Explaining a Deep Learning Model for Aspect-Based Sentiment Classification Using Post-hoc Local Classifiers

Vlad Miron[1], Flavius Frasincar[1][0000−0002−8031−758X](✉), and Maria Mihaela Truşcă[2]

[1] Erasmus University Rotterdam, Burgemeester Oudlaan 50, 3062 PA Rotterdam, the Netherlands
[2] Bucharest University of Economic Studies, 010374 Bucharest, Romania
vlad.ioan.miron@gmail.com, frasincar@ese.eur.nl, maria.trusca@csie.ase.ro

Abstract. Aspect-Based Sentiment Classification (ABSC) models are increasingly utilised given the surge in opinionated text displayed on the Web. This paper aims to explain the outcome of a black box state-of-the-art deep learning model used for ABSC, LCR-Rot-hop++. We compare two sampling methods that feed an interpretability algorithm which is based on local linear approximations (LIME). One of the sampling methods, SS, swaps out different words from the original sentence with other similar words to create neighbours to the original sentence. The second method, SSb, uses SS and then filters its neighbourhood to better balance the sentiment proportions in the localities created. We use a 2016 restaurant reviews dataset for ternary classification and we judge the interpretability algorithms based on their hit rate and fidelity. We find that SSb can improve neighbourhood sentiment balance compared to SS, reducing bias for the majority class, while simultaneously increasing the performance of LIME.

Keywords: aspect-based sentiment classification · explainable artificial intelligence · sampling methods

1 Introduction

In today's world, the amount of opinionated text shared on the Web is growing at unprecedented speeds. All of this text can be very valuable in gauging the public's perception of a given topic and thus allowing a brand to learn more about their customers to improve an existing product or service [3]. Sentiment analysis [6] has also been shown to be useful for consumers trying to make more informed decisions, allowing them to evaluate a given product or service more holistically, based on the aggregated opinions of many past customers [17]. A subfield of sentiment analysis is Aspect-Based Sentiment Analysis (ABSA) where the sentiment is computed with respect to aspects of the entity of interest [15]. ABSA comprises two steps: Aspect Detection (AD), which determines

the aspects [18], and Aspect-Based Sentiment Classification (ABSC), which determines the sentiment related to the previously discovered aspects [1]. We focus on ABSC. The main downfall of deep learning models for ABSC is their black box nature. Interpretability algorithms aim to solve this issue.

This paper aims to explore a state-of-the-art deep learning model, used for ABSC using one interpretability technique and two sampling methods on a restaurant reviews dataset. [19] introduces the Hybrid Approach for ABSA using BERT embeddings (HAABSA++), which is the basis of our work through its back up algorithm, LCR-Rot-hop++ (Left-Centre-Right separated neural network with Rotatory attention repeated for a number of iterations). We focus on LCR-Rot-hop++ due to its good performance.

We group interpretability algorithms by the taxonomy proposed by [9]. We split the algorithms into intrinsic or post-hoc, and local or global. We analyse post-hoc algorithms because an intrinsically interpretable Deep Neural Network (DNN) would suffer greatly in terms of accuracy. Then, we aim for local interpretability algorithms as our main goal is to explain to an individual the result produced by the model. In use cases such as these, a global approach may offer an interpretation which is too vague or even not applicable to the individual requesting an explanation. Thus, we employ post-hoc, local interpretability algorithms.

Furthermore, the chosen interpretability algorithm should require creating a local neighbourhood of instances around the prediction it aims to explain. This allows it to cater better to the instances explained locally, as methods that do not use a local neighbourhood have difficulties in gaining valuable insight in individual outcomes. Additionally, the sampling methods should feed the interpretability algorithm with roughly equal proportions of class instances, in our case, negative, neutral, and positive sentiment opinions. This is important as otherwise, the DNN becomes biased towards the majority class. Therefore, using a local neighbourhood and making sure to balance the sentiment proportions of its instances should increase the performance of the interpretability algorithm.

The central research question is thus *"Which sampling method and post-hoc, local classifier configuration is best suited to increase the interpretability of LCR-Rot-hop++?"*.

We consider one interpretability technique that satisfies our desired properties: Local Interpretable Model-Agnostic Explanations (LIME) [12]. We introduce two sampling methods that are used by the interpretability algorithm: Similarity-based Sampling (SS), which is similar to the method introduced by [12], and Similarity-based Sampling with balanced sentiment (SSb), which is an extension of SS. Our goal in this paper is to Sample and Interpret LCR-Rot-hop++ (SI-LCR-Rot-hop++), in order to gain insight into the model predictions.

