Contrastive Explanations of Text Classifiers as a Service

Lorenzo Malandri[1,3], **Fabio Mercorio**[1,3], **Mario Mezzanzanica**[1,3], **Navid Nobani**[2], **Andrea Seveso**[2,3]
[1]Dept of Statistics and Quantitative Methods, University of Milano Bicocca, Italy
[2]Dept of Informatics, Systems and Communication, University of Milano Bicocca, Italy
[3]CRISP Research Centre crispresearch.eu, University of Milano Bicocca, Italy

Abstract

The recent growth of black-box machine-learning methods in data analysis has increased the demand for explanation methods and tools to understand their behaviour and assist human-ML model cooperation. In this paper, we demonstrate `ContrXT` , a novel approach that uses natural language explanations to help users to comprehend how a back-box model works. `ContrXT` provides time contrastive (t-contrast) explanations by computing the differences in the classification logic of two different trained models and then reasoning on their symbolic representations through Binary Decision Diagrams. `ContrXT` is publicly available at `ContrXT.ai` as a python pip package.

1 Introduction and Contribution

Consider a text classifier ψ_1, retrained with new data and resulting into ψ_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 explain why the logic is changed. Hence, such a user might wonder *"why do the criteria used by ψ_1 result in class c, but ψ_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_i, but property Q at time t_j?"* (Miller, 2019; Van Bouwel and Weber, 2002).

1.1 Contribution

In this paper we demonstrate `ContrXT` as a service, built on top of the approach we presented in (Malandri et al., 2022). `ContrXT` (Contrastive eXplainer for Text classifier) is a tool that 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.[1] We present a system to deliver `ContrXT` as a service, together with detailed insights and metrics. Among them, we introduce an explanation of how - and to what extent - the single classification rules differ through time, along with examples of instances with the the rules used by classifiers highlighted in the text.

To date, there is no work (other than `ContrXT`) 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)).

2 `ContrXT` in a nutshell

`ContrXT` aims at explaining how a classifier changes its predictions through time. Below 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 $\mathcal{D}_i$ under C consists of $|C|$ independent problems of classifying each $d \in \mathcal{D}_i$ under a class c_i for $i = 1, \ldots, |C|$. Hence, a *classifier* for c_i is a function $\psi : \mathcal{D} \times C \to \mathbb{B}$ approximating an unknown target function $\dot{\psi}$.
Output: Two black-box classifiers on the same class set.

(B) Post-hoc interpretation. Following the study about ML post-hoc explanation methods

[1]http://ContrXT.ai

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations, pages 46 - 53

of (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:

$$p_g* = \arg\max_{p_g \in I} \frac{1}{X} \sum_{x \in X} S(p_g(x), \psi(x)) \quad s.t. \Omega(p_g) \leq \Gamma \tag{1}$$

where I represents a set of possible white box models to be chosen as surrogates, and S is the fidelity of the surrogate p_g, that measures how well it fits the predictions of the black box model ψ. In addition to (Burkart and Huber, 2021), `ContrXT` adds $\Omega(p_g) \leq \Gamma$ 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 Γ is a bounding parameter. In the global case, the surrogate model p_g approximates ψ over the whole training set X taken from $\mathcal{D}$ which is representative of the distribution of the predictions of ψ.[2]
Output: Two white-box, rule based surrogates $p_{1,2}$ of $\psi_{1,2}$

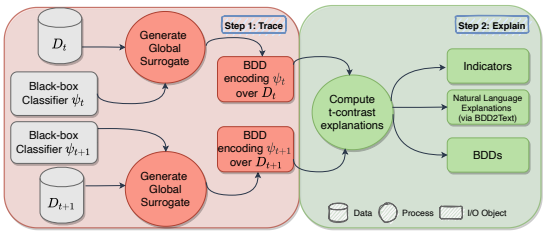

Figure 1: Overview of `ContrXT`, taken from (Malandri et al., 2022)

(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 $p_{g1,2}$.

