Model-Agnostic Local Explanations with Genetic Algorithms for Text Classification

Qingfeng Du
School of Software Engineering
Tongji University
Shanghai, China
du_cloud@tongji.edu.cn

Jincheng Xu
School of Software Engineering
Tongji University
Shanghai, China
xujincheng@tongji.edu.cn

***Abstract*—The interpretability of black-box text classification models has been receiving widespread attention in recent years accompanying the growing popularity of artificial intelligence. To garner user trust on the model's decision-making process, it is imperative to provide faithful instance-wise justifications and rationalize the prediction in a human-readable way. In this paper, we address this challenge by introducing *Locally Universal Rules (LURs)* as model-agnostic local explanations. LURs are a subset of input words sufficient for the model to arrive at a particular prediction, even if the rest of words are perturbed slightly. We show the identification of the optimal LUR is NP-complete. Consequently, we propose a population-based algorithm *LUR-Locator* to perform the constrained optimization efficiently. We conduct extensive experiments to evaluate our algorithm on a cross product of well-established text classification datasets and models. The empirical results demonstrate that LURLocator can efficiently generate high-quality local explanations, as compared to existing explanatory methods.**

***Index Terms*-Model-Agnostic Explanations; Local Explanations; Genetic Algorithms; Text Classification**

I. Introduction

In the era of Artificial Intelligence (AI), machine learning or deep learning models have been widely deployed in a variety of real-world applications. However, due to the increasingly complex architectures, most models act as a black box without any clear explanations. The lack of interpretability can pose potential threats to the lives of individuals in today's AI systems [1]. To establish widespread public trust on model's behaviors, it is necessary to develop explanatory methods to provide insights into these opaque model.

One of the simplest ways to understand the decision-making process is to select a subset of input features as *local explanations* [2], such as a list of words for texts, a group of super-pixels for images or a set of if-else rules for tabular data, where the slight perturbations on the rest of features can hardly change the current prediction. These local explanations are presented in a feature-selective way [3], where a subset of input features responsible for the model's prediction are selected. Compared to feature-additive local explanations [3] which compute the numerical weights for each input feature as their separate contributions to the model's decision and provide too much redundant information, the feature-selective local explanations are very concise and straightforward. They are easy to understand even by non-expert users. For example, users may want to understand why their comments on social medias are judged as offensive sentences by the AI system, and by feature-selective local explanations they can quickly locate the key words and modify them.

In this paper, we introduce *Locally Universal Rules (LUR)* as model-agnostic local explanations from the perspective of feature-selection for text classification tasks, where "locally" means these explanations can only approximate the model in the neighborhood of a particular data point rather than keep faithful globally [4][5], and "universal" means the model prediction will keep the same in the perturbed distribution as long as these local explanations hold. Subsequently we propose a population-based optimization algorithm, *LURLocator*, to reduce the search space greatly when identifying the optimal LUR. We shed light on Natural Language Processing (NLP) as the driving domain here to present LURs and LURLocator in details, but our work can be easily extended to other domains, such as image data or tabular data.

II. Related Work

Interpretable AI systems have been gaining the spotlights due to its significant value in human trust nowadays [4]. Some work perform a deep investigation on the important building blocks of model architectures such as attention [6], or on a specific family of models, such as RNN [7] and CNN [8]. Usually, model-specific explanations are more applicable for AI experts to learn the intrinsic mechanisms and debug the model [2]. The second group offer transparency on any deep learning models. They are either implemented with full access to internal parameters [9][10], or in a black-box scenario [11]. The third group zoom in on model-agnostic explanations [4][12], and they can be used to interpret either machine learning models or deep learning models.