SS works by changing a given percentage of the words in the initial sentence x with other words in the embedding space that are similar (i.e., words between which the distance in the embedding space is relatively small). SSb filters the neighbours of x based on the sentiment they show when being fed into LCR-

Rot-hop++, aiming to get as close as possible to creating neighbourhoods that are of equal size for each of the three labels.

Our first contribution stands in increasing the class balance using SSb and a tuned version of SSb. The second contribution stands in improving the performance of LIME, especially as we perform sensitivity analysis on a key hyperparameter of our sampling methods. We gain a better understanding of what factors have a positive impact on LIME, allowing us to optimise its results, simultaneously improving its neighbourhood class balance and performance. The code for our paper is written in Python 3.6.5 and made publicly available on GitHub, `https://github.com/VladMiron00/SI-ABSA`.

This paper is structured as follows. Section 2 discusses the development of our base model, as well as our interpretability technique, positioning them in the literature. Section 3 shows the characteristics of our data. Section 4 presents in more detail the base deep learning model, sampling approaches, interpretability algorithm, and evaluation measures used. Section 5 discusses the results obtained by our sampling approaches, interpretability algorithm, and performs sensitivity analysis on an influential hyperparameter of our sampling methods to improve the class balance and performance of our interpretability algorithm. Lastly, Sect. 6 gives our conclusion and suggestions for future work.

2 Related Works

This section showcases the latest developments regarding the topics researched. Subsection 2.1 reviews the current literature surrounding ABSC. Subsection 2.2 presents adjacent work regarding the black box interpretability algorithms used.

2.1 Aspect-Based Sentiment Classification

Aspect-Based Sentiment Analysis (ABSA) aims to capture the sentiment of aspects discussed in text [15]. It includes the steps of Aspect Detection (AD) responsible for finding the discussed aspects [18] and Aspect-Based Sentiment Classification (ABSC) responsible for determining the sentiment concerning the previously discussed aspects [1]. We focus on ABSC and assume the aspect to be given.

In previous work, [16] took an ontology approach to ABSC. Observing how the ontological approach failed from time to time in detecting the sentiment, researchers proposed using a deep learning model as a backup, employing a hybrid approach to solve the shortcomings of ABSC, also known as the Hybrid Approach for ABSA (HAABSA) [20]. The first version of what would become the backup DNN of HAABSA was introduced by [21] under the name of LCR-Rot, which stands for Left-Centre-Right separated neural network with Rotatory attention. This DNN splits the sentence into the target (which is the centre or the aspect) and its left and right contexts, assigning weights to the words in the different parts of the sentence based on how important they are with regard to the target.

Then, [20] applied an improved version of LCR-Rot, LCR-Rot-hop, which iterates over the rotatory mechanism of LCR-Rot multiple times to ensure consistency. The latest HAABSA extension is presented by [19] as LCR-Rot-hop++, which improves LCR-Rot-hop by adding a hierarchical attention layer to DNN. Furthermore, LCR-Rot-hop++ changes the context-independent word embeddings of LCR-Rot-hop to the context-dependent BERT embeddings. This allows LCR-Rot-hop++ to better deal with word polysemy.

2.2 Black Box Interpretability Algorithms

Based on the current literature we note that the interpretability of deep learning attention models for ABSC has not been studied to a large extent [2], [4], [14].

That said, diagnostic classification has been tried on HAABSA, namely with the contribution of [8]. This implementation used LCR-Rot-hop as the backup for the ontological approach. Its findings show that context representation is the main factor in determining the relations between the target and other words in the sentence. Furthermore, the performance of the model determining the sentiment value shown concerning the target does not greatly vary along with the 10 iterations (hops) that were implemented. [5] conducts a similar study and obtain comparable results, using LCR-Rot-hop++ instead of LCR-Rot-hop.

An interpretability algorithm which satisfies our desired characteristics is the Local Agnostic attribute Contribution Explanation (LACE) [10]. This rule-based algorithm brings interpretability to a prediction by analyzing the joint effect of subsets of features that can be formed out of the initial feature pool on the model outcome. This model is local because the prediction to be explained is in the vicinity of the feature value subsets chosen. We eliminate LACE from our analysis because it uses an ad-hoc method to generate its classifiers.

Another interpretability algorithm that fits our desired properties was introduced by [13], being named Anchor. Anchor is a rule-based method that works by selecting a set of instances in the neighbourhood of the instance we want to explain. It assesses which features in the set of local instances are most influential. We eliminate Anchor from our analysis because of its slow convergence.

[7] proposes SHAP, a method that works using the principles of cooperative game theory. SHAP calculates a coefficient that shows how important a feature is for each subset of features it can be included in and then averages out the importance coefficient over all possible subsets. We exclude this approach from our analysis because it does not make use of a neighbourhood sampling method.

3 Data

The data used in this paper follows the preprocessing rules and manipulations of [19], using the same *SemEval 2016 Task 5 Subtask 1 Slot 3* dataset [11].

