An Analysis of LIME for Text Data

Dina Mardaoui
Polytech Nice, France

Damien Garreau
Université Côte d'Azur, Inria, CNRS, LJAD, France

Abstract

Text data are increasingly handled in an automated fashion by machine learning algorithms. But the models handling these data are not always well-understood due to their complexity and are more and more often referred to as "black-boxes." Interpretability methods aim to explain how these models operate. Among them, LIME has become one of the most popular in recent years. However, it comes without theoretical guarantees: even for simple models, we are not sure that LIME behaves accurately. In this paper, we provide a first theoretical analysis of LIME for text data. As a consequence of our theoretical findings, we show that LIME indeed provides meaningful explanations for simple models, namely decision trees and linear models.

1 Introduction

Natural language processing has progressed at an accelerated pace in the last decade. This time period saw the second coming of artificial neural networks, embodied by the apparition of recurrent neural networks (RNNs) and more particularly long short-term memory networks (LSTMs). These new architectures, in conjunction with large, publicly available datasets and efficient optimization techniques, have allowed computers to compete with and sometime even beat humans on specific tasks.

More recently, the paradigm has shifted from recurrent neural networks to *transformers networks* (Vaswani et al., 2017). Instead of training models specifically for a task, large *language models* are trained on supersized datasets. For instance, `Webtext2` contains the text data associated to 45 millions links (Radford et al., 2019). The growth in complexity of these models seems to know no limit, especially with regards to their number of parameters. For instance, BERT (Devlin et al., 2018) has roughly 340 millions of parameters, a meager number compared to more recent models such as GTP-2 (Radford et al., 2019, 1.5 billions) and GPT-3 (Brown et al., 2020, 175 billions).

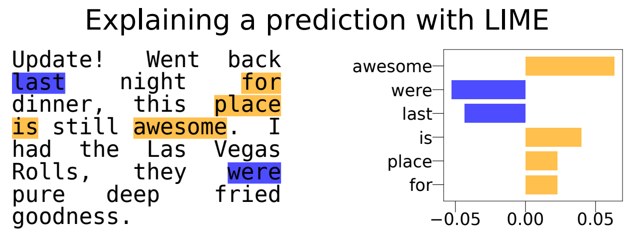

Figure 1: Explaining the prediction of a random forest classifier on a Yelp review. *Left panel:* the document to explain. The words deemed important for the prediction are highlighted, in orange (positive influence) and blue (negative influence). *Right panel:* values of the largest 6 interpretable coefficients, ranked by absolute value.

Faced with such giants, it is becoming more and more challenging to understand how particular predictions are made. Yet, *interpretability* of these algorithms is an urgent need. This is especially true in some applications such as healthcare, where natural language processing is used for instance to obtain summaries of patients records (Spyns, 1996). In such cases, we do not want to deploy in the wild an algorithm making near perfect predictions on the test set but for the wrong reasons: the consequences could be tragic.

In this context, a flourishing literature proposing *interpretability methods* emerged. We refer to the survey papers of Guidotti et al. (2018) and Adadi and Berrada (2018) for an overview, and to Danilevsky et al. (2020) for a focus on natural language processing. With the notable exception of SHAP (Lundberg and Lee, 2017), these methods do not come with any guarantees. Namely, given a simple model already interpretable to some extent, we cannot be sure that these methods provide meaningful explanations. For

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

instance, explaining a model that is based on the presence of a given word should return an explanation that gives high weight to this word. Without such guarantees, using these methods on the tremendously more complex models aforementioned seems like a risky bet.

In this paper, we focus on one of the most popular interpretability method: *Local Interpretable Model-agnostic Explanations* Ribeiro et al. (2016, LIME), and more precisely its implementation for text data. LIME's process to explain the prediction of a model f for an example ξ can be summarized as follows:

(i). from a corpus of documents $\mathcal{C}$, create a TF-IDF transformer ϕ embedding documents into $\mathbb{R}^D$;

(ii). create n perturbed documents $x_1, \ldots, x_n$ by deleting words at random in ξ;

(iii). for each new example, get the prediction of the model $y_i := f(\phi(x_i))$;

(iv). train a (weighted) linear surrogate model with inputs the absence / presence of words and responses the y_is.

The user is then given the coefficients of the surrogate model (or rather a subset of the coefficients, corresponding to the largest ones) as depicted in Figure 1. We call these coefficients the *interpretable coefficients.*

The model-agnostic approach of LIME has contributed greatly to its popularity: one does not need to know the precise architecture of f in order to get explanations, it is sufficient to be able to query f a large number of times. The explanations provided by the user are also very intuitive, making it easy to check that a model is behaving in the appropriate way (or not!) on a particular example.

Contributions. In this paper, we present the first theoretical analysis of LIME for text data. In detail,

- we show that, when the number of perturbed samples is large, **the interpretable coefficients concentrate with high probability around a fixed vector** β that depends only on the model, the example to explain, and hyperparameters of the method;
- we provide an **explicit expression of** β, from which we gain interesting insights on LIME. In particular, **the explanations provided are linear in** f;
- for simple decision trees, we go further into the computations. We show that **LIME provably provides meaningful explanations**, giving large coefficients to words that are pivotal for the prediction;
- for linear models, we come to the same conclusion by showing that the interpretable coefficient associate to a given word is approximately equal to **the product of the coefficient in the linear model and the TF-IDF transform of the word** in the example.