Local explanations are closely related to model-agnostic explanatory methods. However, when the term of *local explanations* is mentioned in literature, it usually reflects diverse motivations when dealing with user-specific requirements, and consequently researchers establish different but sometimes

This work was supported by National Key R&D Program of China (Grant No. 2020YFB2103300)
DOI reference number: 10.18293/SEKE2021-040

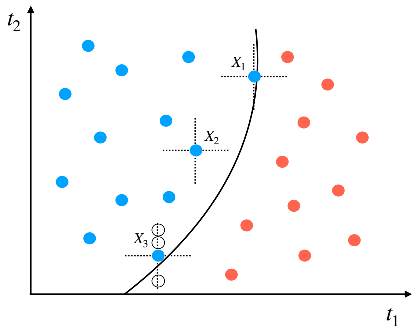

Fig. 1. A toy prediction function for classification. The two axes are separate input dimensions, the black curve is the decision boundary, and the blue or red circles are documents belonging to two different classes.

overlapping definitions. In [13], local explanations are referred to as *rationales* which are short and coherent, yet sufficient pieces of text for predictions in NLP, and an encoder-generator framework is proposed to automatically generate rationales to regularize the model during training. In [4], *LIME* is presented to learn an interpretable model by locally approximating a particular data point. In [14], local explanations are considered as prediction interpretability (as opposed to model interpretability), and a variety of local explanatory methods are evaluated with human judgements. *Anchors* are formally introduced in [12] as local and sufficient conditions so that the slight perturbations on the rest of the features have negligible effects on the prediction. *sufficient input subsets* are presented in [2] as a minimal input pattern to make sure that the model can produce the same prediction even though all other feature values are lost. Among all the previous work, the definition of LURs is most similar to that of anchors [12]. We will demonstrate the comparison between them in experiments.

III. Problem Setup and Analysis

Firstly we present the definition of *perturbation distribution*:

Definition 1 (Perturbation Distribution). *Let $X = (t_1, t_2, \cdots, t_n)$ be an input document consisting of n words, and $A = X * I$ be a set of words where $I = \{0, 1\}^n$ indicates the presence or absence of each word. The perturbation distribution $\mathcal{D}_X(\cdot|A)$ is a conditional distribution of documents. Suppose we have a document $X' = (t'_1, t'_2, \cdots, t'_n)$ sampled from $\mathcal{D}_X(\cdot|A)$. For each $t \in X \wedge t \in A$, we have $t_i = t'_i$ where i is the position of t in X. For each $t \in X \wedge t \notin A$, we have $L(t_i, t'_i) < \epsilon$ where L is the distance of words and ϵ is a numerical threshold.*

Refer to Figure 1 for an example. In the neighborhood of X_3, the vertical dashed line is the perturbation distribution $\mathcal{D}_X(\cdot|A = \{t_1\})$ and the hollow circles are possible documents X' sampled from $\mathcal{D}_X(\cdot|A = \{t_1\})$.

In perturbation-based local explanations, $L(t_i, t'_i)$ should be treated as the distance of words in a semantically meaningful space even when the words are embedded in another representation space in the model, since the perturbation distribution $\mathcal{D}_X(\cdot|A)$ should be interpretable to human eyes [4][12]. In reality, when drawing samples from $\mathcal{D}_X(\cdot|A)$, we have a set of candidate synonyms in the vicinity of t. It is worth mentioning that a candidate word at the edge of the chosen area is likely to have a large semantic distance from t, and consequently the sampling noise will be introduced. To improve the robustness of $\mathcal{D}_X(\cdot|A)$ against the sampling noise, we use the softmax function parameterized by temperature T to compute the normalized probability of words sampling:

$$p(t'_i|t_i) = \frac{exp(L(t_i, t'_i)/T)}{\sum exp(L(t_j, t'_j)/T)} \tag{1}$$

Based on $\mathcal{D}_X(\cdot|A)$, the LUR can be defined as follows:

Definition 2 (LUR). *Let $F : X \rightarrow \mathbb{R}^1$ be a function representing a text classification model. Suppose $D'_A = \{X'|X' \sim \mathcal{D}_X(\cdot|A)\}$ is a dataset approximating the true distribution of $\mathcal{D}_X(\cdot|A)$. The locally universal rule (LUR), denoted by A, is such a set of words satisfying $Acc(A) = \mathbb{E}_{D'_A}[F(X) = F(X')] \geq \tau$, where $Acc(A)$ denotes the performance achieved by A and $\tau \in [0, 1]$ is a specified threshold serving as the lower bound on the expected performance.*

We show the existence of the LUR as follows:

Proposition 1. *For a document X, a text classification model F and a threshold τ, at least one LUR exists.*

Proof: Assume to the contrary that no LUR exists for some X, F and τ. Assign X to A, and we have $Acc(A) = \mathbb{E}_{D'_A}[F(X) = F(X')] = \mathbb{E}_{\{X\}}[F(X) = F(X')] = 1 \geq \tau$. Therefore A is a valid LUR, and the original assumption must be false. So Proposition 1 is true.

