is not a* ***father***.To realize this experiment, we extract negative instances, i.e., that contain the token *not*, *no*, or *without*, from the *Negation* dataset. We label those where the model, GPT-2 or Mistral 7B. predicts *wrong* next tokens, i.e., semantically misaligned with the negation. We report the average importance score of the negation tokens in each of the 15 labeled instances. Figure 6 shows the results for Mistral 7B: SyntaxShap and SyntaxShap-W rank the importance of negations as 3rd or greater in 80% of the cases. They give low importance to the negation tokens when the model is not able to

capture them. LIME and the naive computation of Shapley values by NaiveShap assigns 3rd rank or greater for 60% of the negations. Partition is the worst at reflecting the irrelevance of negations, ranking them as 1st or 2nd in 60% of the cases.

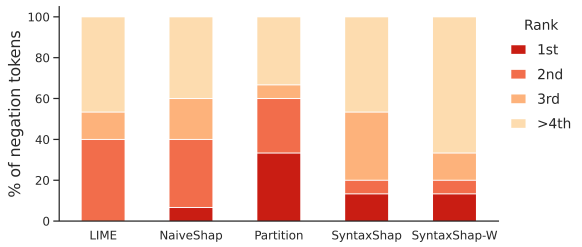


Figure 6: Importance rank distribution of negation tokens *not*, *no*, and *without* when the model does not capture the negations in the predicted next token.

SyntaxShap(-W) assigns lower importance ranks to input tokens which semantic is not captured in the model's prediction compared to LIME, NaiveShap, and Partition.

6 Discussion

Addressing stochasticity The traditional faithfulness metrics like fidelity, AOPC (Samek et al., 2016; Nguyen, 2018) or log-odds (Shrikumar et al., 2017; Chen et al., 2019) scores take a deterministic approach to evaluate explanations computed on stochastic predictions. This paper evaluated autoregressive LMs that adopt top-k sampling to randomly select a token among the k tokens with the highest probability. To account for this stochasticity, we proposed additional evaluation metrics, div@K and acc@K, that consider not only the final prediction but the top-K predictions, balancing the model's probabilistic nature. Nevertheless, further methods that address the stochastic nature of the models should be designed in future research.

Integrating linguistic knowledge To ensure that the explainability methods produce meaningful explanations that mimic autoregressive LM behavior, we need to go beyond the faithfulness type of evaluation and consider further explainability aspects. In this paper, we study explanations on other dimensions related to semantic interpretation and coherency of explanations. There is potential for more linguistically tailored evaluation methods in the future. The motivation is as follows. The next token prediction task can be seen as a multi-class classification with a large number of classes. The classes have diverse linguistic properties, i.e., tokens have different roles in the sentence, some being more content- and others function-related. We might want to consider these different roles when evaluating the quality of explanations. On the one hand, with controlled perturbations on the input sentences, we can evaluate the role of semantics and syntax on the next token prediction task. On the other hand, when computing the explanation fidelity, we might consider prediction changes from one category of tokens (e.g., function words) to another (e.g., content words), which would give us a more linguistic-aware explanation quality assessment.

Considering humans When designing evaluation methods, we need to consider humans since, ideally, they should understand model behavior from the produced explanations. There is one main concern, though. As prior work has shown (Sevastjanova and El-Assady, 2022), LM explainability can suffer from human false rationalization of model behaviors. We typically expect the explanations to align with our mental models of language.However, LMs learn language differently from humans; thus, explanations can theoretically differ from our expectations. Thus, future work should design evaluation methods that clearly show the importance of the words for the model and the reasons why this importance (potentially) does not align with human expectations.

7 Conclusion

We proposed SyntaxShap - a local, model-agnostic syntax-aware explanability method. Our method is specifically tailored for text data and meant to explain text generation tasks by autoregressive LMs, whose interpretability in that context has not yet been addressed. SyntaxShap is the first SHAP-based method to incorporate the syntax of input sentences by constraining the construction of word coalitions on the dependency trees. Our experimental results demonstrate that SyntaxShap and its weighted variant can improve explanations in many aspects: they generate more faithful, coherent, and semantically rich explanations than the standard model-agnostic explainability methods in NLP. This study addresses a pressing and significant issue regarding the explainability of autoregressive models, contributing to an ongoing dialogue in the research community.

