OLEA: Tool and Infrastructure for Offensive Language Error Analysis in English

Marie Grace[*], **Xajavion "Jay" Seabrum**[*], **Dananjay Srinivas**[*], **Alexis Palmer**
University of Colorado Boulder
first.last@colorado.edu; olea.ask@gmail.com[†]

Abstract

State-of-the-art models for identifying offensive language often fail to generalize over more nuanced or implicit cases of offensive and hateful language. Understanding model performance on complex cases is key for building robust models that are effective in real-world settings. To help researchers efficiently evaluate their models, we introduce OLEA, a diagnostic, open-source, extensible Python library that provides easy-to-use tools for error analysis in the context of detecting offensive language in English. OLEA packages analyses and datasets proposed by prior scholarship, empowering researchers to build effective, explainable and generalizable offensive language classifiers.

1 Introduction

Offensive language[1] detection models are integral to online platforms' moderation systems. Such systems excel at detecting and filtering out messages with explicit keywords and mentions, however these systems are known (1) to perform poorly on messages that are implicitly offensive or have negation (Röttger et al., 2020; Palmer et al., 2020); (2) to be subject to annotator biases (Sap et al., 2021); (3) not to be robust to diachronic language (Florio et al., 2020); and (4) to be insensitive to and to overdetect AAE as offensive language (Sap et al., 2021; Blodgett et al., 2016). Failing to address these issues and gaps can cause marginalized groups to be further dehumanized or attacked (Mathew et al., 2021; Kennedy et al., 2020).

Models have been shown to be ineffective at generalizing across these complexities (Yin and Zubiaga, 2021), tending to aggregate different types of hate speech under broad labels, causing large within-class variances (Waseem et al., 2017a). In response, prior research has curated diagnostic datasets such as HateCheck (Röttger et al., 2020) and COLD (Palmer et al., 2020), to evaluate existing models on specific types of hate speech. Such evaluation datasets allow us to view model performance as a continuum, and move away from monolithic F1 scores that can obscure a model's limitations and explainability (Kennedy et al., 2020).

We introduce OLEA,[2,3] an extensible, open-source Offensive Language Error Analysis tool and infrastructure designed to a) evaluate offensive language classifiers on different types of problematic language use, and b) provide detailed feedback about model performance. The library makes it convenient for researchers to analyze their models by providing an extensive set of error-analysis methods, callable with minimal coding, to measure case-specific model performances. In addition, the library provides a common interface for comparing different offensive language classifiers on granular linguistic categories. OLEA provides:

- nuanced error analysis methods focused on understanding model performance on specific linguistic occurrences;
- interfaces to two evaluation datasets that compile a broad typology of offensive language phenomena; and
- scaffolding to support easy distribution of new datasets and associated analysis methods.

OLEA formats evaluation datasets to be easily interpreted, and uses popular Python packages such as Pandas (Wes McKinney, 2010) and Matplotlib (Hunter, 2007) to encapsulate the data and error analysis methods.

2 Background and Related Work

Offensive language is complex, and systems for detecting it automatically need to be able to handle both explicit *and* implicit cases (Schmidt and

[*] The first three authors contributed equally.
[†] Please direct inquiries about the library to this email.
[1] We use the term "offensive language" to encompass offensive language and hate speech. This paper contains censored offensive language examples.
[2] https://pypi.org/project/olea/
[3] https://www.youtube.com/watch?v=e8VVhP6kNlY

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics
System Demonstrations, pages 209–218
May 2-4, 2023

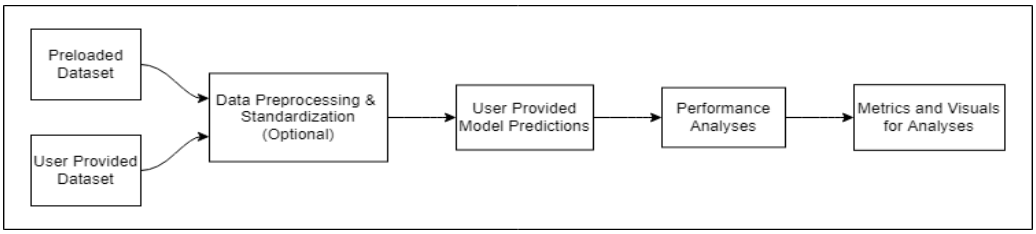


Figure 1: Diagram of the OLEA Library pipeline.