We want to emphasize that all our results apply to the default implementation of LIME for text data[1] (as of October 12, 2020), with the only caveat that we do not consider any feature selection procedure in our analysis. All our theoretical claims are supported by numerical experiments, the code thereof can be found at `https://github.com/dmardaoui/lime_text_theory`.

Related work. The closest related work to the present paper is Garreau and von Luxburg (2020a), in which the authors provided a theoretical analysis of a variant of LIME in the case of tabular data (that is, unstructured data belonging to $\mathbb{R}^N$) when f is linear. This line of work was later extended by the same authors (Garreau and von Luxburg, 2020b), this time in a setting very close to the default implementation and for other classes of models (in particular partition-based classifiers such as CART trees and kernel regressors built on the Gaussian kernel). While uncovering a number of good properties of LIME, these analyses also exposed some weaknesses of LIME, notably cancellation of interpretable features for some choices of hyperparameters.

The present work is quite similar in spirit, however we are concerned with *text data.* The LIME algorithm operates quite differently in this case. In particular, the input data goes first through a TF-IDF transform (a non-linear transformation) and there is no discretization step since interpretable features are readily available (the words of the document). Therefore both the analysis and our conclusions are quite different, as it will become clear in the rest of the paper.

2 LIME for text data

In this section, we lay out the general operation of LIME for text data and introduce our notation in the process. From now on, we consider a model f and look at its prediction for a fixed example ξ belonging to a corpus $\mathcal{C}$ of size N, which is built on a dictionary $\mathcal{D}$ of size D. We let $\|\cdot\|$ denote the Euclidean norm, and S^{D-1} the unit sphere of $\mathbb{R}^D$.

Before getting started, let us note that LIME is usually used in the *classification* setting: f takes values in $\{0, 1\}$ (say), and $f(\phi(\xi))$ represents the class at-

[1] `https://github.com/marcotcr/lime`

tributed to ξ by f. However, behind the scenes, LIME requires f to be a real-valued function. In the case of classification, this function is the probability of belonging to a certain class according to the model. In other words, the *regression* version of LIME is used, and this is the setting that we consider in this paper. We now detail each step of the algorithm.

2.1 TF-IDF transform

LIME works with a vector representation of the documents. The TF-IDF transform (Luhn, 1957; Jones, 1972) is a popular way to obtain such a representation. The idea underlying the TF-IDF is quite simple: to any document, associate a vector of size D. If we set $w_1, \ldots, w_D$ to be our dictionary, the jth component of this vector represents the importance of word w_j. It is given by the product of two terms: the term frequency (TF, how frequent the word is in the document), and the inverse term frequency (IDF, how rare the word is in our corpus). Intuitively, the TF-IDF of a document has a high value for a given word if this word is frequent in the document and, at the same time, not so frequent in the corpus. In this way, common words such as “the” do not receive high weight.

Formally, let us fix $\delta \in \mathcal{C}$. For each word $w_j \in \mathcal{D}$, we set m_j the number of times w_j appears in δ. We also set $v_j := \log \frac{N+1}{N_j+1} + 1$, where N_j is the number of documents in $\mathcal{C}$ containing w_j. When presented with $\mathcal{C}$, we can pre-compute all the v_js and at run time we only need to count the number of occurrences of w_j in δ. We can now define the normalized TF-IDF:

Definition 1 (Normalized TF-IDF). We define the *normalized TF-IDF* of δ as the vector $\phi(\delta) \in \mathbb{R}^D$ defined coordinate-wise by

$$\forall 1 \leq j \leq D, \quad \phi(\delta)_j := \frac{m_j v_j}{\sqrt{\sum_{j=1}^D m_j^2 v_j^2}} . \qquad (1)$$

In particular, $\|\phi(\delta)\| = 1$, where $\|\cdot\|$ is the Euclidean norm.

Note that there are many different ways to define the TF and IDF terms, as well as normalization choices. We restrict ourselves to the version used in the default implementation of LIME, with the understanding that different implementation choices would not change drastically our analysis. For instance, normalizing by the ℓ_1 norm instead of the ℓ_2 norm would lead to slightly different computations in Proposition 4.

Finally, note that this transformation step does not take place for tabular data, since the data already belong to $\mathbb{R}^D$ in this case.

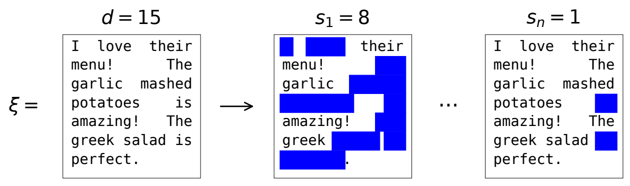
$d = 15$
$s_1 = 8$
$s_n = 1$
$\xi =$
I love their menu! The garlic mashed potatoes is amazing! The greek salad is perfect.
their menu! garlic amazing! greek .
I love their menu! The garlic mashed potatoes amazing! The greek salad perfect.