8 Limitations

Long sentences and limited compute power One limitation of this paper is the limited computing power to explain long sentences for certain methods and models. For example, to run the naive implementation of Shapley values that has a complexity of $\mathcal{O}(n2^n)$ and a number n of tokens between 3 and 20, some sentences require up to 21 million computation steps! Given our Linux machine with 2 GPUs NVIDIA RTX A6000 with 4 GB RAM per CPU, computations for sentences with more than 10 tokens for the Mistral 7B model were intractable. We could only include NaiveShap results for sentences with less than 10 tokens (see Appendix D.1). As the length of the sentences increases, the computation complexity of SyntaxShap does, too, and we might reach the same limitation as NaiveShap. In addition, we limit our analysis to one input sentence because we work on one dependency tree at a time. However, our method can be scaled to text with multiple sentences or a paragraph by breaking it down into multiple dependency trees and running SyntaxShap in parallel. However, by doing this, we might lose sentence correlations.

Incorrect dependency tree Our method heavily relies on the dependency tree, assuming it correctly captures the syntactic relationships between the words. However, the Python module `spaCy` sometimes generates arguable dependencies from the perspective of linguists, and its accuracy drops when implemented for languages other than English. Therefore, SyntaxShap is, for now, only meant to be used for English grammatically non-convoluted sentences to limit the uncertainty coming from the construction of the dependency tree itself.

Tokenization and word segmentation An important limitation is related to the tokenization of the sentences. The tokenization might break down words into multiple tokens. However, SyntaxShap computation is based on a dependency tree in which nodes must be words. Here, we have not addressed this problem. We choose to only include the ones where no word is split by the tokenizer (see Appendix C). For future work, we suggest modifying the tree parsing to allow for duplicated nodes. This token dependency tree will have tokens of the same word as separate nodes in this tree with similar roles.

9 Ethics Statement

The data and resources utilized in this study are openly available and widely used by numerous existing works. The datasets employed consist of factual statements devoid of subjective judgments or opinions. It is acknowledged that pre-trained LMs, such as GPT-2 and Mistral 7B, may inherently inherit biases as highlighted in previous research (Radford et al., 2019), potentially influencing the generated next token. For example, certain tokens like *beautiful* may tend to appear more frequently in contexts associated with female characteristics. While the primary objective of this study is to produce explanations that faithfully represent the model's predictions, it is recognized that these explanations may also carry inherent biases. It is imperative to acknowledge that the results generated by our approach may not always align with human mental models and could potentially be used in applications that have the potential to cause harm.

References

2020. Genericskb: A knowledge base of generic statements. Allen Institute for AI.

Vamshi Ambati. 2008. Dependency structure trees in syntax based machine translation. In *Adv. MT Seminar Course Report*, volume 137.

Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. 2019. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832.

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

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. 2019. L-shapley and c-shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations*.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal stanford dependencies: A cross-linguistic typology. `http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf`. Accessed: 2023-11-9.

Tiwalayo Eisape, Vineet Gangireddy, Roger Levy, and Yoon Kim. 2022. Probing for incremental parse states in autoregressive language models. In *Findings of the Association for Computational Linguistics:*

EMNLP 2022, pages 2801–2813, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Adam Ek, Jean-Philippe Bernardy, and Shalom Lappin. 2019. Language modeling with syntactic and semantic representation for sentence acceptability predictions. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 76–85, Turku, Finland. Linköping University Electronic Press.

Christopher Frye, Colin Rowat, and Ilya Feige. 2020. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Daniel Fryer, Inga Strümke, and Hien Nguyen. 2021. Shapley values for feature selection: The good, the bad, and the axioms.

Tom Heskes, Ioan Gabriel Bucur, Evi Sijben, and Tom Claassen. 2020. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Aikaterini-Lida Kalouli, Rita Sevastjanova, Christin Beck, and Maribel Romero. 2022. Negation, coordination, and quantifiers in contextualized language models. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3074–3085, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Enja Kokalj, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. 2021. BERT meets shapley: Extending SHAP explanations to transformer-based classifiers. In *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*, pages 16–21, Online. Association for Computational Linguistics.