Refer to Figure 1 for better understandings. Suppose $\tau = 0.9$. For X_1, since $Acc(A = \{t_2\}) \approx 0.5 < \tau$, $\{t_2\}$ is not a valid LUR. There only exists one LUR $A = \{t_1\}$. Similarly, $A = \{t_1\}$ or $\{t_2\}$ for X_2, and $A = \{t_1, t_2\}$ for X_3.

Subsequently, we define *the optimal LUR* as follows:

Definition 3 (The Optimal LUR). *Suppose $S : A \rightarrow \mathbb{R}^1$ be a function representing the number of words in A. For an optimal LUR A, it should satisfy the following conditions: (1) $Acc(A) \geq \tau$ (2) $\nexists A^*, Acc(A^*) \geq \tau \wedge S(A^*) < S(A)$.*

The LUR with a shorter length is preferred as the optimal LUR for two reasons: (1) Local explanations should be concise for users to understand [2]. (2) A shorter local explanation is expected to cover more instances in realistic settings [12].

Now we present the identification of the optimal LUR as a constrained optimization problem formally:

Definition 4 (The Identification Problem). *For an input document X, the identification of the optimal LUR is to find such a minimal set of words A, so that $A = \arg\min_{A \subseteq X} S(A)$, subject to $Acc(A) \geq \tau$.*

We show this problem is computationally intractable:

Proposition 2. *The identification of the optimal LUR is an NP-complete problem.*

Proof: We prove that a simpler version of the original problem can be reduced to the NP-complete subset sum problem [15]. The simpler problem is $A = \arg\max_{A\subset X} Acc(A)$. Let the set of non-negative values be $\{v_1, v_2, \cdots, v_n\}$, and the target be K. Let the embedding of $t_i \in X$ be $E(t_i) = (v_i, 0, \cdots, 0)$ and the embedding of its synonym $t'_i \in X'$ be $E(t'_i) = (0, 0, \cdots, 0)$. The classifier is defined as follows:

$$F(X) = g(\sum_{i=0}^{n}\sum_{j} E(t_i)_j)$$
$$g(a) = \begin{cases} 0 & a \neq K \\ 1 & a = K \end{cases} \tag{2}$$

where $g(x)$ is a step function. To solve the problem, we have to find a set of words as A, so that the sum of all the values in the embedding space equals to K exactly. Now the simplified problem suffices the subset sum problem. Hence, we have that the original identification problem is NP-complete.

IV. The Proposed Algorithm

The possible search space of different combinations of words grows exponentially with the document length if we aim to identify the optimal LUR, and it is impossible to deal with the heavy computational burden in practice. Instead of an exhaustive search, we develop a heuristic algorithm, *LURLocator*, following the principle of genetic algorithms, to search for a near-optimal solution efficiently. The key operators include *initialization*, *fitness*, *selection*, *crossover* and *mutation*. The pseudo-code of LURLocator is shown in Algorithm 1, and the framework is illustrated in Figure 2.

The first step of our algorithm is to initialize an empty set D' (line 1), which is the set of instances sampled from the perturbation distribution and will be iteratively expanded later. All the LURs will be evaluated on D' to estimate the accuracy.

Line 2-3 implements the *initialization* operator, returning an array of chromosomes as the initial population $\mathcal{P}^0$. We select one word $t_{i\%n}$ (% is the modular operation) as the LUR A_i in each chromosome $\mathcal{P}^0_i$. The chromosome is encoded as a binary vector of length n, where the words in A_i take the value of 1 and others take 0.

The evolution continuous for k_{gen} generations in an iterative process to search for the near-optimal LUR (line 4).