The descriptive statistics of the sentences remaining after preprocessing are presented in Table 1, where train and test data represent the basis of our analysis. The sentences which could not be classified by the ontology approach are

Table 1: Sentiment labels within the filtered SemEval 2016 datasets.

Dataset	Negative		Neutral		Positive		Total	
	Freq.	%	Freq.	%	Freq.	%	Freq.	%
Train data	489	26	72	3.8	1319	70.2	1880	100
Test data	135	20.8	32	4.9	483	74.3	650	100
Rem. test data	82	33	22	8.9	144	58.1	248	100
Used test data	8	32	2	8	15	60	25	100

collected in *"remaining test data"*, which is the dataset that LCR-Rot-hop++ should run on as a backup for the ontology. To avoid long run times (as each instance requires its own local model) we create a smaller dataset that emulates *"remaining test data"*. Thus, *"used test data"* aims to have similar sentiment proportions to *"remaining test data"*, while trimming down the number of instances from 248 to 25. The dataset *"used test data"* is fed in all LCR-Rot-hop++ runs.

As the positive class is present in a clear majority of instances, it causes the ontological approach to disproportionately classify incorrectly the neutral and negative instances within the test sample. This created bias for the majority class explains why neutral and negative sentiment sentences increase in proportion within *"remaining test data"*.

4 Methodology

In this section, we present the methods behind the analysed ABSC model. Subsection 4.1 presents this paper's chosen model for ABSC, LCR-Rot-hop++. We explore LCR-Rot-hop++ using different sampling methods applied on the interpretability algorithm. Subsection 4.2 shows the sampling methods used by our interpretability algorithm. Subsection 4.3 discusses the used interpretability algorithm. Lastly, Subsect. 4.4 presents evaluation measures for the sampling methods and interpretability algorithm.

4.1 LCR-Rot-hop++

As [19] describes, LCR-Rot-hop++ splits the input (a sentence), into three parts: the left context, the centre, and the right context. The centre is the aspect target value of the sentence, having T words, where T may be larger or equal to 1. A sentence may look like *"this restaurant is amazing"*, where *"this"* is the left context, *"restaurant"* is the target, and *"is amazing"* is the right context. These words are then embedded using BERT, a method based on transformers.

Next, three Bidirectional Long Short-Term Memory (Bi-LSTM) models are used, each Bi-LSTM corresponding to a different part of the sentence. The system consists of four parts: the left and right contexts which are used to obtain one target representation for the left context and another target representation for the right context, and the two target representations which are used to produce the left and right context representations. The rotatory attention mechanism

is continuously used to change the representation of the different parts of the sentence for the hierarchical attention mechanism, which takes them as input and weights them. After attaining these four weighted representations, LCR-Rot-hop++ proceeds to feed them back into the rotatory attention mechanism and weigh them again using the hierarchical attention mechanism for the desired number of iterations.

After the last rotation, the final representations of the four parts are used as input for the Multi-Layer Perceptron (MLP). The MLP uses a softmax function to output the aspect-level sentiment predictions.

4.2 Sampling Methods

As the "S" in SI-LCR-Rot-hop++ suggests, the sampling methods are a key part of our paper. Their existence is required for the approach of LIME, as it is based on creating a local neighbourhood around the prediction $x \in X$ (X is the set of instances) that it aims to explain. The neighbourhood it creates is denoted as Z_x, where $z \in Z_x$ is a perturbed sample, being similar to x in many regards, except for a couple of features which are changed. The feature changes refer to swapping one or more words f from the original sentence x with other words that are similar out of the set of F features which compose the used *SemEval-2016* restaurant reviews datasets. We create local neighbours by changing words (features) only in the left or the right context of the original sentence x.

Since a sentence x is originally represented as a sequence of word embeddings, we change its format to input it into our algorithms. We achieve this by using the modified x' instances, which are a binary representation of x indicating which features $f \in F$ are contained within the sentence x, $x' \in \{0,1\}^{|F|}$.

Algorithm 1 shows how interpretability algorithm m_x (in our case LIME) functions for any given instance $x \in X$. The explanation that algorithm m_x provides for x is denoted $\xi_m(x)$. The neighbourhood size Z_x is denoted as n_x.

Algorithm 1 Using m_x to explain prediction of $b(x)$

Arguments of method: Black box model b, interpretation model m_x, instance $x \in X$, desired neighbourhood size n_x
$Z'_x \leftarrow \emptyset$
for $i \in \{1, 2, ..., n_x\}$ **do**
  $z'_i \leftarrow apply\_sampling\_method\_on(x)$
  $Z'_x \leftarrow Z'_x \cup z'_i$
end for
$\xi_m(x) \leftarrow m_x(Z'_x, b)$
return $\xi_m(x)$

The size of the neighbourhoods created by the Similarity-based Sampling method (SS) and the Similarity-based Sampling method with balanced sentiment (SSb) differ, as SS creates neighbourhoods of 5000 local instances, while SSb trims down the initial, larger, neighbourhood created by SS to a balanced neighbourhood of 150 perturbations for each instance x.