Edoardo Mosca, Ferenc Szigeti, Stella Tragianni, Daniel Gallagher, and Georg Groh. 2022. SHAP-Based explanation methods: A review for NLP interpretability. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4593–4603, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain. Association for Computational Linguistics.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673.

Rita Sevastjanova and Mennatallah El-Assady. 2022. Beware the Rationalization Trap! When Language Model Explainability Diverges from our Mental Models of Language. In *Communication in Human-AI Interaction Workshop at IJCAI-ECAI'22*.

Lloyd S Shapley et al. 1953. *A value for n-person games*. Princeton University Press Princeton.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

Thinh Hung Truong, Timothy Baldwin, Karin Verspoor, and Trevor Cohn. 2023. Language models are not naysayers: an analysis of language models on negation benchmarks. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 101–114, Toronto, Canada. Association for Computational Linguistics.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2023. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*.

A Textual data

A.1 Text generation

Text generation tasks involve predicting the next word in a sequence, like in language modeling, which can be considered a simpler form of text generation. Other tasks may involve generating entire paragraphs or documents. Text generation can also be framed as a sequence-to-sequence (seq2seq) task that aims to take an input sequence and generate an output sequence for machine translation and question answering. Autoregressive models like GPT (Generative Pre-trained Transformer) generate text one word at a time in an autoregressive manner, conditioning each word on the previously generated words. In this paper, we focused on the next token generation task given one single sentence as input. We work with factual sentences from *Generics* and *ROCStories* datasets, which often expect a semantically rich final token to complete the clause. Multiple predictions are possible, but only a few are correct. Here is an example of a sentence in the *Generics* dataset: *Studio executive is an employee of a film*. The GPT-2 model predicts *studio* as the next token with the random seed 0. We can expect other predictions like *company*, *firm*, or *corporation*. But the number of possibilities is still very limited.

A.2 Dependency parsing

Dependency parsing is a natural language processing technique that involves analyzing the grammatical structure of a sentence to identify the relationships between words (de Marneffe et al.). It involves constructing a tree-like structure of dependencies, where each word is represented as a node, and the relationships between words are represented as edges. Each relationship has one head and a dependent that modifies the head, and it is labeled according to the nature of the dependency between the head and the dependent. These labels can be found at Universal Dependency Relations (de Marneffe et al.). Dependency Parsing is a powerful technique for understanding the meaning and structure of language and is used in various applications, including text classification, sentiment analysis, and machine translation. We use the Python module `spaCy` (version 3.7.2) (Honnibal and Montani, 2017) to generate dependency trees on the input sentences. The number of tokens varies from 5 to 20 tokens for the *Generics* and *ROCStories* datasets, producing very diverse and complex parsing trees. This diversity enriches our analysis and strengthens our results. The code source for dependency parsing was extracted from `https://stackoverflow.com/questions/7443330/how-do-i-do-dependency-parsing-in-nltk`.

B SyntaxShap: characteristics and proofs

B.1 SyntaxShap and the Shapley axioms

The four axioms satisfied by Shapley values, i.e., efficiency, additivity, nullity, and symmetry, do not generally provide any guarantee that the computed contribution value is suited to feature selection, and may, in some cases, imply the opposite (Fryer et al., 2021). We define here new axioms for SyntaxShap values since two of the four Shapley axioms cannot be satisfied by tree-constraint values.

Efficiency The evaluation function $v(S)$ in SyntaxShap is the output probability for the predicted next token given the full input sentence, i.e., $v(S) = f_{\hat{y}}(S)$ where $\hat{y} = \text{argmax}(f(x))$. Because of the non-linearity of LMs, SyntaxShap evaluation function is non-monotonic. It does not necessarily increase if you add more features. For this reason, SyntaxShap does *not* satisfy the *efficiency* axiom. This implies that the SyntaxShap values of each word do not sum up to the SyntaxShap value of the whole sentence.