Figure 2: The sampling scheme of LIME for text data. To the left, the document to explain ξ, which contains $d = 15$ distinct words. The new samples $x_1, \ldots, x_n$ are obtained by removing s_i random words from ξ (in blue). In the nth sample, one word is removed, yielding two deletions in the original document.

2.2 Sampling

Let us now fix a given document ξ and describe the sampling procedure of LIME. Essentially, the idea is to sample new documents similar to ξ in order to see how f varies in a neighborhood of ξ.

More precisely, let us denote by d the number of distinct words in ξ and set $D_\ell := \{w_1, \ldots, w_d\}$ the *local dictionary*. For each new sample, LIME first draws uniformly at random in $\{1, \ldots, d\}$ a number s_i of words to remove from ξ. Subsequently, a subset $S_i \subseteq \{1, \ldots, d\}$ of size s_i is drawn uniformly at random: all the words with indices contained in S_i are *removed* from ξ. Note that the multiplicity of removals is independent from s_i: if the word “good” appears 10 times in ξ and its index belongs to S, then all the instances of “good” are removed from ξ (see Figure 2). This process is repeated n times, yielding n new samples $x_1, \ldots, x_n$. With these new documents come n new binary vectors $z_1, \ldots, z_n \in \{0, 1\}^d$, marking the absence or presence of a word in x_i. Namely, $z_{i,j} = 1$ if w_j belongs to x_i and 0 otherwise. We call the z_is the *interpretable features*. Note that we will write $\mathbf{1} := (1, \ldots, 1)^\top$ for the binary feature associated to ξ: all the words are present.

Already we see a difficulty appearing in our analysis: when removing words from ξ at random, $\phi(\xi)$ is modified in a non-trivial manner. In particular, the denominator of Eq. (1) can change drastically if many words are removed.

In the case of tabular data, the interpretable features are obtained in a completely different fashion, by discretizing the dataset.

2.3 Weights

Let us start by defining the *cosine distance*:

Definition 2 (Cosine distance). For any $u, v \in \mathbb{R}^d$, we define

$$d_{\cos}(u,v) := 1 - \frac{u \cdot v}{\|u\| \cdot \|v\|} \,. \tag{2}$$

Intuitively, the cosine distance between u and v is small if the *angle* between u and v is small. Each new sample x_i receives a positive weight π_i, defined by

$$\pi_i := \exp\left(\frac{-d_{\cos}(\mathbf{1}, z_i)^2}{2\nu^2}\right) \,, \tag{3}$$

where ν is a positive *bandwidth parameter*. The intuition behind these weights is that x_i can be far away from ξ if many words are removed (in the most extreme case, $s = d$, all the words from ξ are removed). In that case, z_i has mostly 0 components, and is far away from $\mathbf{1}$.

Note that the cosine distance in Eq. (3) is actually multiplied by 100 in the current implementation of LIME. Thus there is the following correspondence between our notation and the code convention: $\nu_{\text{LIME}} = 100\nu$. For instance, the default choice of bandwidth, $\nu_{\text{LIME}} = 25$, corresponds to $\nu = 0.25$.

We now make the following important remark: **the weights only depends on the number of deletions.** Indeed, conditionally to S_i having exactly s elements, we have $z_i \cdot \mathbf{1} = d - s$ and $\|z_i\| = \sqrt{d-s}$. Since $\|\mathbf{1}\| = \sqrt{d}$, using Eq. (3), we deduce that $\pi_i = \psi(s/d)$, where we defined the mapping

$$\begin{aligned} \psi \colon [0,1] &\longrightarrow \mathbb{R} \\ t &\longmapsto \exp\left(\frac{-(1-\sqrt{1-t})^2}{2\nu^2}\right) \,. \end{aligned} \tag{4}$$

We can see in Figure 3 how the weights are given to observations: when s is small, then $\psi(s/d) \approx 1$ and when $s \approx d$, $\psi(s/d)$ which is a small quantity depending on ν. Note that the complicated dependency of the weights in s brings additional difficulty in our analysis, and that we will sometimes restrict ourselves to the large bandwidth regime (that is, $\nu \to +\infty$). In that case, $\pi_i \approx 1$ for any $1 \leq i \leq n$.

Euclidean distance between the interpretable features is used instead of the cosine distance in the tabular data version of the algorithm.

2.4 Surrogate model

The next step is to train a surrogate model on the interpretable features $z_1, \ldots, z_n$, trying to approximate the responses $y_i := f(\phi(x_i))$. In the default implementation of LIME, this model is linear and is obtained by weighted ridge regression (Hoerl and Kennard, 1970). Formally, LIME outputs

$$\hat{\beta}_n^\lambda \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^{d+1}} \left\{ \sum_{i=1}^n \pi_i (y_i - \beta^\top z_i)^2 + \lambda \|\beta\|^2 \right\} \,, \tag{5}$$