4.2.1 Similarity-based Sampling Method. LIME needs to have a neighbourhood of local instances generated around an input instance. The Similarity-based Sampling (SS) method is similar to that of [12]. It works by analysing the embedding space for each word w_1 in sentence x and finding another similar word. The first step in this task is assigning a POS tag to the elements of set F, consisting of all the different words in the *SemEval-2016* datasets. The possible tags are noun, verb, adjective, adverb, adposition, and determiner. Assuming the same POS tag for w_1 and w_2, the distance between the 2 words is calculated by the formula:

$$D(w_1, w_2) = 1 - \frac{w_1 \cdot w_2}{||w_1||||w_2||}. \tag{1}$$

Algorithm 2 generates a local instance z' for any $x \in X$ via SS. We transform the local instance z to attain the binary representation of the instance, z'.

Algorithm 2 Using SS to generate a local instance z' for instance x ($apply\_SS(x)$)

Arguments of method: instance $x \in X$
$z \leftarrow \emptyset$
for $w_1 \in x$ **do**
  $change\_boolean \leftarrow change\_function(change\_probability)$
  if $change\_boolean$ is $True$ **then**
    $distances \leftarrow \emptyset$
    for $w_2 \in F$ **do**
      $distances \leftarrow distances \cup D(w_1, w_2)$
    end for
    $top\_n\_words \leftarrow pick\_top\_n\_words(distances)$
    $z \leftarrow z \cup picking\_algorithm(top\_n\_words)$
  else
    $z \leftarrow z \cup w_1$
  end if
end for
$z' \leftarrow transform(z)$
return z'

Algorithm 2 changes the words in the left and the right contexts of the original instance x with other words in the dataset which are similar. It iterates through each word $w_1 \in x$, deciding if it has to replace said word or not.

A higher *change_probability* suggests that more words in any given sentence x are going to be replaced with other words from the dataset. If the decision is to not change the current word in the sentence, the perturbed instance z receives the original word w_1. If the decision is to change the current word, we start this process by calculating the distances between word w_1 and all of the other words in its embedding space, words contained in F. Note that the original word w_1 is included in the pool of words which may be chosen for its replacement, meaning that a *change_probability* of 100% does not guarantee that all words in all perturbations are different to the ones in the original instance.

Next, we create the ranking *top_n_words*, where words that are closer to w_1 rank higher. We need to pick out one word from *top_n_words*. This ranking shows which words are closest to w_1, assigning a probability to each word.

Said probability is decided by the *picking_algorithm*, which is set for a weighted pick. The weight is given based on the ranking, where words closer to the first position in the ranking receive a higher weight and are thus more likely to be picked. We consider the word chosen a suitable replacement for w_1 in sentence x as it is contextually similar and grammatically identical to w_1.

After iterating through the words in x we attain the perturbed sentence z, which we transform into binary format and return as z', that is output.

4.2.2 Similarity-based Sampling Method with Balanced Sentiment.

Similarity-based Sampling method with balanced sentiment (SSb) builds on top of SS by aiming to solve its main issue. Because SS creates local instances for $x \in X$ by replacing words in x with other similar words, the created neighbours are bound to be similar to x, and thus to get the same label as the original instance. Since the original dataset contains a clear majority of positive labelled instances, this sentiment distribution is likely to carry over to the local instances that SS creates, possibly affecting the performance of the interpretability algorithms due to the bias that class imbalance creates for the majority class.

SSb aims to solve this issue and provide LIME with a more balanced set of instances to train on, achieving this by taking into account the sentiment of the neighbours created when choosing whether to keep them or not (Algorithm 3).

Algorithm 3 Using SSb for LIME to create a balanced neighbourhood $Z_x^{b\prime}$ for instance x

Arguments of method: instance $x \in X$
$Z_x \leftarrow \emptyset$
$Z_x^b \leftarrow \emptyset$
$M \leftarrow 0$
$SS\_neigh\_size \leftarrow 5000$
$SSb\_neigh\_size \leftarrow 150$
while $M < SS\_neigh\_size$ **do**
  $z' \leftarrow apply\_SS(x)$
  $z \leftarrow transform(z')$
  $Z_x \leftarrow Z_x \cup z$
  $M \leftarrow M + 1$
end while
$neighs\_sentiments \leftarrow get\_sentiments(Z_x)$
$chosen\_perturb \leftarrow get\_balanced\_perturb(neighs\_sentiments, SSb\_neigh\_size)$
for $z \in Z_x$ **do**
  if $z \in chosen\_perturb$ **then**
    $Z_x^b \leftarrow Z_x^b \cup z$
  end if
end for
$Z_x^{b\prime} \leftarrow transform(Z_x^b)$
return $Z_x^{b\prime}$