In the $fitness$ operator (line 5-8), we measure the quality of the population $\mathcal{P}^{i-1}$ in accuracy. Suppose D' has also been encoded with binary vectors, where the words in X take the value of 1 and others take 0. Firstly we count the number of $X' \in D'_{A_j}$ in D', and at least B_1 instances are expected to estimate the accuracy of A_j (line 8). If there are not enough instances, we sample B_1 instances from the perturbation distribution $\mathcal{D}_X(\cdot|A_j)$ immediately to expand D' (line 6-7). The prediction results through F will be recorded, so that we do not have to perform repetitive predictions for the same instance in the subsequent iterations.

Algorithm 1 The LURLocator Algorithm

```
Input: Input document $X = (t_1, t_2, \cdots, t_n)$; The function $F$ as the text classification model; The function $S$ as the number of words in a LUR; Batch sizes $B_1$, $B_2$; The expected performance $\tau$; Number of generations $k_{gen}$; Size of the population $k_{pop}$; Number of parents $k_{par}$; Number of mutations $k_{mut}$
Output: The near-optimal LUR $A$.
1:  $D' = \varnothing$
2:  for $i = 1, 2, \cdots, k_{pop}$ do
3:      $\mathcal{P}^0_i \leftarrow$ Encode $A_i = \{t_{i\%n}\} \subseteq X$ as a binary vector
4:  for $i = 1, 2, \cdots, k_{gen}$ do
5:      for all $A_j \in \mathcal{P}^{i-1}$ do
6:          if $D'_{A_j}.size < B_1$ then
7:              $D'_{A_j} \leftarrow$ Sample $B_1$ instances from $\mathcal{D}_X(\cdot|A_j)$
8:          $Acc^i(A_j) \leftarrow \mathbb{E}_{D'_{A_j}}[F(X) = F(X')]$
9:      $Fit^i \leftarrow \{\mathcal{P}^{i-1}_j | Acc^i(A_j) \geq \tau\}$
10:     if $Fit^i.size \geq k_{par}$ then
11:         $Fit^i \leftarrow$ Sort $Fit^i$ so that $\forall j_1 \leq j_2, S(A_{j_1}) \leq S(A_{j_2})$
12:         $Parent^i \leftarrow Fit^i_1, Fit^i_2, \cdots, Fit^i_{k_{par}}$
13:     else
14:         $\mathcal{P}^{i-1} \leftarrow$ Sort $\mathcal{P}^{i-1}$ so that $\forall j_1 \leq j_2, Acc^i(A_{j_1}) \geq Acc^i(A_{j_2})$
15:         $Parent^i \leftarrow \mathcal{P}^{i-1}_1, \mathcal{P}^{i-1}_2, \cdots, \mathcal{P}^{i-1}_{k_{par}}$
16:     for $j = 1, 2, \cdots, (k_{pop} - k_{par})$ do
17:         $Child^i_j \leftarrow$ Randomly select two individuals from $Parent^i$ and recombine them
18:         $Child^i_j \leftarrow$ Randomly flip $k_{mut}$ genes in $Child^i_j$
19:     $\mathcal{P}^i \leftarrow Parent^i + Child^i$
20: for all $A \in Parent^{k_{gen}}$ do
21:     $D'_A \leftarrow$ Sample $B_2$ instances from $\mathcal{D}_X(\cdot|A)$ to $D'$
22:     $Acc(A) \leftarrow \mathbb{E}_{D'_A}[F(X) = F(X')]$
23: $A \leftarrow \arg\min_{A\in Parent^{k_{gen}}} S(A)$, subject to $Acc(A) \geq \tau$
24: return $A$
```

When we attempt to select D'_{A_j} from D', we have to compare each instance in D' with the encoded chromosome $\mathcal{P}^{i-1}_j$ to see whether they match with each other. The comparison of 0/1 bits runs in $\mathcal{O}(n \times D'.size)$ time. Given a large n or $D'.size$, the process will be time-consuming. To decrease the number of comparisons and accelerate the selection of D'_{A_j}, we propose an alternative approach based on inverted index. We transpose the matrix of D', so that each row represents a word in X and each column represents an instance in D'. For each word in A_j, we select the corresponding rows in the transposed matrix and multiply these 0/1 rows in an element-wise manner. In the resulting vector of length $D'.size$, the elements are equal to 1 only if all of its multiplicators are equal to 1. Refer to the second subgraph of the second row in Figure 2 for an example, where $A_j = \{t_4, t_5\}$. When we compute the element-wise multiplication of $t_4 = (0, 1, ..., 0)$