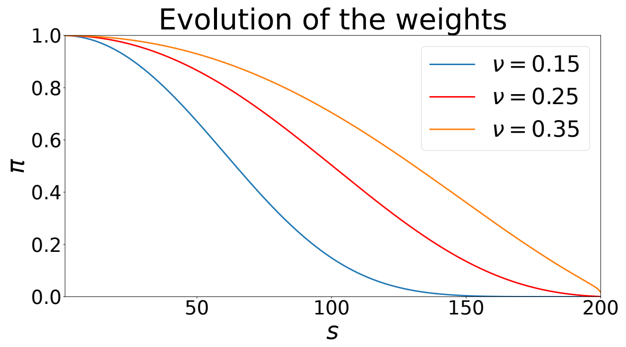

Figure 3: Weights as a function of the number of deletions for different bandwidth parameters ($\nu = 0.25$ is default). LIME gives more weights to documents with few deletions ($s/d \approx 0$ means that $\psi(s/d) \approx 1$ regardless of the bandwidth).

where $\lambda > 0$ is a regularization parameter. We call the components of $\hat{\beta}_n^\lambda$ the *interpretable coefficients*, the 0th coordinate in our notation is by convention the intercept. Note that some feature selection mechanism is often used in practice, limiting the number of interpretable features in output from LIME. We do not consider such mechanism in our analysis.

We now make a fundamental observation. In its default implementation, LIME uses the default setting of `sklearn` for the regularization parameter, that is, $\lambda = 1$. Hence the first term in Eq. (5) is roughly of order n and the second term of order d. Since we experiment in the large n regime ($n = 5000$ is default) and with documents that have a few dozen distinct words, $n \gg d$. To put it plainly, we can consider that $\lambda = 0$ in our analysis and still recover meaningful results. We will denote by $\hat{\beta}_n$ the solution of Eq. (5) with $\lambda = 0$, that is, ordinary least-squares.

We conclude this presentation of LIME by noting that the main free parameter of the method is the bandwidth ν. As far as we know, there is no principled way of choosing ν. The default choice, $\nu = 0.25$, does not seem satisfactory in many respects. In particular, other choices of bandwidth can lead to different values for interpretable coefficients. In the most extreme cases, they can even change sign, see Figure 4. This phenomenon was also noted for tabular data in Garreau and von Luxburg (2020b).

3 Main results

Without further ado, let us present our main result. For clarity's sake, we split it in two parts: Section 3.1 contains the concentration of $\hat{\beta}_n$ around β^f whereas Section 3.2 presents the exact expression of β^f.

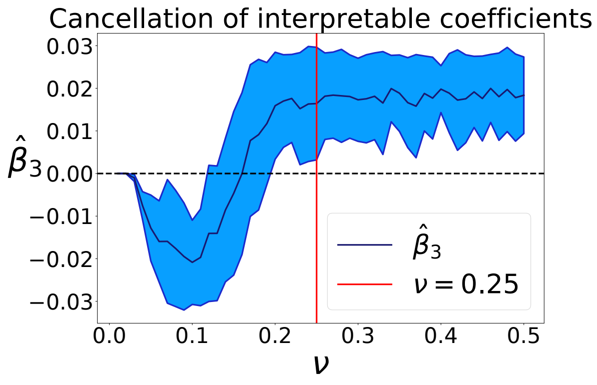

Figure 4: In this experiment, we plot the interpretable coefficient associated to the word "came" as a function of the bandwidth parameter. The red vertical line marks the default bandwidth choice ($\nu = 25$). We can see that LIME gives a negative influence for $\nu \approx 0.1$ and a positive one for $\nu > 0.2$.

3.1 Concentration of $\hat{\beta}_n$

When the number of new samples n is large, we expect LIME to stabilize and the explanations not to vary too much. The next result supports this intuition.

Theorem 1 (Concentration of $\hat{\beta}_n$). *Suppose that the model f is bounded by a positive constant M on S^{D-1}. Recall that we let d denote the number of distinct words of ξ, the example to explain. Let $0 < \epsilon < M$ and $\eta \in (0,1)$. Then, there exist a vector $\beta^f \in \mathbb{R}^d$ such that, for every*

$$n \gtrsim \max\left\{M^2 d^9 \mathrm{e}^{\frac{10}{\nu^2}}, M d^5 \mathrm{e}^{\frac{5}{\nu^2}}\right\} \frac{\log \frac{8d}{\eta}}{\epsilon^2},$$

we have $\mathbb{P}\left(\|\hat{\beta}_n - \beta^f\| \geq \epsilon\right) \leq \eta$.

We refer to the supplementary material for a complete statement (we omitted numerical constants here for clarity) and a detailed proof. In essence, Theorem 1 tells us that we can focus on β^f in order to understand how LIME operates, provided that n is large enough. The main limitation of Theorem 1 is the dependency of n in d and ν. The control that we achieve on $\|\hat{\beta}_n - \beta\|$ becomes quite poor for large d or small ν: we would then need n to be unreasonably large in order to witness concentration.

We notice that Theorem 1 is very similar in its form to Theorem 1 in Garreau and von Luxburg (2020b) except that (i) the dimension is replaced by the number of distinct words in the document to explain, and (ii) there is no discretization parameter in our case. The differences with the analysis in the tabular data framework will be more visible in the next section.

3.2 Expression of β^f

Our next result shows that we can derive an explicit expression for β^f. Before stating our result, we need to introduce more notation. From now on, we set x a random variable such that $x_1, \ldots, x_n$ are i.i.d. copies of x. Similarly, π corresponds to the draw of the π_is and z to that of the z_is.