Z_x^b is the balanced neighbourhood of instance x. This neighbourhood is obtained by filtering the original, larger, neighbourhood of Z_x. Z_x is created using

Algorithm 2 (called by $apply\_SS(x)$) and it contains SS_neigh_size (5000 in this paper) perturbations. In SSb, we trimmed down the neighbourhood size of SS to a balanced, smaller, neighbourhood of SSb_neigh_size (150) perturbations.

We obtain the unbalanced neighbourhood of SS_neigh_size instances by iteratively applying the SS algorithm and saving its output. Then, we run LCR-Rot-hop++ to obtain the sentiment predictions for each perturbation created for instance x, using the function $get\_sentiments(Z_x)$. We store the results in a counter vector, $neighs\_sentiments$, which shows how many perturbations of each sentiment correspond to an instance x. This vector is then fed into the $get\_balanced\_perturb()$ function along with the balanced neighbourhood size to determine how many perturbations of each sentiment we should keep. Ideally, for an original neighbourhood of 5000 instances and a balanced neighbourhood of 150 instances, we should have at least 50 perturbations of each of the three sentiments in the original neighbourhood. This would allow us to obtain a balanced neighbourhood of roughly 33.3% of each sentiment.

Last, we iterate through the original set of perturbations Z_x and pick out based on the indexes saved in $chosen\_perturb$ the local instances that we add to the balanced set Z_x^b. We return this set of neighbours in binary format.

4.3 LIME

This section discusses the second component of SI-LCR-Rot-hop++, our interpretability method, namely LIME [12]. LIME is a popular model-agnostic interpretability algorithm which samples instances in the neighbourhood of x and applies a linear approximation to said instances to attain a model that performs similarly on a local level to the original black box.

For our paper, we are dealing with a ternary classification problem with the classes $K = \{-1, 0, 1\}$, corresponding to negative, neutral, and positive sentiments, respectively. The underlying interpretable model g is log-linear, as the method used is the multinominal logistic regression. It works by training 3 ($|K|$) binary classification models $g^{(k)}(x)$ for each instance $x \in X$, corresponding to the 3 combinations possible when you consider one of the $|K|$ classes, k, as the benchmark and you group the other 2 classes under k^c. Its formula is:

$$g^{(k)}(x') = ln(Pr[b(x) = k|x]) = \beta_0^{(k)} + \sum_{j \in F} \beta_j^{(k)} x'_j - ln(L). \tag{2}$$

The binary form representation of the instance we aim to explain is x'. The interpretable model computes the natural logarithm of the probability that the black box model classifies x as the sentiment k ($k \in K$). $\beta_j^{(k)}$ with $j \in F$ represents the marginal effect of feature j on the binary sentiment classification of x. L is the normalization term, explicited as $\sum_{k \in K} e^{\beta^{(k)} x'}$ with $\beta^{(k)}$ denoting the vector that contains the set of all coefficients $\beta_j^{(k)}$ with $j = \{0, 1, ..., |F|\}$.

We can draw interpretability insight from these marginal effects, with the mention of keeping the number of marginal effects chosen limited. We select a

maximum of S features with the highest influence to be included in the interpretability model. The influence of feature j (e_j) is calculated as the sum of the absolute marginal effects of feature j on classes $k \in K$, $e_j = \sum_k |\beta_j^{(k)}|$ with $k = 1, 2, 3$.

The interpretation brought by LIME stands in the set S of marginal effects, corresponding to the top $|S|$ out of $|F_x|$ most influential features of F_x (F_x being the set of words in sentence x). To determine this set of marginal effects we apply a weighted multinominal logistic regression trained on Z_x according to Equation 2. The neighbourhood of x, Z_x, is generated via SS or SSb and the weights attributed to the local instances included in the neighbourhood depend on the proximity of the perturbations z' to x, π_x, which is defined by an exponential kernel, $\pi_x(z) = exp(-D(x,z)^2/\sigma^2)$. σ is the width of the kernel and $D(x,z)$ is the distance function applied on the word embeddings of instances x and z.

4.4 Performance Evaluation

Subsubsection 4.4.1 presents an instance evaluation measure for the sampling methods picked and Subsubsect. 4.4.2 shows evaluation measures for LIME.

4.4.1 Measure for the Sampling Methods. As previously discussed, achieving balanced labels within the neighbourhoods created using SS and SSb is important in training an unbiased model.