(D) eXplain. This step takes as input the BDDs $b_{1,2}$, that formalise the logic of the surrogates $p_{g1,2}$, and computes the BDDs encoding the *differences* between the two. Step D manipulates the BDDs generated from the Trace step to explain how ψ_1 and ψ_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_{t_i}. As this is the key idea of `ContrXT`, we formalise the following.

Definition: T-contrast explanations using BDDs Given $f_1 : \mathbb{B}^n \rightarrow \mathbb{B}$ and $f_2 : \mathbb{B}^m \rightarrow \mathbb{B}$ we define:

$$f_1 \oslash f_2 = \neg f_1 \wedge f_2 \quad (2) \qquad f_1 \oslash f_2 = f_1 \wedge \neg f_2 \quad (3)$$

The goal of the operator $\oslash$ ($\oslash$) 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:

$$b_{\oslash}^{b_1,b_2} = b_1 \oslash b_2 \quad (4) \qquad b_{\oslash}^{b_1,b_2} = b_1 \oslash b_2 \quad (5)$$

where $b_{\oslash}$ ($b_{\oslash}$) 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

- $var(b)$ the variables of b;
- $sat(b_{\oslash}^{b_1,b_2})$ all the true (satisfied) paths of $b_{\oslash}^{b_1,b_2}$ removing $var(b_1) \setminus var(b_2)$;
- $sat(b_{\oslash}^{b_1,b_2})$ all the true (satisfied) paths of $b_{\oslash}^{b_1,b_2}$ removing $var(b_2) \setminus var(b_1)$.

Both $b_{\oslash}^{b_1,b_2}$ and $b_{\oslash}^{b_1,b_2}$ encode the differences in the logic used by b_1 and b_2 in terms of feature presence (i.e., classification paths). Indeed, $b_{\oslash}^{b_1,b_2}$ ($b_{\oslash}^{b_1,b_2}$) can be queried to answer a T-contrast question like *"Why did a path on b_1 have a true (false) value, but now it is false (true) in b_2?"*. Clearly, features discarded (added) by b_2 are removed from paths of $b_{\oslash}^{b_1,b_2}$ ($b_{\oslash}^{b_1,b_2}$) as they are used by ψ_1.
Output: Two BDDs $b_{\oslash}^{b_1,b_2}$ and $b_{\oslash}^{b_1,b_2}$ encoding the rules used by b_2 but not by b_1 and vice-versa.

[2] in case the surrogate is a decision tree, $\Omega(p_g)$ might be the number of leaf nodes whilst it could be the number of non-zero coefficients in case of a logistic regression

(E) Generation of final explanations: Starting from $b_{\circledcirc}^{b_1,b_2}$ and $b_{\circledcirc}^{b_1,b_2}$, 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 *add* and *del* across classes, we compute the *Add_Global* (*Del_Global*) as the number of paths to true in $b_{\circledcirc}$ ($b_{\circledcirc}$) over the corresponding maximum among all the $b_{\circledcirc}^{c}$ ($b_{\circledcirc}^{c}$) with $c \in C$.

In the case of a multiclass classifier, as for 20newsgroup, `ContrXT` suggests focusing on classes that changed more with respect the indicators distribution.

$$Add(b_{\circledcirc}^{b_1,b_2}) = \frac{|sat(b_{\circledcirc}^{b_1,b_2})|}{|sat(b_{\circledcirc}^{b_1,b_2})| + |sat(b_{\circledcirc}^{b_1,b_2})|} \quad (6)$$

$$Del(b_{\circledcirc}^{b_1,b_2}) = \frac{|sat(b_{\circledcirc}^{b_1,b_2})|}{|sat(b_{\circledcirc}^{b_1,b_2})| + |sat(b_{\circledcirc}^{b_1,b_2})|} \quad (7)$$

Natural Language Explanations (NLE) exhibits the added/deleted paths derived from $b_{\circledcirc}$ and $b_{\circledcirc}$ 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).