Definition 3 (α coefficients). Define $\alpha_0 := \mathbb{E}[\pi]$ and, for any $1 \leq p \leq d$,

$$\alpha_p := \mathbb{E}[\pi \cdot z_1 \cdots z_p] . \tag{6}$$

Intuitively, when ν is large, α_p corresponds to the probability that p distinct words are present in x. The sampling process of LIME is such that α_p does not depend on the exact set of indices considered. In fact, α_p only depends on d and ν. We show in the supplementary material that it is possible to compute the α coefficients in closed-form as a function of d and ν:

Proposition 1 (Computation of the α coefficients). *Let $0 \leq p \leq d$. For any $d \geq 1$ and $\nu > 0$, it holds that*

$$\alpha_p = \frac{1}{d}\sum_{s=1}^{d}\prod_{k=0}^{p-1}\frac{d-s-k}{d-k}\psi\left(\frac{s}{d}\right) .$$

From these coefficients, we form the normalization constant

$$c_d := (d-1)\alpha_0\alpha_2 - d\alpha_1^2 + \alpha_0\alpha_1 . \tag{7}$$

We will also need the following.

Definition 4 (σ coefficients). For any $d \geq 1$ and $\nu > 0$, define

$$\begin{cases} \sigma_1 & := -\alpha_1 , \\ \sigma_2 & := \frac{(d-2)\alpha_0\alpha_2 - (d-1)\alpha_1^2 + \alpha_0\alpha_1}{\alpha_1 - \alpha_2} , \\ \sigma_3 & := \frac{\alpha_1^2 - \alpha_0\alpha_2}{\alpha_1 - \alpha_2} . \end{cases} \tag{8}$$

With these notation in hand, we have:

Proposition 2 (Expression of β^f). *Under the assumptions of Theorem 1, we have $c_d > 0$ and, for any $1 \leq j \leq d$,*

$$\beta_j^f = c_d^{-1}\Big\{\sigma_1\mathbb{E}[\pi f(\phi(x))] + \sigma_2\mathbb{E}[\pi z_j f(\phi(x))] \tag{9}$$
$$+ \sigma_3\sum_{\substack{k=1\\k\neq j}}^{d}\mathbb{E}[\pi z_k f(\phi(x))]\Big\} .$$

We also have an expression for the intercept which can be found in the supplementary material, as well as the proof of Proposition 2. At first glance, Eq. (9) is quite similar to Eq. (6) in Garreau and von Luxburg (2020b), which gives the expression of β_j^f in the tabular data case. The main difference is the TF-IDF transform in the expectation, personified by ϕ, and the additional terms (there is no σ_3 factor in the tabular data case). In addition, the expression of the σ coefficients is much more complicated than in the tabular data case. We now present some immediate consequences of Proposition 2.

Linearity of explanations. Perhaps the most striking feature of Eq. (9) is that it is **linear in** f. More precisely, the mapping $f \mapsto \beta^f$ is linear in f: for any given two functions f and g, we have

$$\beta^{f+g} = \beta^f + \beta^g \, .$$

Therefore, because of Theorem 1, the explanations $\hat{\beta}_n$ obtained for a finite sample of new examples are also approximately linear in the model to explain. We illustrate this phenomenon in Figure 5. This is remarkable: many models used in machine learning can be written as a linear combination of smaller models (*e.g.*, generalized linear models, kernel regressors, decision trees and random forests). In order to understand the explanations provided by these complicated models, one can try and understand the explanations for the elementary elements of the models first.

Large bandwidth. It can be difficult to get a good sense of the values taken by the σ coefficients, and therefore of β. Let us see how Proposition 2 simplifies in the large bandwidth regime and what insights we can gain. We denote by β_∞ the limit of β when $\nu \to +\infty$. When $\nu \to +\infty$, we prove in the supplementary material that, for any $1 \leq j \leq d$, up to $\mathcal{O}(1/d)$ terms and a numerical constant, the j-th coordinate of β_∞ is then approximately equal to

$$\left(\beta_\infty^f\right)_j \approx \mathbb{E}\left[f(\phi(x)) | w_j \in x\right] - \frac{1}{d}\sum_{k\neq j} \mathbb{E}\left[f(\phi(x)) | w_k \in x\right] .$$

Intuitively, the interpretable coefficient associated to the word w_j is high if **the expected value of the model when word w_j is present is significantly higher than the typical expected value when other words are present**. We think that this is reasonable: if the model predicts much higher values when w_j belongs to the example, it surely means that w_j being present is important for the prediction. Of course, this is far from the full picture, since (i) this reasoning is only valid for large bandwidth, and (ii) in practice, we are concerned with $\hat{\beta}_n$ which may be not so close to β^f for small n.

