The Good, the Bad, and the Explainer:
A Tool for Contrastive Explanations of Text Classifiers

Lorenzo Malandri¹,³, Fabio Mercorio¹,³*, Mario Mezzanizanica¹,³, Navid Nobani² and Andrea Seveso²,³

¹Dept of Statistics and Quantitative Methods, University of Milano Bicocca, Italy  
²Dept of Informatics, Systems and Communication, University of Milano Bicocca, Italy  
³CRISP Research Centre crispresearch.eu, University of Milano Bicocca, Italy  
{lorenzo.malandri, fabio.mercorio, mario.mezzanizanica, andrea.seveso}@unimib.it

Abstract

In the last few years, we have been witnessing the increasing deployment of machine learning-based systems, which act as black boxes whose behaviour is hidden to end-users. As a side-effect, this contributes to increasing the need for explainable methods and tools to support the coordination between humans and ML models towards collaborative decision-making. In this paper, we demonstrate ContrXT, a novel tool that computes the differences in the classification logic of two distinct trained models, reasoning on their symbolic representation through Binary Decision Diagrams. ContrXT is available as a pip package and API.

1 Introduction and Contribution

Consider a text classifier \(\psi_1\), retrained with new data and resulting into \(\psi_2\). The underlying learning function of the newly trained model might lead to outcomes considered as contradictory by the end users when compared with the previous ones, as the system does not motivate why the logic is changed. Hence, such a user might wonder "why do the criteria used by \(\psi_1\) result in class \(c_1\), but \(\psi_2\) does not classify on \(c\) anymore?". This is posed as a T-contrast question, namely, "Why does object \(A\) have property \(P\) at time \(t_1\), but property \(Q\) at time \(t_2\)?" [Miller, 2019; Van Bouwel and Weber, 2002].

Contribution. ContrXT (Contrastive eXplainer for Text classifier) is a tool that implements the approach we proposed in [Malandri et al., 2022a]. ContrXT computes model-agnostic global T-contrast explanations from any black box text classifiers. ContrXT, as a novelty, (i) encodes the differences in the classification criteria over multiple training phases through symbolic reasoning, and (ii) estimates to what extent the retrained model is congruent with the past. ContrXT is available as an off-the-shelf Python tool on Github, a pip package, and as a service through REST-API [Malandri et al., 2022c].

To date, there is no work that the authors are aware of that computes T-contrast explanation globally, as clarified by the most recent state-of-the-art surveys on XAI for supervised ML (see [Burkart and Huber, 2021; Mueller et al., 2019]).

*Contact Author. fabio.mercorio@unimib.it - mercorio.com

2 ContrXT in a Nutshell

ContrXT aims at explaining how a classifier changes its predictions through time. We describe the five building blocks composing ContrXT, as in Fig.1: (A) the two text classifiers, (B) their post-hoc interpretation using global, rule-based surrogate models, (C) the Trace step, (D) the eXplain step and, finally, (E) the generation of the final explanations through indicators and Natural Language Explanations (NLE).

(A) Text classifiers. ContrXT takes as input two text classifiers \(\psi_{1,2}\) on the same target class set \(C\), and the corresponding training datasets \(D_{1,2}\). As clarified in [Sebastiani, 2002], classifying \(D_i\) under \(C\) consists of \(|C|\) independent problems of classifying each \(d \in D_i\) under a class \(c_i\) for \(i = 1,\ldots,|C|\). Hence, a classifier for \(c_i\) is a function \(\psi: D \times C \rightarrow \mathbb{B}\) approximating an unknown target function \(\psi\).

Output: Two black-box classifiers on the same class set.

(B) Post-hoc interpretation. Following the study about ML post-hoc explanation methods [Burkart and Huber, 2021], one of the approaches consists in explaining a black box model globally by approximating it to a suitable interpretable model (i.e., the surrogate) solving the following:

\[
p_{\gamma}^* = \arg \max_{\gamma} \frac{1}{N} \sum_{x \in X} S(p_{\gamma}(x), \psi(x))
\]

where \(I\) represents a set of possible white box models to be chosen as surrogates, and \(S\) is the fidelity of the surrogate \(p_{\gamma}\), that measures how well it fits the predictions of the black box model \(\psi\). In addition to [Burkart and Huber, 2021], ContrXT adds \(\Omega(p_{\gamma}) \leq \Gamma\) as a constraint to Eq. 1 to keep the surrogate simple enough to be understandable while maximising the fidelity score. The constraint measures the complexity of the model whilst \(\Gamma\) is a bounding parameter.¹ In the global case, the surrogate model \(p_{\gamma}\) approximates \(\psi\) over the whole training set \(X\) taken from \(D\) which is representative of the distribution of the predictions of \(\psi\).