2.1 `ContrXT` as a Service

`ContrXT` has been implemented through Python as a pip package, using scikit-learn for generating surrogates and pyEDA package for synthesising BDDs. It takes as input the training data and the predicted labels by the classifier.

The user can specify (i) the coverage of the dataset to be used (default: 100%), otherwise a sampling procedure is used; (ii) to obtain explanations either for the multiclass case (default: one class vs all) or the two-class case (class vs class, by restricting the surrogate generation to those classes); (iii) the Γ value as a measure of complexity of the surrogate.

`ContrXT` can be used either as a pip Python package[3] or as a service through REST API. In the former case, `ContrXT` can be easily installed via `pip install contrxt`. Then, it can be executed as in Code 1. A python notebook ready to use is available on the Google Colab platform.[4]

```
from contrxt.contrxt import ContrXT
exp = ContrXT(
    X_t1, predicted_labels_t1,
    X_t2, predicted_labels_t2,
    save_path='results/',
    hyperparameters_selection=True,
)
exp.run_trace()
exp.run_explain()
exp.explain.BDD2Text()
```

Code 1: Invoke `ContrXT` with few lines of code.

The API is written using Python and the Flask library (Grinberg, 2018) and can be invoked using a few lines code shown in Code 3. Users are required to upload two csv files for time 1 and 2 Each csv is expected to have two columns respectively for corpus (texts to be classified) and predicted (the outcome of the classifier) for which the schema is shown in the following JSON.

```
schema = {
    "type": "csv",
    "columns": {
        "corpus": {"type": "string"},
        "predicted": {"type": "string"},
    },
}
```

Code 2: API schema

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 seconds, executing the whole `ContrXT` process for the *20newsgroups* dataset for each. Time needed to upload/download datasets and to generate PDF versions of the BDDs are not considered. We followed (Menascé, 2002) to determine the number of users/requests our API web server can tolerate in order to guarantee an acceptable response time (set to 5 minutes) while increasing the throughput, i.e., requests per second.

Our architecture reached a throughput of 2.55 users per second, as seen in Fig. 2. Beyond this value, the API service keeps working, putting additional requests into a queue.

```
import requests, io
from zipfile import ZipFile
```

[3] https://pypi.org/project/contrxt/
[4] https://tinyurl.com/ContrXT-pyn

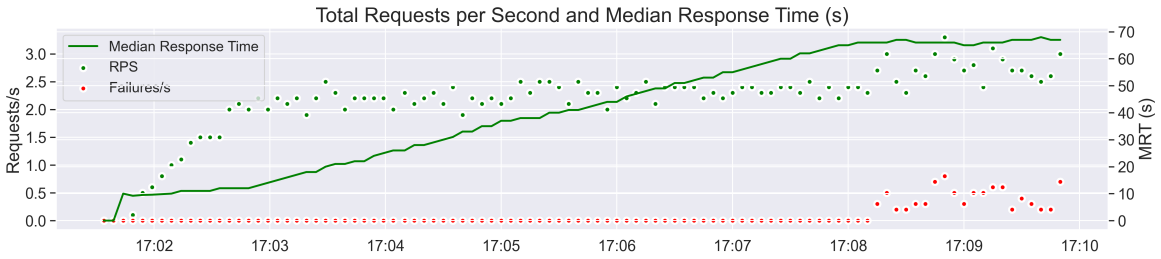

Figure 2: Load testing provided by Locust.io: MRT (median response time, green), Request per Seconds (throughput, green) and the number of failures (requests reached the 5 min timeout, red).

```
files = {
    'time_1': open(t1_csv_path, 'rb'),
    'time_2': open(t2_csv_path, 'rb')
    }
r = requests.post('[see details on
    github repo]',files=files)
result = ZipFile(io.BytesIO(r.content))
```

Code 3: Complete Python code to call `ContrXT` API

3 Experimental 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-known 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 using popular classifiers such as linear regression (LR), random forest (RF), support vector machines (SVM), Naive Bayes (NB), Bidirectional Gated Recurrent Unit (bi-GRU) (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 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.