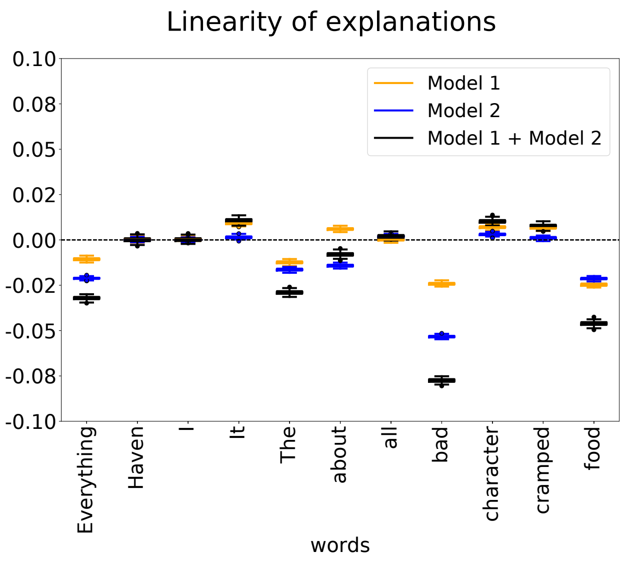

Figure 5: The explanations given by LIME for the sum of two models (here two random forests regressors) are the sum of the explanations for each model, up to noise coming from the sampling procedure.

3.3 Sketch of the proof

We conclude this section with a brief sketch of the proof of Theorem 1, the full proof can be found in the supplementary material.

Since we set $\lambda = 0$ in Eq. (5), $\hat{\beta}_n$ is the solution of a weighted least-squares problem. Denote by $W \in \mathbb{R}^{n\times n}$ the diagonal matrix such that $W_{i,i} = \pi_i$, and set $Z \in \{0,1\}^{n\times(d+1)}$ the matrix such that its ith line is $(1, z_i^\top)$. Then the solution of Eq. (5) is given by

$$\hat{\beta}_n = \left(Z^\top W Z\right)^{-1} Z^\top W y \, ,$$

where we defined $y \in \mathbb{R}^n$ such that $y_i = f(\phi(x_i))$ for all $1 \leq i \leq n$. Let us set $\hat{\Sigma}_n := \frac{1}{n} Z^\top W Z$ and $\hat{\Gamma}_n^f := \frac{1}{n} Z^\top W y$. By the law of large numbers, we know that both $\hat{\Sigma}_n$ and $\hat{\Gamma}_n^f$ converge in probability towards their population counterparts $\Sigma := \mathbb{E}[\hat{\Sigma}_n]$ and $\Gamma^f := \mathbb{E}[\hat{\Gamma}_n]$. Therefore, provided that Σ is invertible, $\hat{\beta}_n$ is close to $\beta^f := \Sigma^{-1}\Gamma^f$ with high probability.

As we have seen in Section 2, the main differences with respect to the tabular data implementation are (i) the interpretable features, and (ii) the TF-IDF transform. The first point lead to a completely different Σ than the one obtained in Garreau and von Luxburg (2020b). In particular, it has no zero coefficients, leading to more complicated expression for β^f and additional challenges when controlling $\left\|\Sigma^{-1}\right\|_{\mathrm{op}}$. The second point is quite challenging since, as noted in Section 2.1, **the TF-IDF transform of a document changes radically when deleting words at random in the document.** This is the main reason why we have to resort to approximations when dealing with linear models.

4 Expression of β^f for simple models

In this section, we see how to specialize Proposition 2 to simple models f. Recall that our main goal in doing so is to investigate whether it makes sense or not to use LIME in these cases. We will focus on two classes of models: decision trees (Section 4.1) and linear models (Section 4.2).

4.1 Decision trees

In this section we focus on simple decision trees built on the presence or absence of given words. For instance, let us look at the model returning 1 if the word "food" is present, or if "about" and "everything" are present in the document. Ideally, LIME would give high positive weights to "food," "about," and "everything," if they are present in the document to explain, and small weight to all other words.

We first notice that such simple decision trees can be written as sums of products of the binary features. Indeed, recall that we defined $z_j = \mathbf{1}_{w_j \in x}$. For instance, suppose that the first three words of our dictionary are "food," "about," and "everything." Then the model from the previous paragraph can be written

$$g(x) = z_1 + (1 - z_1) \cdot z_2 \cdot z_3 \, . \tag{10}$$

Now it is clear that the z_js can be written as function of the TF-IDF transform of a word, since $w_j \in x$ if, and only if, $\phi(x)_j > 0$. Therefore this class of models falls into our framework and we can use Theorem 1 and Proposition 2 in order to gain insight on the explanations provided by LIME. For instance, Eq. (10) can be written as $f(\phi(x))$ with, for any $\zeta \in \mathbb{R}^D$,

$$f(\zeta) := \mathbf{1}_{\zeta_1 > 0} + (1 - \mathbf{1}_{\zeta_1 > 0}) \cdot \mathbf{1}_{\zeta_2 > 0} \cdot \mathbf{1}_{\zeta_3 > 0} \, .$$

By linearity, it is sufficient to know how to compute β^f when f is a product of indicator functions.