Output: Two white-box, rule based surrogates \(p_{1,2}\) of \(\psi_{1,2}\)

¹In case the surrogate is a decision tree, \(\Omega(p_{\gamma})\) might be the number of leaf nodes whilst it could be the number of non-zero coefficients in case of a logistic regression.
(C) Trace. This step aims at tracing the logic of the models \( p_{1,2} \) while working on a datasets \( D_{1,2} \). It generates the classifiers’ patterns through a global interpretable predictor (i.e., the surrogate), then it is encoded into the corresponding Binary Decision Diagram (BDD) [Bryant, 1986]. A BDD is a rooted, directed acyclic graph with one or two terminal nodes of out-degree zero, labelled 0 or 1. BDDs are usually reduced to canonical form, which means that given an identical ordering of input variables, equivalent Boolean functions will always reduce to the same BDD. Reduced ordered BDDs allow ContrXT to (i) compute compact representations of Boolean expressions, (ii) apply efficient algorithms for performing all kinds of logical operations, and (iii) guarantee that for any function \( f : \mathbb{B}^n \rightarrow \mathbb{B} \) there is one BDD representing it, testing whether it is true or false in constant time.

Output: two BDDs \( b_{1,2} \) representing the logic of \( p_{g_{1,2}} \).

(D) eXplain. This step takes as input the BDDs \( b_{1,2} \), that formalises the logic of the surrogates \( p_{g_{1,2}} \), and computes the BDDs encoding the differences between the two. Step D manipulates the BDDs generated from the Trace step to explain how \( \psi_1 \) and \( \psi_2 \) differ (i) quantitatively by calculating the distance metric defined below (aka, Indicators), and (ii) qualitatively by generating the BDDs of the added/deleted patterns over multiple datasets \( D_{1,2} \). As this is the key idea of ContrXT, we formalise the following.

**Definition 2.1 (T-contrast explanations through BDDs)**

Given \( f_1 : \mathbb{B}^n \rightarrow \mathbb{B} \) and \( f_2 : \mathbb{B}^m \rightarrow \mathbb{B} \) we define:

\[
\begin{align*}
f_1 \otimes f_2 &= \neg f_1 \land f_2 \quad (2) \\
f_1 \circ f_2 &= f_1 \land \neg f_2 \quad (3)
\end{align*}
\]

The goal of the operator \( \otimes (\odot) \) is to obtain a boolean formula that is true iff a variables assignment that satisfies (falsifies) \( f_1 \) is falsified (satisfied) in \( f_2 \) given \( f_1 \) (\( f_2 \)). Let \( b_1 \) and \( b_2 \) be two BDDs generated from \( f_1 \) and \( f_2 \) respectively, we synthesise the following BDDs:

\[
\begin{align*}
b_{\odot,1,2} &= b_1 \odot b_2 \quad (4) \\
b_{\odot,1,2} &= b_1 \circ b_2 \quad (5)
\end{align*}
\]

where \( b_{\odot} (b_{\circ}) \) is the BDD that encodes the reduced ordered classification paths that are falsified (satisfied) by \( b_1 \) and satisfied (falsified) by \( b_2 \). We also denote as:

- \( \text{var}(b) \) the variables of \( b \);
- \( \text{sat}(b_{\odot,1,2}) \) all the true (satisfied) paths of \( b_{\odot,1,2} \) removing \( \text{var}(b_1) \) \( \cup \text{var}(b_2) \);
- \( \text{sat}(b_{\odot,1,2}) \) all the true (satisfied) paths of \( b_{\odot,1,2} \) removing \( \text{var}(b_2) \) \( \cup \text{var}(b_1) \).