3.1 Results Comment for 20newsgroup

One might inspect how the classification changes from ψ_1 to ψ_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

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

ML Algo	Model F1-w		Surrogate Fidelity F1-w	
	D_{t_1}	D_{t_2}	D_{t_1}	D_{t_2}
LR	.88	.83	.76 (±.06)	.78 (±.07)
RF	.78	.74	.77 (±.06)	.79 (±.07)
SVM	.89	.84	.76 (±.06)	.78 (±.06)
NB	.91	.87	.76 (±.06)	.78 (±.06)
bi-GRU	.79	.70	.77 (±.06)	.78 (±.06)
BERT	.84	.72	.78 (±.05) •	.83 (±.06) •

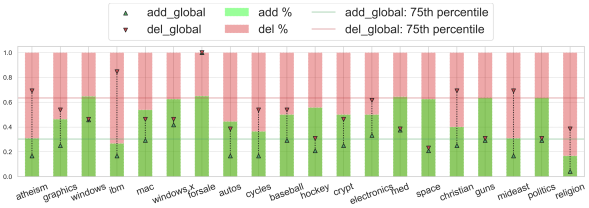

Figure 3: Indicators for the changes in classification paths from t_1 to t_2 for each *20newsgroup* class. On the x-axis, we present the classification classes, and on the y-axis the ADD/DEL indicators

classes at time t_1 (*deleted paths*). Focusing on the class *atheism* of Fig. 3 the number of deleted paths is higher than the added ones. Fig. 4 reveals that the presence of the word *bill* leads the ψ_2 to assign the label *atheism* whilst the presence of such a feature was not a criterion for ψ_1. Conversely, ψ_1 used the feature *keith* to assign the label, whilst ψ_2 discarded this rule. Actually, both terms refer to the name of the posts' authors.

The example of Fig. 4 sheds light on the goal of `ContrXT`, which is providing to the final user a way to investigate why ψ_2 classified documents to a different class with respect to ψ_1, as well as monitoring future changes. NLE allows the user to discover that -though the accuracy of ψ_1 and ψ_2 is high[5]- the underlying learning functions (i) learned

[5]The Spearman correlation test revealed the accuracy is not correlated with the ADD/DEL indicators, confirming they

terms that should have been discarded during the preprocessing, (ii) ψ_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.

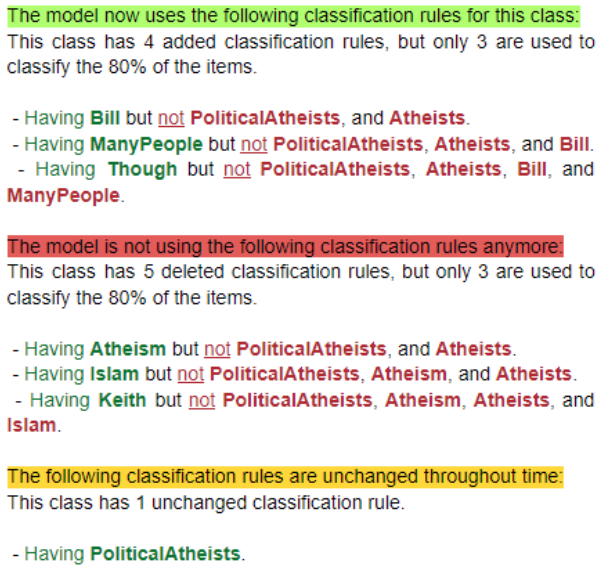

Figure 4: NLE for *alt.atheism* using the BERT model of Tab. 1

Get Rule Examples. The NLE shows the differences between the two models. However, a user might also wish to see example instances in the datasets where these rules apply.