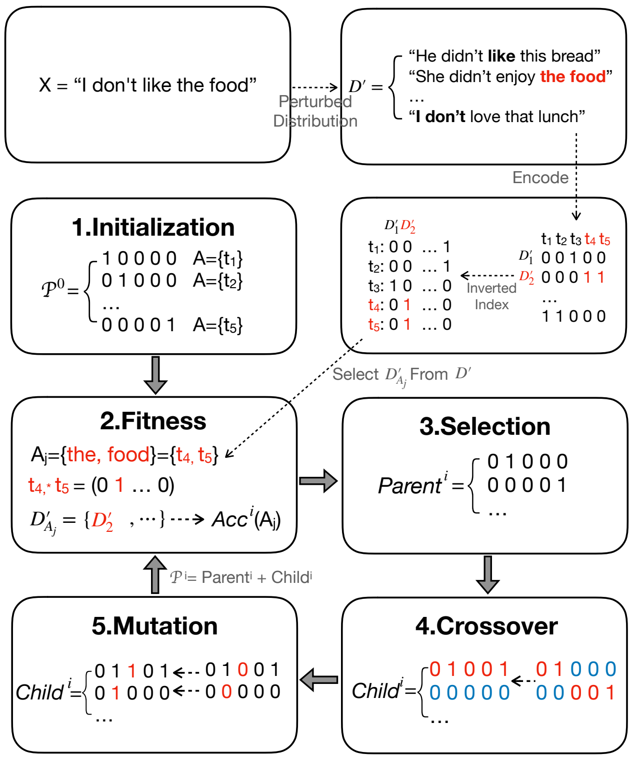
X = "I don't like the food"
Perturbed Distribution
D' =
"He didn't like this bread"
"She didn't enjoy the food"
...
"I don't love that lunch"
Encode
1.Initialization
2.Fitness
3.Selection
4.Crossover
5.Mutation
Inverted Index
Select D'_{A_j} From D'
$\mathcal{P}^i$= Parent^i + Child^i

Fig. 2. The framework of LURLocator

and $t_5 = (0, 1, ..., 0)$ in the transposed matrix and get the result of $(0, 1, ..., 0)$, only the second element (indicating D'_2) is equal to 1, meaning that D'_2 matches $A_j = \{t_4, t_5\}$ and belongs to D'_{A_j}. In other words, the index of the element equal to 1 in the resulting vector is the position of the instance in D' that belongs to D'_{A_j}. This alternative approach runs in $\mathcal{O}(S(A_j) \times D'.size)$ time and achieves a speed-up of $(n/S(A_j) - 1)$x compared to the original implementation. The gains in efficiency is especially significant when $S(A_j) \ll n$.

We implement the *selection* operator in line 9-15. We filter out the chromosomes whose LURs reach the performance threshold of τ, denoted as Fit^i (line 9). If the size of Fit^i is larger than k_{par}, it means that we already have enough chromosomes as parents, from which we can select the ones encoded with the k_{par} shortest LURs as $Parent^i$ for futural breeding (line 10-12). Otherwise, we sort all the chromosomes in $\mathcal{P}^{i-1}$ according to Acc^i, and retrieve the ones achieving the k_{par} best accuracies (line 13-15). This branch is to make sure that if there are not enough qualified chromosomes achieving the expected fitness, we can still preferentially select more fit individuals with as high accuracy as possible.

In *crossover* (line 17), a set of children will be generated by randomly choosing pairs of parents and performing crossover operations where the cut point is at the center of each parent. In this way, the new offspring has the genes from two different parents whose characteristics can be partly inherited. In the stage of *mutation* (line 18), k_{mut} genes in the children in the representation of binary vectors will be flipped (1 to 0 or 0 to 1). Both crossover and mutation operators ensure the diversity of individuals in the subsequent new population, so that we can explore more candidate solutions.

Both $Parent^i$ and $Child^i$ will be added together as the ith generation of the population (line 19), for the purpose that both the qualified solutions ($Parent^i$) and the possible candidate solutions ($Child^i$) can be safely preserved.