Both \( b_{\odot,1,2} \) and \( b_{\odot,1,2} \) encode the differences in the logic used by \( b_1 \) and \( b_2 \) in terms of feature presence (i.e., classification paths). Indeed, \( b_{\odot,1,2} \) (\( b_{\odot,1,2} \)) can be queried to answer a T-contrast question like “Why does a path on \( b_1 \) had a true (false) value, but now it is false (true) in \( b_2 \)”.

**Output**: Two BDDs \( b_{\odot,1,2} \) and \( b_{\odot,1,2} \) encoding the rules used by \( b_2 \) but not by \( b_1 \) and vice-versa.

(E) Generation of final explanations. Starting from \( b_{\odot,1,2} \) and \( b_{\odot,1,2} \), the final explanations are provided through a set of indicators and Natural Language Explanations. Indicators estimate the differences between the classification paths of the two BDDs through the Add and Del values (see Eq. 6 and 7). To compare \( \text{Add} \) and \( \text{Del} \) across classes, we compute the \( \text{Add}_{\text{Global}} (\text{Del}_{\text{Global}}) \) as the number of paths to true in \( b_{\odot} \) (\( b_{\circ} \)) over the corresponding maximum among all the \( b_{\odot} \) (\( b_{\circ} \)) with \( c \in C \). In the case of a multiclass classifier, as for 20newsgroup, ContrXT suggest focusing on classes that changed more with respect the indicators distribution.

\[
\begin{align*}
\text{Add}(b_{\odot,1,2}) &= \frac{\text{sat}(b_{\odot,1,2})}{|\text{sat}(b_{\odot,1,2})| + |\text{sat}(b_{\odot,1,2})|} \quad (6) \\
\text{Del}(b_{\odot,1,2}) &= \frac{\text{sat}(b_{\odot,1,2})}{|\text{sat}(b_{\odot,1,2})| + |\text{sat}(b_{\odot,1,2})|} \quad (7)
\end{align*}
\]

Natural Language Explanations (NLE) exhibits the added/deleted paths derived from \( b_{\odot} \) and \( b_{\circ} \) to final users through natural language. ContrXT uses the last four steps of six NLG tasks described by [Gatt and Krahmer, 2018], responsible for microplanning and realisation. In our case, the structured output of BDDs obviates the necessity of document planning which is covered by the first two steps.

The explanation is composed of two main parts, corresponding to Add and Del paths. Content of each part is generated by parsing the BDDs, extracting features, aggregating them using Frequent Itemsets technique [Rajaraman and Ullman, 2011] to reduce the redundancy, inserting the related parts in the predefined sentences [Rosenthal et al., 2016].

3 Results on a Benchmark Dataset

Evaluation. ContrXT was evaluated in terms of approximation quality to the input model to be explained (i.e., the fidelity of the surrogate) on 20newsgroups, a well-established benchmark used in [Jin et al., 2016] to build a reproducible text classifier, and in [Ribeiro et al., 2016], to evaluate LIME’s effectiveness in providing local explanations. We ran ContrXT over different classifiers, trained through the most used algorithms, such as linear regression (LR), random forest (RF), support vector machines with RBF (SVM), Naive Bayes (NB), Bidirectional Gated Recurrent Unit (biGRU) [Cho et al., 2014], and BERT [Devlin et al., 2019] (bert-base-uncased) with a sequence classification layer on top. Results are shown in Table 1. We considered and evaluated all the global surrogate models surveyed by [Burkart and Huber, 2021], representing the state of the art. Approaches falling outside the goal of ContrXT (e.g., SP-LIME [Ribeiro et al., 2016] and k-LIME [Hall et al., 2017] whose outcome
<table>
<thead>
<tr>
<th>ML</th>
<th>Model F1-w Surrogate F1-w</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$D_{t_1}$ $D_{t_2}$</td>
</tr>
<tr>
<td>LR</td>
<td>.88 .83</td>
</tr>
<tr>
<td>RF</td>
<td>.78 .74</td>
</tr>
<tr>
<td>SVM</td>
<td>.89 .84</td>
</tr>
<tr>
<td>NB</td>
<td>.91 .87</td>
</tr>
<tr>
<td>h-GRU</td>
<td>.79 .70</td>
</tr>
<tr>
<td>BERT</td>
<td>.84 .72</td>
</tr>
</tbody>
</table>

Table 1: ContrXT on 20newsgroups ($D_{t_1}$, $D_{t_2}$ from [Jin et al., 2016]) varying the ML algorithm. • indicates the best surrogate.

is limited to the feature importance values) and papers that did not provide the code were discarded.