To do so, `ContrXT` provides the *get_rule_examples* function, which requires the user to specify a rule to be applied and the number of examples to show. `ContrXT` applies the rule to D_1 and D_2, specifying the number of document classified by that rule and provides some examples, highlighting in the text the portion in which the rule applies, as in Fig. 5.

Notice this function is also useful to check the consistency of a specific rule, that is, for an *add* rule, its prevalence should be higher in D_1, for a *del* rule the opposite, while for a *still* rule the should be roughly equivalent in both D_1 and D_2.

3.2 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), an online service that provides participants for research

provide additional insights beyond the quality of the trained models

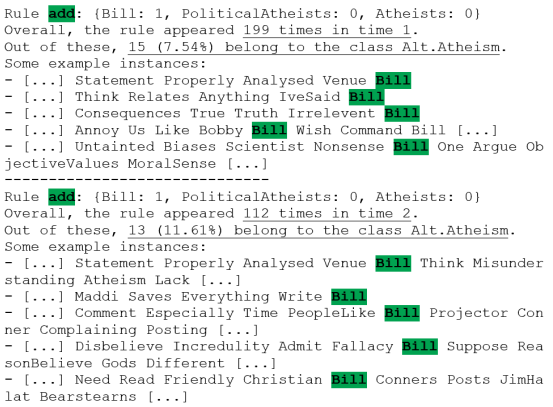

Figure 5: `ContrXT` shows examples in which a rule applies for the class *alt.atheism*.

studies. Participants 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, a statistical measure of the extent of agreement among users. We reached an alpha value of 0.7, which Krippendorff (2004) considers as acceptable to positively assess the subjects consensus.

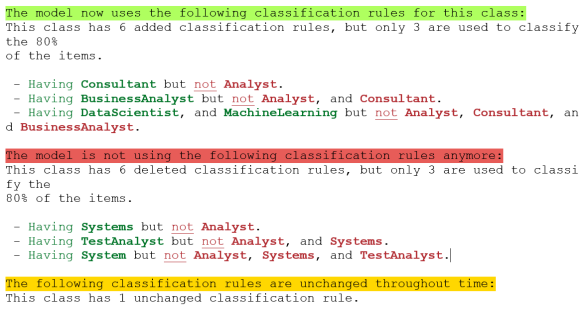

Figure 6: NLE for *2511, Systems Analysts* using the RF model.

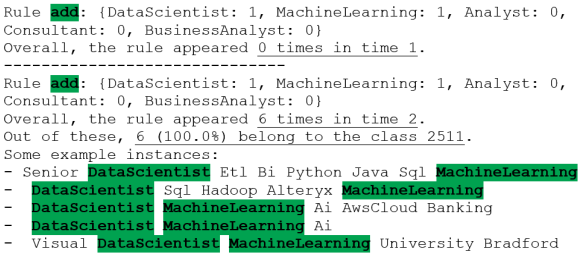

Figure 7: `ContrXT` shows examples in which a rule applies for the class *2511, Systems Analysts* .

3.3 ContrXT in a real-life scenario

In the last years, the problem of extracting knowledge from online job ads (OJA, aka, online job vacancies) in terms of occupations and skills is growing in interest in academic, industrial, and government organisations to monitor and understanding labour market changes (i) timely and (ii) at a very fine-grained geographical level, (iii) catching novelties in terms of novel occupations and skills as soon as they emerge in the real-labour market. This is the goal of labour market intelligence (aka, skill intelligence) which refers to the use and design of AI algorithms and frameworks to analyse labour market data for supporting decision making (see, e.g., (Giabelli et al., 2021a,c; Turrell et al., 2018; Zhang et al., 2019)).