After k_{gen} generations, we select the optimal LUR from $Parent^{k_{gen}}$ with the shortest length and the expected accuracy (line 20-24). To avoid over-fitting on D', we re-sample B_2 instances as D'_{A_i} to evaluate each possible LUR. If no LUR satisfies the expected τ, we return $A = X$ as the near-optimal LUR according to Proposition 1. If more than one LUR meets the criteria, we return the LUR with the highest accuracy.

V. Experiments

A. Experimental Setup

Datasets: We prepare four different benchmarking datasets widely used in the task of text classification, including the sentence polarity dataset from Rotten Tomatoes web pages (RT) [16], AG's News (AG), DBPedia (DBP) and Yahoo Answers (Yah) [17].

Machine Learning Models: We train three different machine learning models, including logistic regression (LR), multinomial naive bayes (NB) and support vector machine (SVM). We use Anchor [12] as the baseline to generate the local explanations here. As we will show later, Anchor is very inefficient when dealing with long documents, so we perform the comparisons on RT whose documents are shorter.

Deep Learning Models: We choose two popular text classification models, including FastText and TextCNN, as the target deep learning models. Later, we will evaluate some gradient-based explanatory methods specially designed for neural networks, so we distinguish the experiments on deep learning models from the traditional machine learning models.

Algorithm Details: We set $B_1 = 100, B_2 = 500, k_{gen} = 10, k_{pop} = 10 * n, k_{par} = n, k_{mut} = 1$, where n is the length of the document. For Yah dataset, we set $k_{gen} = 20$ since it consists of longer documents. We present some practioners' guides for the fine-tuning of parameters: B_1 and B_2 are closely related to the trade-off between the algorithm's efficiency and the LUR's accuracy estimation. A larger k_{gen} improves the possibility of identifying the near-optimal LUR whose natural length is long; A larger k_{pop} contributes to finding the LUR as short as possible; The size of k_{par} should be at least n to ensure a good start point; We suggest setting $k_{mut} = 1$ to ensure a steady evolution. We use the 100 nearest words in the pre-trained embedding space from the vocabulary of each dataset to generate the perturbation distribution instead of setting a fixed ϵ. The performance threshold τ is set to 0.9.

B. Evaluating LURs in Machine Learning

On the test set of RT, we generate LURs with our proposed LURLocator algorithm. We also generate anchors [12] with their original implementations, including the greedy-search

TABLE I
COMPARISONS BETWEEN LUR AND ANCHOR EXPLANATIONS IN MACHINE LEARNING.

Model	LUR			Anchor (Greedy Search)			Anchor (Beam Search)		
	Accuracy	Length	Time	Accuracy	Length	Time	Accuracy	Length	Time
LR	**93.31%**	3.15	2.02s	92.19%	2.93	4.26s	91.38%	2.88	15.26s
NB	**94.02%**	3.46	2.03s	93.04%	3.55	5.36s	92.54%	3.39	21.94s
SVM	**92.90%**	3.63	1.99s	92.01%	3.43	5.21s	91.21%	3.34	22.01s

TABLE II
EXEMPLARY LOCAL EXPLANATIONS FROM THE TEST SET OF RT.

Document	Model	LUR	Anchor (Greedy Seach)	Anchor (Beam Seach)
Two generations within one family	LR	intelligent	family, intelligent	intelligent, age
test boundaries in this intelligent	NB	boundaries	boundaries	within
and restrained coming-of-age drama.	SVM	drama	within, boundaries, drama	within, boundaries, intelligent
The beautiful, unusual music	LR	beautiful	beautiful,films	beautiful, may
is this film's chief draw, but	NB	chief	unusual, chief	beautiful, chief
its dreaminess may lull you to sleep.	SVM	beautiful	beautiful, music, draw	beautiful, draw

algorithm and the beam-search algorithm. For each local explanation A, we evaluate it on D'_A which consists of 10000 test instances sampled from $\mathcal{D}_X(\cdot|A)\}$. The average test accuracy has been reported in Table I. Besides, we also report the average length of local explanations as well as the average time for the generation of one local explanation. The time cost of searching synonyms in the vocabulary has been excluded from the reported time, because synonyms can be identified in advance and saved in a hash table.