Wiegand, 2017; Waseem et al., 2017b). Detecting explicit offensive language often relies on keyword detection (Wiegand et al., 2019), but keyword-driven systems can lead to messages being falsely flagged, causing unchecked or unnoticed racial biases to propagate in the system's decisions (Sap et al., 2021; Blodgett et al., 2016). Implicit offensive language is generally more difficult to detect than its explicit counterpart (ElSherief et al., 2021; Caselli et al., 2020). It is also more likely to change over time, as the world changes, and as users coin new phrases and terms to implicitly refer to minority groups (Florio et al., 2020).

Datasets for this task address different (often overlapping) concerns, making direct comparison difficult. HateXplain (Mathew et al., 2021) and CAD (Vidgen et al., 2021) both provide rationales indicating where annotators see offensive content. OLID (Zampieri et al., 2019a) identifies offensive text and the specific targeted minority group in a three-tiered labeling structure. HateCheck (Röttger et al., 2020) and COLD (Palmer et al., 2020) are described more in Section 3.1.

Linguistic explainability of the prediction *failures* of NLP models has lagged behind performance gains according to benchmark datasets (Hovy, 2022). McMillan-Major et al. (2022) provide an interactive system mostly for end users of offensive language detection systems. Their system helps users explore datasets and understand how different models score and classify individual text inputs.OLEA has complementary functionality, focusing on fine-grained analysis of model performance (especially misclassifications) across existing evaluation datasets. **We focus instead on model developers, providing streamlined error analysis and interpretation of system outputs relative to linguistically-grounded categorizations.**

3 Library Tour and Design

Figure 1 shows an overview of OLEA's core functionalities. Users submit their model's predictions (3.2) on OLEA's preloaded datasets (3.1) and then call error analysis and evaluation functions (3.3). Users may also extend OLEA with new datasets and may write new analysis functions, adding to the library's capabilities (3.4). Most of OLEA's modules expect a Pandas[4] dataframe with the text of the instance to be classified, one or more labels indicating offensiveness, and a predicted label for the instance. Dataframes may include columns with additional information related to the instance and/or its annotation.

3.1 Preloaded diagnostic datasets

The primary function of OLEA is to make it easy for users to evaluate the capabilities of their models in a fine-grained way. We provide interfaces (via HuggingFace's datasets library[5] and the HuggingFace Hub) to two offensive language datasets, both designed specifically for diagnostic evaluation of detection systems. Both datasets include fine-grained annotations and binary offensiveness labels and were curated to compare model performance with linguistic phenomena. Tables 1 and 2 list the features available for analysis.

The **HateCheck** (Röttger et al., 2020) test suite includes labels reflecting specific linguistic constructions often seen in online hate speech, such as use of spelling changes to obscure hateful language and presence of threatening language. HateCheck also includes annotations of specific identities targeted in each instance of hate speech.

COLD (Palmer et al., 2020) provides fine-grained labels of some linguistic phenomena relevant for implicit/complex offensive language. Two examples are presence/absence of slur terms and presence/absence of adjectival nominalizations.

3.2 Submitting predictions

Before using the analysis functions described below, the user needs to submit their model's predic-

[4] https://pandas.pydata.org/docs/
[5] https://huggingface.co/docs/datasets/index