One way to quantify the degree of overall balance within our neighbourhoods is by computing the entropy, calculated as follows:

$$Entropy = -\sum_k p(k) \log_2 p(k), \tag{3}$$

where $p(k)$ is the proportion of sentences labelled as $k \in K$. The higher the entropy, the better the balance of sentiments, as the highest entropy value is achieved when roughly 33.3% of the sentences are of each sentiment.

4.4.2 Measures for LIME. The first performance measure is the *hit rate*, which shows how often the interpretable model m_x, trained in the neighbourhood of instance x, and the black box model b give the same prediction for a given instance $x \in X$. It is calculated as the number of times b and m_x match their predictions for $x \in X$ over the cardinality of X (X being the set of sentences we create neighbourhoods for). This indicates that a larger hit rate is better.

Another quantitative performance measure is the *fidelity*, which shows how often the interpretable model m_x and the black box model b give the same prediction for a local instance in Z_x. It is calculated as the number of times b and m_x match their predictions for all instances x and their perturbations over the cardinality $|Z_x| * |X|$. A larger fidelity value is better.

A high value is needed for both the hit rate and fidelity to ensure that the interpretability model is true to the original black box. To judge the balance

Table 2: Sentiment proportions given the sampling methods.

SI Comb.	Negative		Neutral		Positive		Entropy
	Freq.	%	Freq.	%	Freq.	%	Value
SS for LIME	42737	34.2	410	0.3	81853	65.5	0.95
SSb for LIME	1219	32.5	221	5.9	2310	61.6	**1.2**

between the hit rate and the fidelity we propose to use the harmonic mean of the two. This measure is used as a proxy for the overall performance of the configuration ran, with a higher value being better. The formula for the harmonic mean in our case is:

$$Harmonic\ Mean = \frac{2}{\frac{1}{Hit\ Rate} + \frac{1}{Fidelity}}. \tag{4}$$

5 Results

In this section we present the results of our proposed method evaluation. Subsection 5.1 evaluates our proposed sampling methods with regards to the sentiment proportions created. Subsection 5.2 compares the hit rate and fidelity achieved by LIME under different sampling methods. Subsection 5.3 performs sensitivity analysis on a key hyperparameter of our sampling methods to improve the tuning of SSb and the performance of LIME using SSb.

5.1 Sampling Methods

The results in Table 2 correspond to LCR-Rot-hop++ classification aggregated over the 25 instances $x \in X$ from *"used test data"* and their respective neighbours. The sampling methods need to be run for LIME, which gets fed 5000 local instances for each sentence x for SS and 150 local instances for SSb. The *change_probability* is set to 50%.

Table 2 shows how SSb impacted the proportions of sentiment labels for LIME compared to SS. The number of negative and positive labelled sentences gets marginally decreased, while the neutral category is increased almost twenty times in frequency (from 0.3% to 5.9%). Thus, the entropy increases from 0.95 under SS to 1.2 under SSb, showing the overall improvement in label balance.

5.2 Interpretability Algorithms

To find the best sampling method configuration for LIME, we measure the hit rate and fidelity, as shown in Table 3. We calculate the harmonic mean to judge the overall performance of a given configuration.

Looking at the harmonic mean of the hit rate and the fidelity, we notice SSb performing better than SS. The reduced sample used, containing 25 out of the 248 sentences, may be the reason for values of 100% for the hit rate.

Table 3: Hit rate and fidelity of LIME given the sampling methods.

SI Combination	Hit Rate %	Fidelity %	Harmonic Mean %
LIME with SS	100	90.1	94.8
LIME with SSb	100	91.1	**95.4**

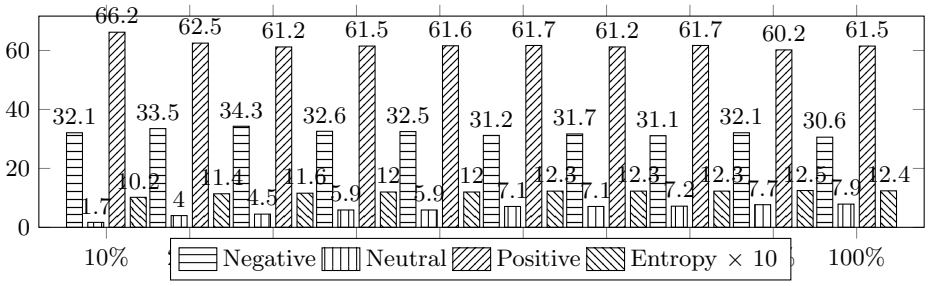

Fig. 1: Sensitivity analysis of SSb for the sentiment distribution under LIME, x-axis shows the value of *change_probability* in presented run.