Symmetry SyntaxShap satisfies the axiom of *symmetry* at each level of the dependency tree. Any two features x_i, x_j that are at the same level of the dependency tree, i.e., $l_i = l_j$, play equal roles and therefore have equal SyntaxShap values:

$$\begin{aligned} &\forall i, j \text{ s.t. } l_i = l_j \\ &[\forall (S \setminus \{x_i, x_j\}) v(S \cup x_i) = v(S \cup x_i)] \\ &\implies \phi_i = \phi_j \end{aligned} \tag{9}$$

Nullity If feature x_i contributes nothing to each submodel it enters, then its SyntaxShap value is zero.

$$[(\forall S) v(S \cup \{x_i\}) = v(S)] \implies \phi_i = 0 \tag{10}$$

Additivity Given two models f and g, the SyntaxShap value of those models is a linear combination of the individual models' SyntaxShap values:

$$\phi_i(f + g) = \phi_i(f) + \phi_i(g) \tag{11}$$

B.2 Computational complexity

One advantage of the SyntaxShap algorithm is its faster computation time compared to the naive Shapley values computations. We estimate the complexity of each algorithm by approximating the total number of computation steps, i.e., formed coalitions and updated values, for the traditional naive SHAP computation and our method.

NaiveShap The Shapley value of feature x requires the 2^{n-1} coalitions of all features excluding x. As we need to update n features, the total number of updates is $n \cdot 2^{n-1}$. The computation complexity is, therefore, in $\mathcal{O}(n2^n)$.

SyntaxShap The SyntaxShap value of feature x at level l requires N_l updates. Considering all the features in the input, the total number of computations is $\sum_{l=1}^{L} n_l \cdot N_l$. To approximate this number, we assume the dependency tree to be balanced and pose $n_l = n/L$. In this case, N_l can be re-written as:

$$\begin{aligned} N_l &= \sum_{p=0}^{l-1} 2^{n/L} + 2^{n/L-1} - l \\ &= l(2^{n/L} - 1) + 2^{n/L-1} \end{aligned}$$

The total number of computations can now be approximated to:

$$\begin{aligned} \frac{n}{L}\sum_{l=1}^{L} N_l &= \frac{n}{L}\sum_{l=1}^{L}\left(l(2^{n/L} - 1) + 2^{n/L-1}\right) \\ &= \frac{n}{L}\left(\frac{L(L+1)}{2}(2^{n/L} - 1) + L2^{n/L-1}\right) \\ &= \frac{n(L+1)}{2}(2^{n/L} - 1) + n2^{n/L-1} \end{aligned}$$

The approximation of the computation complexity in the case of a balanced tree is $\mathcal{O}(nL2^{n/L})$.

C Data preprocessing

SyntaxShap relies on the construction of dependency trees that capture the syntactic dependencies in the sentences. Entities in dependency trees are words as defined in the English dictionary. However, language tokenizers sometimes split words so that the tokenizer vocabulary does not necessarily contain the English one. To account for the disagreement between tokenization and parsing, we filter out the sentences that contain words that do not belong to the tokenizer's vocabulary and might be split into multiple tokens by the tokenizer. Table 3 displays the statistics for the three datasets *Negation*, *Generics*, and *ROCStories*, with the initial number of sentences and the explained sentences after filtering. Our filtering strategy consists of keeping only sentences that do not contain punctuations `!"#$%&'()*+, -./:;<=>?@[\]^_`{}~` given by the Python module `string` and where the tokens excluding prefix and suffix tokens, correspond to the words.

	Negation	Generics	ROCStories
Initial size	534	5777	2275
GPT-2 filter	366	1434	1318
Mistral filter	332	858	1046

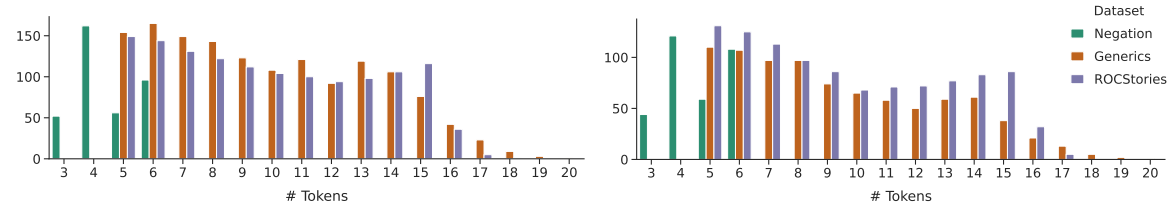
Table 3: Dataset statistics.