It can be clearly seen that our algorithm produces reasonable accuracy gains compared to the two baselines. The average length is comparable for LURs and anchors while significantly shorter than the average document length, so both local explanations achieve a similar level of conciseness and coverage in practical use. In terms of the running time, LURLocator leads to better efficiency compared to the two baselines.

Since genetic algorithms are stochastic and it is impossible to report the average-case complexity for LURLocator, we analyze it in the *worst case*. The time overhead mainly depends on model predictions. If we need to sample a new batch of instances every time, model predictions run in $\mathcal{O}(k_{gen} * k_{pop} * B_1 + k_{par} * B_2)$ time. Given our suggested parameters, $\mathcal{O}$ scales linearly with the document length n.

To further investigate the relationship between the document length and the required time, we report the distribution of time cost in Figure 3. For anchors, we only report the greedy-search algorithm since the beam-search algorithm requires much more time. Along the vertical direction, most green markers are above the red markers, and the distribution of green markers is more discrete. We attribute this observation to the fact that LURLocator is more efficient and stable when dealing with a fixed-length document. Along the horizontal direction, while the time cost of both algorithms keeps increasing with document length, our algorithm shows slower growth definitely with a higher Pearson's correlation coefficient. The results demonstrate that our algorithm has better scalability when dealing with longer documents.

We display some examples in Table II. Interestingly, the selected words from LURs and anchors are often overlapped with each other. A possible explanation is that these words contain the most important information for the current prediction. Besides, the local explanations across different models vary a lot. In fact, local explanations only reflect the model's decision-making process rather than human reasoning, and different models can have their own views on the decision boundary, which results in different local explanations for the same instance. It is worth mentioning that the examples themselves can not indicate which explanatory method is more faithful, since the objective judgments does not necessarily align with the true model views.

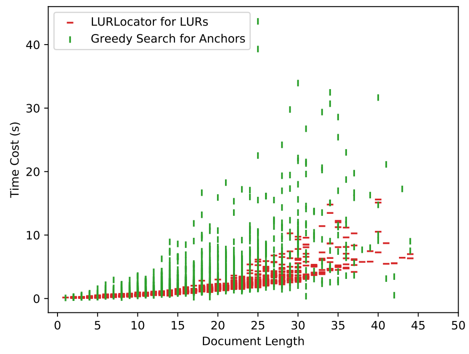

Fig. 3. The required time for the generation of local explanations on the test set of RT. The model is LR here. Each scatter point represents a test sample. The horizontal axis represents the document length, and the horizontal axis represents the time cost. (LURLocator: Pearson $\rho = 0.797$, p-value=1.27e-23; Greedy Search: Pearson $\rho = 0.589$, p-value=1.52e-98)

C. Evaluating LURs in Deep Learning

We compare LURLocator with one perturbation-based explanatory method LIME [4] and three feature-additive methods popular in deep learning, including sensitivity analysis (SA) [9], gradients $\times$ inputs (GI) [10] and leave-one-out (LOO) [11]. Since the last three methods assign importance scores to every input features, we select the most salient words

TABLE III
COMPARISONS BETWEEN LURS AND OTHER LOCAL EXPLANATIONS IN DEEP LEARNING.

Dataset	Model	Length	LURLocator	LIME	Rand	SA	GI	LOO
RT.	FastText	1.79	**99.04%**	44.64%	44.92%	45.21%	45.50%	45.21%
	TextCNN	1.71	**98.47%**	58.62%	54.69%	60.34%	64.94%	64.37%
AG.	FastText	2.42	**95.48%**	58.65%	57.15%	57.54%	62.31%	63.29%
	TextCNN	2.29	**96.52%**	75.43%	71.40%	76.82%	78.60%	77.95%
DBP.	FastText	2.24	**97.47%**	92.89%	92.38%	92.71%	94.29%	93.92%
	TextCNN	1.96	**97.05%**	92.07%	91.19%	92.64%	93.67%	93.89%
Yah.	FastText	6.88	**93.64%**	40.17%	38.96%	39.17%	41.83%	40.70%
	TextCNN	7.95	**93.77%**	75.86%	71.25%	78.89%	80.87%	79.18%

as sufficient conditions whose length is the same as LUR. We also set a *Rand* baseline to randomly select words as local explanations. Table III reports the average test accuracy and the average length of LURs.