5.3 Sensitivity Analysis

In the previous sections, we find that SSb is a useful extension, being able to improve both the class balance, measured by the entropy, and the hit rate and fidelity judged jointly using the harmonic mean. We are now interested if we are able to bring a further beneficial effect to our interpretability algorithm by iteratively altering an influential hyperparameter, *change_probability*, with the intent of finding out what value brings the best results. We use *change_probability* with values from 10% to 100% with 10% increments for LIME with SSb. We do not perform this analysis for SS because we have observed how it underperforms both in terms of class balance and performance measures compared to SSb.

Figure 1 shows an increasing trend for the class balance of LIME as the *change_probability* shifts from 10% to 90%, where it achieves its peak entropy of 1.25. The results up to the run using a 90% *change_probability* show that altering the original sentence x to a larger extent leads to neighbouring sentences that are more likely to receive labels different from the label of x (more diverse).

The unexpected result is in the last run, using a probability of changing the words of 100%, where we notice a slight decrease in the balance of sentiments compared to the run using 90%, as entropy drops from 1.25 to 1.24. It seems that as the hyperparameter of interest reaches high values, the balance of sentiments becomes a matter of chance. Therefore, it is possible that as the *change_probability* exceeds 90%, the impact on the balance of sentiments becomes unpredictable, as it may improve or not across runs or datasets.

Figure 2 shows the impact of the varying *change_probability* on the fidelity and hit rate of LIME, measures which are summed up using the harmonic mean. There is a clear up trend in the performance of the models as the mentioned

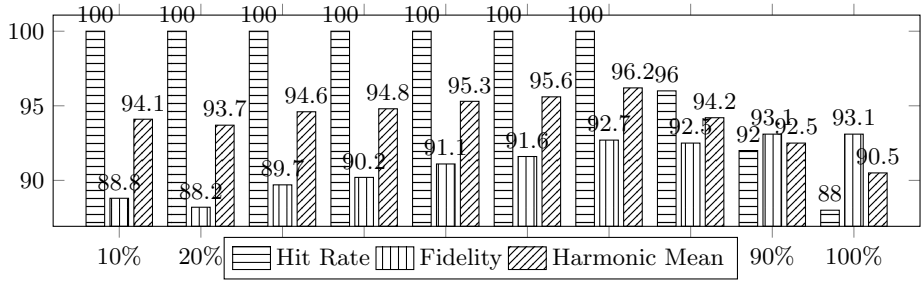

Fig. 2: Sensitivity analysis of SSb for the quantitative measures of LIME, x-axis shows the value of *change_probability* in presented run.

hyperparameter increases in value. Thus, a more diverse neighbourhood improves performance. A peak harmonic mean of 96.2% is reached for a *change_probability* of 70%.

One interesting observation regarding Fig. 2 concerns the fact that a trade off between the hit rate and the fidelity appears as the probability to change a feature increases. This is to be expected as higher probabilities of changing a word imply a more diverse set of neighbours in terms of the range of vocabulary. Thus, as the sentences in the neighbourhood Z_x start to differ more from x, LIME gets less trained to recognize and correctly classify x, reducing the hit rate. At the same time, LIME gets used to training on a broader range of neighbours, recognizing the varying sentiments they have, leading to an increase in fidelity.

Another interesting observation about Figure 2 is that the benefit of increasing the word replacement probability ceases to exist as the probability to change a word reaches or exceeds 80%. Although more word replacements create more diverse sentences and more balanced label proportions, from about 80% word replacement probability onward, the perturbed sentences start to not make as much grammatical or contextual sense as before.

For example, given a word replacement probability of 100%, the sentence *"the owner is belligerent to guests that have a complaint."* (where *"owner"* is the target) turned in one of the perturbations into *"an owner was belligerent about drinks which use the disappointment."*. In contrast, for a change probability of 50%, one perturbation looked like *"this owner is belligerent to people that make a request."*. This is just an anecdote, but it goes to show that using high feature changing probabilities risks creating neighbours which are not representative of a real review left by a customer, reducing the performance obtained by the interpretability algorithm.

To conclude on our results concerning LIME, it seems that in our runs a *change_probability* of 70% is optimal, as it reaches the best value for the harmonic mean, while drastically improving sentiment balance compared to SS. To be exact, the class balance measured by the entropy increases from 0.95 under SS to 1.23 under tuned SSb. Further, the harmonic mean of the hit rate and the fidelity increases from 94.8% under SS to 96.2% under tuned SSb.