Figure 7 displays the length distribution of sentences in each dataset after filtering. *Negation* dataset contains short sentences with less than 6 tokens. It is used in our study for experiments on coherency and semantic alignment of explanations. *Generics* and *ROCStories* are more complex and realistic. The majority of their input sentences have between 5 and 15 tokens, with a few exceptions of longer sentences. They also have a greater diversity of words and syntactic complexity as identified by the dependency distance and token diversity in Table 1.

D Additional results

D.1 Mistral 7B on the full datasets

To compare NaiveShap method with SyntaxShap with Mistral 7B, we have to filter out long sentences for which the computation is tractable with our compute power. Table 4 presents the new statistics of *Generics* and *ROCStories* datasets. We keep approximately 60% of the input sentences for both datasets. Figure 8 displays the performance of SyntaxShap and SyntaxShap-W and the baselines Random, LIME, NaiveShap and Partition. We evaluate them on the three faithfulness metrics like in section 5.2. NaiveShap shows similar performance on the div@10 and acc@10 scores as our methods SyntaxShap and SyntaxShap-W. We only notice a score difference for the fidelity metric, where NaiveShap generates less faithful explanations for the ROCStories dataset if you only consider the top-1 model prediction. These observations consolidate the conclusions drawn in section 5.2, namely that both our method and NaiveShap produce better



(a) Filtering with GPT-2 model.

(b) Filtering with Mistral 7B model.

Figure 7: Number of tokens distribution for the three datasets *Inconsistent Negation Dataset* (Negation), *Generics KB* (Generics) and *ROCStories Winter2017* (ROCStories).

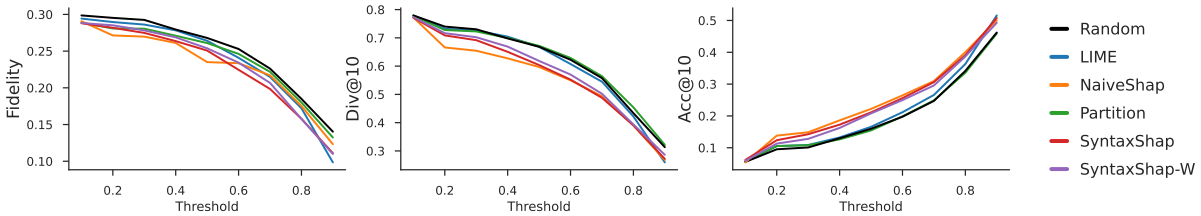
explanations on the faithfulness dimension than the other three methods.

	Generics	**ROCStories**
Without NaiveShap	858	1046
With NaiveShap	512	608

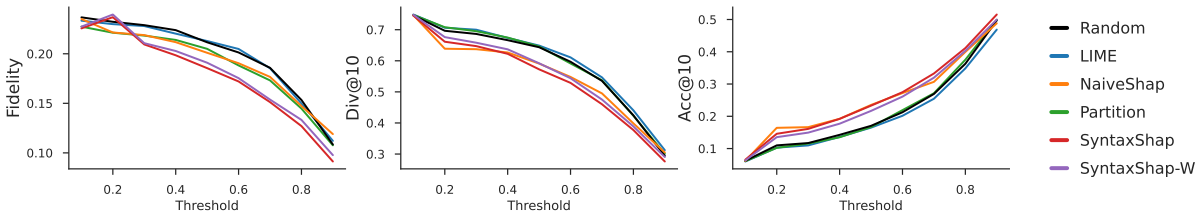
Table 4: Dataset statistics before and after keeping the input sentences for which NaiveShap is computationally tractable. The statistics are for Mistral 7B filtering.