We can observe that the average length of LURs on Yah is longer than other datasets because the average document length on Yah is much longer. In fact, if we increase k_{gen} or k_{pop}, the search space of the optimal LUR will be expanded, and consequently a shorter average length can be achieved. Another interesting observation is that the average length of LURs on RT here is significantly shorter than the results in Table I. A possible reason is that the deep learning models based on neural networks are more robust to perturbations than the traditional machine learning models, so a shorter LUR is enough to achieve a high accuracy.

In terms of test accuracy, LURLocator always outperforms alternative solutions by a large margin. The results can be further improved if we increase the threshold τ. The *Rand* baseline achieves a remarkable result (over 90%) on DBP. The intuitive explanation is that the decision boundary to segment the data points on DBP is extremely robust (Refer to Figure 1 where X_2 safely locates inside the decision boundary), so that slight perturbations can hardly change the prediction. In fact, the accuracy achieved by *Rand* will be decreased if we increase the number of synonyms in the perturbation distribution intentionally. From the results, it can be concluded that LURs excel at providing sufficient justifications, while existing feature-additive explanatory methods have trouble selecting such a set of words.

D. Conclusions

In this paper, we shed light on the problem of model interpretabilty for text classification and introduce *locally universal rules (LURs)*, which are a minimal set of input features sufficient to rationalize the instance-wise predictions. We propose *LURLocator* based on genetic algorithms to identify the optimal LUR. Extensive experiments are performed on a variety of models and datasets to evaluate the proposed algorithm as well as the generated LURs. The results show that our algorithm leads to better performance.

REFERENCES

[1] V. D. S. Silva, A. Freitas, and S. Handschuh, "On the semantic interpretability of artificial intelligence models," *CoRR*, vol. abs/1907.04105, 2019.

[2] B. Carter, J. Mueller, S. Jain, and D. K. Gifford, "What made you do this? understanding black-box decisions with sufficient input subsets," in *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, 2019, pp. 567–576.

[3] O. Camburu, E. Giunchiglia, J. Foerster, T. Lukasiewicz, and P. Blunsom, "Can I trust the explainer? verifying post-hoc explanatory methods," *CoRR*, vol. abs/1910.02065, 2019.

[4] 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*, 2016, pp. 1135–1144.

[5] G. Plumb, D. Molitor, and A. S. Talwalkar, "Model agnostic supervised local explanations," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018, pp. 2520–2529.

[6] S. Serrano and N. A. Smith, "Is attention interpretable?" in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy*, 2019, pp. 2931–2951.

[7] W. J. Murdoch, P. J. Liu, and B. Yu, "Beyond word importance: Contextual decomposition to extract interactions from lstms," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*, 2018.

[8] A. Jacovi, O. S. Shalom, and Y. Goldberg, "Understanding convolutional neural networks for text classification," in *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018*, 2018, pp. 56–65.

[9] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K. Müller, "How to explain individual classification decisions," *J. Mach. Learn. Res.*, vol. 11, pp. 1803–1831, 2010.

[10] M. Denil, A. Demiraj, and N. de Freitas, "Extraction of salient sentences from labelled documents," *CoRR*, vol. abs/1412.6815, 2014.

[11] J. Li, W. Monroe, and D. Jurafsky, "Understanding neural networks through representation erasure," *CoRR*, vol. abs/1612.08220, 2016.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, 2018, pp. 1527–1535.

[13] T. Lei, R. Barzilay, and T. S. Jaakkola, "Rationalizing neural predictions," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, 2016, pp. 107–117.

[14] D. Nguyen, "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)*, 2018, pp. 1069–1078.

[15] K. G. Murty and S. N. Kabadi, "Some np-complete problems in quadratic and nonlinear programming," *Mathematical Programming*, vol. 39, no. 2, pp. 117–129, 1987.

[16] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, University of Michigan, USA*, 2005, pp. 115–124.

[17] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Annual Conference on Neural Information Processing Systems*, 2015, pp. 649–657.