We now make an important remark: since the new example $x_1, \ldots, x_n$ are created by deleting words at random from the text ξ, x **only contains words that are already present in** ξ. Therefore, without loss of generality, we can restrict ourselves to the local dictionary (the distinct words of ξ). Indeed, for any word w not already in ξ, $\mathbf{1}_{w \in x} = 0$ almost surely. As before, we denote by D_ℓ the local dictionary associated to ξ, and we denote its elements by $w_1, \ldots, w_d$. We can compute in closed-form the interpretable coefficients for a product of indicator functions:

Proposition 3 (Computation of β^f, product of indicator functions). *Let $J \subseteq \{1, \ldots, d\}$ be a set of p distinct indices and set $f(x) = \prod_{j \in J} \mathbf{1}_{x_j > 0}$. Then, for any $j \in J$,*

$$\beta_j^f = c_d^{-1}\left[\sigma_1 \alpha_p + \sigma_2 \alpha_p + (d - p)\sigma_3 \alpha_{p+1} + (p-1)\sigma_3 \alpha_p\right]$$

and, for any $j \in \{1, \ldots, d\} \setminus J$,

$$\beta_j^f = c_d^{-1}\left[\sigma_1 \alpha_p + \sigma_2 \alpha_{p+1} + (d - p - 1)\sigma_3 \alpha_{p+1} + p\sigma_3 \alpha_p\right].$$

In particular, when $p = 0$, Proposition 3 simplifies greatly and we find that $1 \leq k \leq d$, $\beta_k^f = \mathbf{1}_{k=j}$. It is already a reassuring result: when the model is just indicating if a given word is present, **the explanation given by LIME is one for this word and zero for all the other words**.

It is slightly more complicated to see what happens when $p \geq 1$. To this extent, let us set $j \in J$ and $k \notin J$. Then it follows readily from Proposition 14 that

$$\beta_j^f - \beta_k^f = c_d^{-1}(\sigma_2 + \sigma_3)(\alpha_p - \alpha_{p+1}) \, .$$

Since $\alpha_p \approx 1/(p+1)$ and $\sigma_2 + \sigma_3 \approx 6$, we deduce that $\beta_j^f \gg \beta_k^f$. Moreover, from Definition 3 and 4 one can show that $\beta_k^f = \mathcal{O}(1/d)$ when ν is large. Thus Proposition 14 tells us that **LIME gives large positive coefficients to words that are in the support of f and small coefficients to all the other words**. This is a satisfying property.

Together with the linearity property, Proposition 14 allows us to compute β^f for any decision tree that can be written as in Eq. (10). We give an example of our theoretical predictions in Figure 6. As predicted, **the words that are pivotal in the prediction have high interpretable coefficients, whereas the other words receive near-zero coefficients**. It is interesting to notice that words that are near the root of the tree receive a greater weight. We present additional experiments in the supplementary material.

4.2 Linear models

We now focus on linear models, that is, for any document x,

$$f(\phi(x)) := \sum_{j=1}^{d} \lambda_j \phi(x)_j \, , \tag{11}$$

where $\lambda_1, \ldots, \lambda_d$ are arbitrary fixed coefficients. We have to resort to approximate computations in this case: from now on, we assume that $\nu = +\infty$. We start with the simplest linear function: all coefficients are zero except one, that is, $\lambda_k = 1$ if $k = j$ and 0 otherwise in Eq. (11), for a fixed index j. We need to

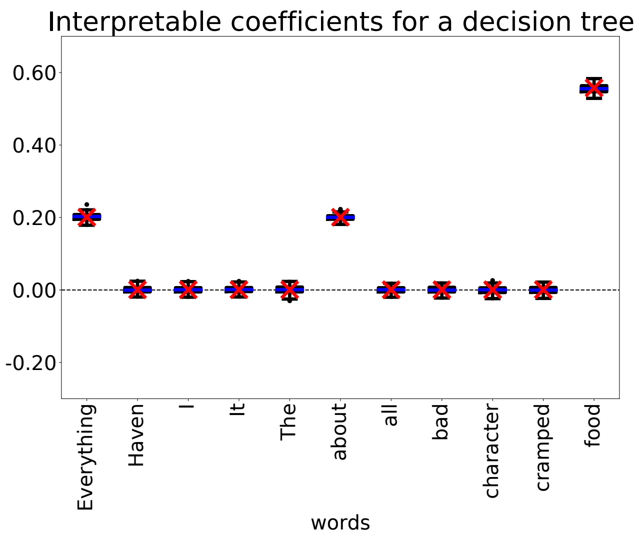

Figure 6: Theory *vs* practice for the tree defined by Eq. (10). The black whisker boxes correspond to 100 runs of LIME with default settings ($n = 5000$ new examples and $\nu = 0.25$) whereas the red crosses correspond to the theoretical predictions given by our analysis. The example to explain is a Yelp review with $d = 35$ distinct words.

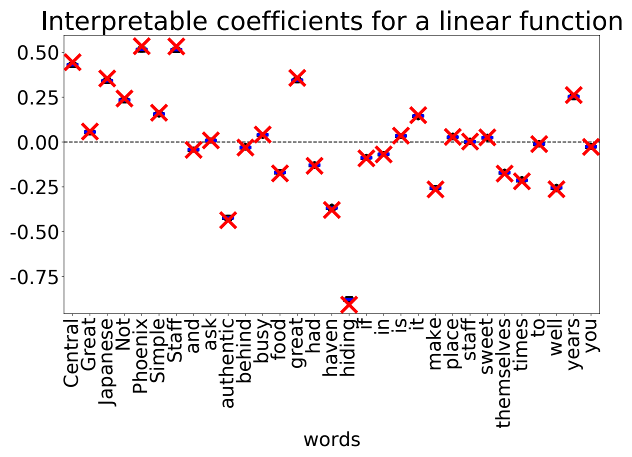

Figure 7: Theory *vs* practice for an arbitrary linear model. The black whisker boxes correspond to 100 runs of LIME with default settings ($n = 5000$ and $\nu = 0.25$). The red crosses correspond to our theoretical predictions: $\beta_j \approx 1.36\lambda_j\phi(\xi)_j$. Here $d = 29$.