6 Conclusion

In this work, we propose SSb, an extension of a sampling method (SS) to improve the balance of sentiments of the sentences that the interpretability algorithm (LIME) uses. We further improve the performance of LIME by performing sensitivity analysis on a key hyperparameter for the sampling methods, *change_probability*. We measure sentiment balance using the entropy and the model performance using the harmonic mean of the hit rate and fidelity. We find optimal results by setting the *change_probability* at 70% when running SSb. This configuration yields an increase in the entropy (and thus in the class balance) from a value of 0.95 under SS to 1.23 under tuned SSb. The harmonic mean increases from 94.8% under SS to 96.2% under tuned SSb. Thus, we manage to find a configuration that improves both the class balance and the model performance simultaneously for LIME.

A possible future research opportunity lies in further improving the method of balancing sentiment proportions of the perturbations. This may be achieved by using both BERT embeddings and sentiment-aware BERT embeddings. The sampling method may replace a word in sentence x only with another word that is close in the BERT embedding space (being a contextually feasible replacement) and far in the sentiment-aware BERT embedding space (increasing the chance that the replacement will change the original sentiment label attributed to the sentence). This way, we will build on purpose rather than by chance perturbations which are not only suitable neighbours given the context but also diverse in sentiment.

References

1. Brauwers, G., Frasincar, F.: A survey on aspect-based sentiment classification. ACM Computing Surveys **55**(4), 65:1–65:37 (2023)
2. Du, M., Liu, N., Hu, X.: Techniques for interpretable machine learning. arXiv preprint arXiv:1808.00033 (2018)
3. Godbole, S., Roy, S.: Text classification, business intelligence, and interactivity: Automating c-sat analysis for services industry. In: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008). pp. 911–919. ACM (2008)
4. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Pedreschi, D., Giannotti, F.: A survey of methods for explaining black box models. arXiv preprint arXiv:1802.01933 (2018)
5. Kunal, G., Frasincar, F., Truşcă, M.M.: Explaining a deep neural model with hierarchical attention for aspect-based sentiment classification. In: 22nd International Conference on Web Engineering (ICWE 2022). LNCS, vol. 13362, pp. 268–282. Springer (2022)
6. Liu, B.: Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge university press, 2nd edition (2020)
7. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In 31st Annual Conference on Neural Information Processing Systems (NIPS 2017) **30**, 4765–4774 (2017)

8. Meijer, L., Frasincar, F., Truşcă, M.M.: Explaining a neural attention model for aspect-based sentiment classification using diagnostic classification. In: 36th ACM/SIGAPP Symposium on Applied Computing (SAC 2021). pp. 821–827. ACM (2021)
9. Molnar, C.: Interpretable machine learning. In: Second International Workshop in eXplainable Knowledge Discovery in Data Mining (XKDD 2020). CCIS, vol. 1309. Springer (2020)
10. Pastor, E., Baralis, E.: Explaining black box models by means of local rules. In: 34th ACM/SIGAPP Symposium on Applied Computing (SAC 2019). pp. 510–517. ACM (2019)
11. Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, y., Qin, B., de Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., María Jiménez-Zafra, S., Eryiğit, G.: SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In: 10th International Workshop on Semantic Evaluation (SemEval 2016). pp. 19–30. ACL (2016)
12. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016). pp. 1135–1144. ACM (2016)
13. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018). vol. 32, pp. 1527–1535. AAAI Press (2018)
14. Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., Zhong, C.: Interpretable machine learning: Fundamental principles and 10 grand challenges. Statistics Surveys **16**, 1–85 (2022)
15. Schouten, K., Frasincar, F.: Survey on aspect-level sentiment analysis. IEEE Transactions on Knowledge and Data Engineering **28**(3), 813–830 (2016)
16. Schouten, K., Frasincar, F., de Jong, F.: Ontology-enhanced aspect-based sentiment analysis. In: 17th International Conference on Web Engineering (ICWE 2017). LNCS, vol. 10360, pp. 302–320. Springer (2017)
17. Singla, Z., Randhawa, S., Jain, S.: Statistical and sentiment analysis of consumer product reviews. In: 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT 2017). pp. 1–6. IEEE (2017)
18. Truşcă, M.M., Frasincar, F.: Survey on aspect detection for aspect-based sentiment analysis. Artificial Intelligence Review **56**(5), 3797–3846 (2023)
19. Truşcă, M.M., Wassenberg, D., Frasincar, F., Dekker, R.: A hybrid approach for aspect-based sentiment analysis using deep contextual word embeddings and hierarchical attention. In: 20th International Conference on Web Engineering (ICWE 2020). LNCS, vol. 12128, pp. 365–380. Springer (2020)
20. Wallaart, O., Frasincar, F.: A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models. In: 16th Extended Semantic Web Conference (ESWC 2019). LNCS, vol. 11503, pp. 363–378. Springer (2019)
21. Zheng, S., Xia, R.: Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention. arXiv preprint arXiv:1802.00892 (2018)