D.2 Number of tokens and faithfulness

This section analyzes the relationship between the number of tokens in the input sentences and the performance of the explainability algorithms. We vary the number of tokens from 5 to 15 tokens to have at least 50 sentences of the same length for both *Generics* and *ROCStories* and have a decent number of inputs to average upon (see the number of tokens distribution in Figure 7). Figure 9 and 10 show that the performance of all methods is robust to the increase in the number of tokens. SyntaxShap can be applied to a diverse range of sentence lengths. Note that for Mistral 7B in Figure 10 the results of NaiveShap are limited to sentences with less than 10 tokens because of NaiveShap intractable computations with our compute power (see section 8).



(a) Mistral 7B, Generics dataset.



(b) Mistral 7B, ROCStories dataset.

Figure 8: Faithfulness of the explanations of Mistral 7B predictions by the methods Random, LIME, NaiveShap, Partition, and our methods SyntaxShap and SyntaxShap-W, the weighted variant. Long sentences were removed because of NaiveShap's intractable computation time. The scores were averaged on 512 instances for Generics dataset and 608 instances for ROCStories dataset.

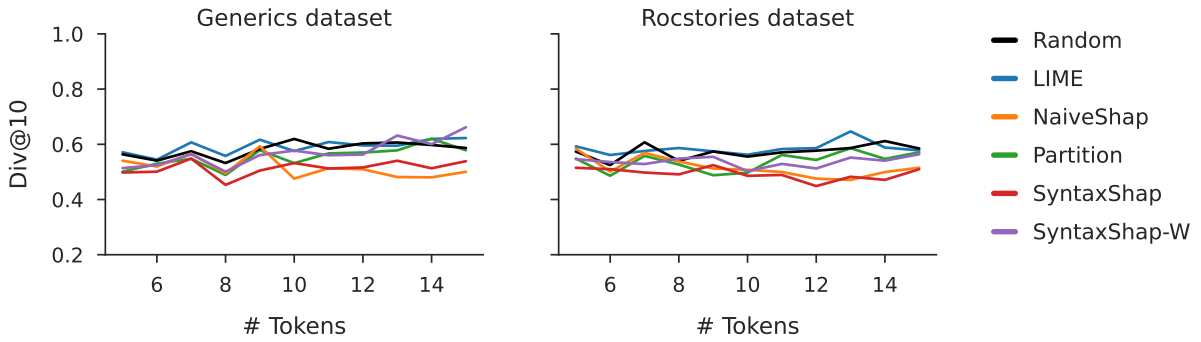


Figure 9: Performance of the explainers for GPT-2 model when varying the number of tokens from 5 to 15.

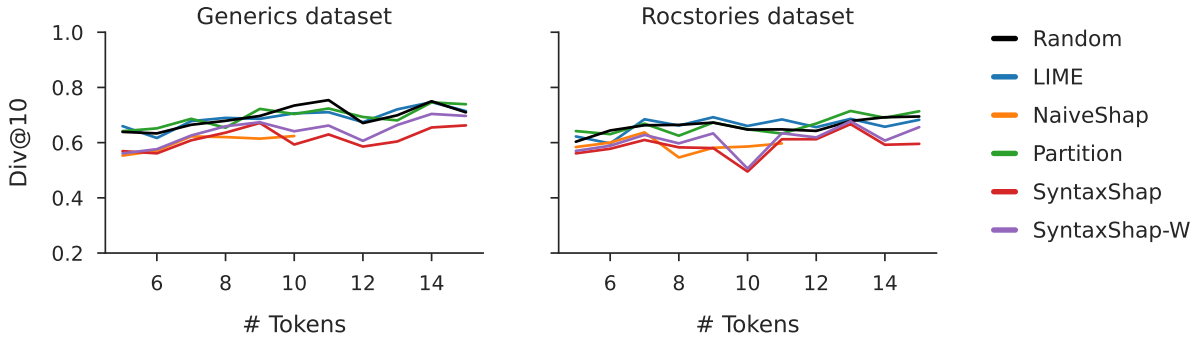


Figure 10: Performance of the explainers for Mistral 7B model when varying the number of tokens from 5 to 15. Naiveshap div@10 scores were only computed for sentences with less than 10 tokens.