introduce additional notation before stating or result. For any $1 \leq j \leq d$, define

$$\omega_k := \frac{m_j^2 v_j^2}{\sum_{\ell=1}^d m_\ell^2 v_\ell^2} \, ,$$

where the m_ks and v_ks were defined in Section 2.1. For any J that is a strict subset of $\{1, \ldots, d\}$, define $H_S := \sum_{j\in J} \omega_j$. Recall that S denotes the random subset of indices chosen by LIME in the sampling step (see Section 2.2). Define $E_j = \mathbb{E}\left[(1 - H_S)^{-1/2} \middle| S \not\ni j\right]$ and for any $k \neq j$, $E_{j,k} = \mathbb{E}\left[(1 - H_S)^{-1/2} \middle| S \not\ni j, k\right]$. Then we have the following:

Proposition 4 (Computation of β^f, linear case). *Let $1 \leq j \leq d$ and assume that $f(\phi(x)) = \phi(x)_j$. Then, for any $1 \leq k \leq d$ such that $k \neq j$,*

$$\left(\beta^f_\infty\right)_k = \left[2E_{j,1} - \frac{2}{d}\sum_{\ell\neq k,j} E_{j,\ell}\right]\phi(\xi)_j + \mathcal{O}\left(\frac{1}{d}\right) \, ,$$

and

$$\left(\beta^f_\infty\right)_j = \left[3E_j - \frac{2}{d}\sum_{k\neq j} E_{j,k}\right]\phi(\xi)_j + \mathcal{O}\left(\frac{1}{d}\right) \, .$$

Proposition 4 is proved in the supplementary material. The main difficulty is to compute the expected value of $\phi(x)_j$: this is the reason for the E_j terms, for which we find an approximate expression as a function of the ω_ks. Assuming that the ω_k are small, we can further this approximation and show that $E_j \approx 1.22$ and $E_{j,k} \approx 1.15$. In particular, **these expressions do not depend on j and k.** Thus we can drastically simplify the statement of Proposition 4: for any $k \neq j$, $\left(\beta^f_\infty\right)_k \approx 0$ and $\left(\beta^f_\infty\right)_j \approx 1.36\phi(\xi)_j$. We can now go back to our original goal, Eq. (11). By linearity, we deduce that

$$\forall 1 \leq j \leq d, \quad \left(\beta^f_\infty\right)_j \approx 1.36 \cdot \lambda_j \cdot \phi(\xi)_j \, . \qquad (12)$$

In other words, up to a numerical constant and small error terms depending on d, **the explanation for a linear f is the TF-IDF value of the word multiplied by the coefficient of the linear model.** We believe that this behavior is desirable for an interpretability method: large coefficients in the linear model should intuitively be associated to large interpretable coefficients. But at the same time the TF-IDF of the term is taken into account.

We observe a very good match between theory and practice (see Figure 7). Surprisingly, this is the case even though we assume that ν is large in our derivations, whereas ν is chosen by default in all our experiments. We present experiments with other bandwidths in the supplementary.

5 Conclusion

In this work we proposed the first theoretical analysis of LIME for text data. In particular, we provided a closed-form expression for the interpretable coefficients when the number of perturbed samples is large. Leveraging this expression, we exhibited some desirable behavior of LIME such as the linearity with respect to the model. In specific cases (simple decision trees and linear models), we derived more precise expression, showing that LIME outputs meaningful explanations in these cases.

As future work, we want to tackle more complex models. More precisely, we think that it is possible to obtained approximate statements in the spirit of Eq. (12) for models that are not linear.

Acknowledgments

This work was partly funded by the UCA DEP grant. The authors want to thank André Galligo for getting them to know eachother.

References

A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen. A survey of the state of Explainable AI for Natural Language Processing. *arXiv preprint arXiv:2010.00711*, 2020.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

S. Filipovski. Improved Cauchy-Schwarz inequality and its applications. *Turkish journal of inequalities*, 3(2):8–12, 2019.

D. Garreau and U. von Luxburg. Explaining the explainer: A first theoretical analysis of LIME. In *Proceedings of the 33rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1287–1296, 2020a.

D. Garreau and U. von Luxburg. Looking Deeper into Tabular LIME. *arXiv preprint arXiv:2008.11092*, 2020b.

R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

D. M. W. Powers. Applications and explanations of Zipf's law. In *New methods in language processing and computational natural language learning*, 1998.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

M. T. Ribeiro, S. Singh, and C. Guestrin. "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, 2016.

P. Ross. Generalized hockey stick identities and n-dimensional blockwalking. *The College Mathematics Journal*, 28(4):325, 1997.

P. Spyns. Natural language processing in medicine: an overview. *Methods of information in medicine*, 35(4-5):285–301, 1996.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.