From a statistical perspective, in late 2020 EUROSTAT and Cedefop have joined forces announcing a call for tender (EuroStat, 2020) aimed at establishing results from (CEDEFOP, 2016a,b) fostering AI and Statistics to build up the European Hub of Online Job Ads. In such a scenario, training an ML model would be helpful to support questions such as: *Which occupations will grow in the future and where? What skills will be demanded the most in the next years?* However, once such an ML model has been trained and deployed (see, e.g., (Colombo et al., 2019; Boselli et al., 2018)) it needs to be periodically re-trained as the labour market demand constantly changes through time, mainly due to rise of new emerging occupations and skills (Giabelli et al., 2021a,b). This, in turn, leads policy makers to ask if - and to what extent - the re-trained model is coherent in classifying new job ads with respect to the past criteria.

As an example, let us consider the *systems analysts*, an occupation that changed a lot in the last years driven by technological progresses (Malandri et al., 2021). A policy maker might ask: *"how systems analysts are now classified by the updated model, and how the updated model differs with respect to the previous one?"*

Figure 6 shows the difference in the criteria between the two classifiers for the class *"Systems Analysts"*. The Figure shows that the updated model considers *business analysts* as *Systems Analysts*. Furthermore, the user can easily discover that a novel occupation, i.e., *"data scientist"*, is considered as a system analyst by the updated model. On the other side, Fig. 6 clarifies to the user that the updated model changed its criterion in regard to the term "test analyst", that now does not characterise the class anymore. Being able to catch those differences -class by class- is helpful to end users as it allows understanding to what extent the updated model is coherent with past predictions, as well as its ability to catch the novelty in the domain and terms that might lead the model to misclassification. Furthermore, the Get Rule function provides samples to the user, as shown in Fig. 7.

4 Conclusion, Limitations and Future Work

In this demonstration we presented a system to deliver contrastive explanations of text classifiers as a service. The system is built on top of ContrXT (Malandri et al., 2022), 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. Given two classifiers trained on the same target class set, the system we presented provides time contrastive explanations of their behaviour, together with detailed insights and metrics. Among them, we presented the possibility to highlight how and how much the classification rules differ along time. A load test demonstrated that our architecture has a throughput of 2.55 users per second. Beyond this value, the API service puts the additional requests into a queue but keeps working.

To date, ContrXT is bounded to explain text classifiers. We are working to extend ContrXT to tabular classifiers.

4.1 Demonstration of ContrXT

Video to see ContrXT in action through a video demonstration at https://tinyurl.com/ContrXT-NAACL.

Google Colab to run ContrXT directly on a python notebook, using Google Colab resources at at https://tinyurl.com/ContrXT-pyn

REST-API to embed ContrXT into a third-party application. Notice, it is required to ask for credential at https://tinyurl.com/contrxt-request-form

GitHub to download as well as to contribute to this project, at https://ContrXT.ai

Impact Statement and Ethical Considerations

AI-based decision systems interact with humans in many application domains, including sensitive ones like credit-worthiness, education and law enforcement. An unmitigated data-driven decision-making algorithm can systematically make unfair decisions against certain population subgroups with specific attributes (e.g. race or gender) due to the inherited biases encoded in the data. Even a system which has been carefully trained in order to mitigate such effects can change its behaviour over time, due to changes in the underlying data. The opaque nature of machine learning models can hide those unfair behaviours to the end user.

In this context, `ContrXT` might reveal itself extremely useful in tracing and explaining how the model, which was designed to be fair at time 1, changed its behaviour and rules after being retrained at time 2. This allows one to check whether the model kept fair over time.

An interesting example of application in such sense is the paper *Towards Fairness Through Time (Castelnovo et al., 2021)*, presented at the *2nd Workshop on Bias and Fairness in AI (BIAS)*[6] at ECML-PKDD, which uses `ContrXT` to observe the evolution of a ML model for credit lending over time. Understanding the changing of the gaps between different population subgroups, like gender or race, allows observing whether the mitigation strategies in place are bringing benefits to society, favoring the convergence between individual and group fairness.

References

Roberto Boselli, Mirko Cesarini, Stefania Marrara, Fabio Mercorio, Mario Mezzanzanica, Gabriella Pasi, and Marco Viviani. 2018. Wolmis: a labor market intelligence system for classifying web job vacancies. *Journal of Intelligent Information Systems*, 51(3).