To date, ContrXT relies on decision trees to build the surrogate, though it can employ any surrogate algorithms.

**Results Comment for 20newsgroup.** One might inspect how the classification changes from $\psi_1$ to $\psi_2$ for each class, i.e., which are the paths leading to class $c$ at time $t_1$ (before) that lead to other classes at time $t_2$ (now) (added paths) and those who lead to $c$ at $t_2$ that were leading to other classes at time $t_1$ (deleted paths). Focusing on the class *atheism* of Fig. 2 the number of deleted paths is higher than the added ones. Fig. 3 reveals that the presence of the word *bill* leads the $\psi_2$ to assign the label *atheism* whilst the presence of such a feature was not a criterion for $\psi_1$. Conversely, $\psi_1$ used the feature *keith* to assign the label, whilst $\psi_2$ discarded this rule. Actually, both terms refer to the name of the post’s authors.

The example of Fig. 3 sheds light on the goal of ContrXT, which is providing to the final user a way to investigate why the classifier change its learning criteria with regard to the past instance. NLE allows the user to discover that -though the accuracy of $\psi_1$ and $\psi_2$ is high - the underlying learning functions (i) learned terms that should have been discarded during the preprocessing, (ii) $\psi_2$ persists in relying on those terms, which are changed after retraining (using *bill* instead of *keith*), and (iii) having *political* *atheist* is no longer enough to classify in the class.

**Evaluation through Human Subjects.** We designed a study to assess if - and to what extent - final users can understand and describe what differs in the classifiers’ behaviour by looking at NLE outputs. We recruited 15 participants from prolific.co [Palan and Schitter, 2018] that were asked to look at NLE textual explanations and to select one (or more) statements according to the meaning they catch from NLEs. Results showed that the participants understood the NLE format and answered with an 89% accuracy on average, and an F1-score of 87%. Finally, we computed Krippendorff’s alpha coefficient to estimate the extent of agreement among users. We reached a value of 0.7, which [Krippendorff, 2004] considers as acceptable to positively assess the subjects consensus.

**Getting ContrXT.** ContrXT can be used either as a pip Python package [Malandri et al., 2022b] or as a service through REST API. The API is written using Python and the Flask library [Grinberg, 2018] and can be invoked using a few lines code shown in Listing 1. A load testing has been performed using locust.io to measure the quality of service of the ContrXT’s API, adding a virtual user every 10 sec. Our architecture reached a throughput of 2.55 users per second. Beyond this value, the API service keeps working, putting additional requests into a queue.

**Demo Video.** Available at [https://tinyurl.com/ContrXTIJCAI](https://tinyurl.com/ContrXTIJCAI)

```python
1 import requests, io
2 from zipfile import ZipFile
3 files = {'time_1': open(t1_csv_path, 'rb'), 'time_2': open(t2_csv_path, 'rb')}
4 f = requests.post('[see details on github repo]', files=files)
5 result = ZipFile(f.Btimestamp(t_’content’)

Listing 1: Complete Python code to call ContrXT API

### 4 Conclusion, Limitations and Future Work

We demonstrated ContrXT [Malandri et al., 2022a] a novel model-agnostic tool to globally explain how a black box text classifier change its learning criteria with regard to the past (T-contrast) by manipulating BDDs. The evaluations have been performed on a multiclass benchmark, i.e., 20newsgroup. Our evaluation using different learning algorithms revealed ContrXT can work with the state-of-the-art learning algorithms, reaching an F1-weighted surrogate fidelity up to 0.87 and never below 0.73. The Spearman correlation test revealed the accuracy is not correlated with the ADD/DEL indicators, confirming they provide additional insights beyond the quality of the trained models. ContrXT is available as a Python package on Github. To date, ContrXT is bounded to explain text classifiers. We are working to extend ContrXT in dealing with tabular classifiers.
References