Randal E Bryant. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*.

Nadia Burkart and Marco F Huber. 2021. A survey on the explainability of supervised machine learning. *JAIR*, 70:245–317.

Alessandro Castelnovo, Lorenzo Malandri, Fabio Mercorio, Mario Mazzanzanica, and Andrea Cosentini. 2021. Towards fairness through time. *ECML PKDD, CCIS 1524*, page 1–17.

[6] https://sites.google.com/view/bias2021/

CEDEFOP. 2016a. Real-time labour market information on skill requirements: Setting up the eu system for online vacancy analysis. https://goo.gl/5FZS3E.

CEDEFOP. 2016b. Real-time labour market information on skill requirements: Setting up the eu system for online vacancy analysis. https://goo.gl/5FZS3E.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Emilio Colombo, Fabio Mercorio, and Mario Mezzanzanica. 2019. AI meets labor market: exploring the link between automation and skills. *Information Economics and Policy*, 47.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

EuroStat. 2020. Towards the european web intelligence hub — european system for collection and analysis of online job advertisement data (wih-oja), available at https://tinyurl.com/y3xqzfhp.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *JAIR*, 61.

Anna Giabelli, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Andrea Seveso. 2021a. NEO: A system for identifying new emerging occupation from job ads. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 16035–16037. AAAI Press.

Anna Giabelli, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Andrea Seveso. 2021b. Neo: A system for identifying new emerging occupation from job ads. In *AAAI - Demo track*.

Anna Giabelli, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Andrea Seveso. 2021c. Skills2job: A recommender system that encodes job offer embeddings on graph databases. *Appl. Soft Comput.*, 101:107049.

Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Patrick Hall, Navdeep Gill, Megan Kurka, and Wen Phan. 2017. Machine learning interpretability with h2o driverless ai. *H2O. ai. URL: http://docs. h2o. ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet. pdf*.

Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for text classification. In *IJCAI*, pages 2824–2830.

Klaus Krippendorff. 2004. Reliability in content analysis: Some common misconceptions and recommendations. *Human communication research*, 30(3).

Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Navid Nobani. 2021. MEET-LM: A method for embeddings evaluation for taxonomic data in the labour market. *Comput. Ind.*, 124:103341.

Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, Navid Nobani, and Andrea Seveso. 2022. ContrXT: Generating contrastive explanations from any text classifier. *Information Fusion*, 81:103–115.

Daniel A Menascé. 2002. Load testing of web sites. *IEEE internet computing*, 6(4):70–74.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267.

Shane T Mueller, Robert R Hoffman, William Clancey, Abigail Emrey, and Gary Klein. 2019. Explanation in human-ai systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable ai. *arXiv preprint arXiv:1902.01876*.

Stefan Palan and Christian Schitter. 2018. Prolific. ac—a subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, 17:22–27.

Anand Rajaraman and Jeffrey David Ullman. 2011. *Mining of massive datasets*. Cambridge University Press.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *ACM-SIGKDD*.

Stephanie Rosenthal, Sai P Selvaraj, and Manuela M Veloso. 2016. Verbalization: Narration of autonomous robot experience. In *IJCAI*, volume 16, pages 862–868.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *CSUR*, 34(1).

Arthur Turrell, Bradley Speigner, Jjyldy Djumalieva, David Copple, and James Thurgood. 2018. Using job vacancies to understand the effects of labour market mismatch on uk output and productivity.

Jeroen Van Bouwel and Erik Weber. 2002. Remote causes, bad explanations? *JTSB*, 32(4).

Denghui Zhang, Junming Liu, Hengshu Zhu, Yanchi Liu, Lichen Wang, Pengyang Wang, and Hui Xiong. 2019. Job2vec: Job title benchmarking with collective multi-view representation learning. In *CIKM*.