Feature	Description
functionality	The shorthand for the functionality tested by the test case.
case_id	The unique ID of the test case (assigned to each of the 3,901 cases initially generated)
test_case	The text of the test case.
label_gold	The gold standard label (hateful/non-hateful) of the test case. All test cases within a given functionality have the same gold standard label.
target_ident	Where applicable, the protected group targeted or referenced by the test case. We cover seven protected groups in the test suite: women, trans people, gay people, black people, disabled people, Muslims and immigrants.
direction	For hateful cases, the binary secondary label indicating whether they are directed at an individual as part of a protected group or aimed at the group in general.
focus_words	Where applicable, the key word or phrase in a given test case (e.g. "cut their throats").
focus_lemma	Where applicable, the corresponding lemma (e.g. "cut sb. throat").
ref_case_id	For hateful cases, where applicable, the ID of the simpler hateful case which was perturbed to generate them. For non-hateful cases, where applicable, the ID of the hateful case which is contrasted.
ref_templ_id	The equivalent, but for template IDs.
templ_id	The unique ID of the template from which the test case was generated (assigned to each of the 866 cases and templates from which we generated the 3,901 initial cases).

Table 1: Features available via the HateCheck dataset. Names and descriptions from Röttger et al. (2020).

Feature	Description
ID	The unique ID for the text
Text	The text containing social media messages (some containing offensive language)
Cat	The gold label category of the text
Off	Offensive or not? (Y / N) Majority Vote
Off1, Off2, Off3	Individual annotator labels for Off (Y / N)
Slur	Contains a slur? (Y / N) Majority Vote
Slur1, Slur2, Slur3	Individual annotator labels for Slur (Y / N)
Nom	Contains adjectival nominalization? (Y / N) Majority Vote
Nom1, Nom2, Nom3	Individual annotator labels for Slur (Y / N)
Dist	Contains linguistic distancing? (Y / N) Majority Vote
Dist1, Dist2, Dist3	Individual annotator labels for Dist (Y / N)

Table 2: Features available as part of the COLD dataset. Names and descriptions from Palmer et al. (2020).

tions on the selected dataset, as well as a mapping between the model's predicted labels (e.g. 1, 0) and the labels in the selected dataset (e.g. `hateful`, `non-hateful`).[6] The code snippet below illustrates the process and assumes that the user's data has been stored as a Pandas dataframe named `user_data`. In this example, the model predictions are found in a column called `predictions`. The user has selected three features for potential analysis: `Text`, `is_slur` and `text_length`.

```
from olea.data import Dataset
setup = Dataset(
          data = user_data,
          features = ["Text","is_slur",
                      "text_length"],
          gold_column = "gold_labels",
          text_column = "Text")
predictions = user_data["predictions"]
mapping = {"hateful": 1,
           "non-hateful": 0}
data_submit = setup.submit(
            batch = user_data,
            predictions = predictions,
            map = mapping)
```

The `submit` method passes the relevant parameters to the analysis module.

3.3 Error analysis functions

The heart of our library is a collection of functions for detailed evaluation and error analysis. **Throughout, we evaluate the model's coarse-grained classification performance (e.g. offensive vs. not offensive) for subsets of instances grouped according to a particular feature. The features generally correspond to dataframe columns. For example, we may compare performance for instances containing a slur term to performance for instances with no slur term.** Plots are produced using Matplotlib (Hunter, 2007), and we include the option to save plots to files. Section 4 shows concrete examples of the analysis outputs, and code examples appear in Appendix A.

`analyze_on`. In its most general version, this function evaluates model performance for a categorical column specified by the user. OLEA includes versions of `analyze_on` customized to the two preloaded datasets. The COLD-specific version evaluates performance for features constructed from combinations of four binary features: offensiveness, presence of slur term, presence of adjectival nominalization, and presence of linguistic distancing. The HateCheck-specific version includes linguistic features (e.g. negation, derogation, or profanity) and features related to the identity of the targeted individual or class (e.g. trans people, Muslims, or disabled people).

`check_anno_agreement`. This function is intended for datasets which include labels from multiple annotators, such as COLD. The function compares performance on instances with full annotator

[6] Note that this process applies both for preloaded datasets and for datasets read in from the user's own system.

agreement for the label of offensiveness to performance on instances with partial agreement. Full annotator agreement is taken as a proxy for instances that are "easy" to classify, and partial agreement indicates more complex cases.

aave. This function evaluates performance for instances (likely) written using African American English. The scores are calculated using the TwitterAAE model (Blodgett et al., 2016). These scores represent an inference of the proportion of words in the instance that come from a demographically-associated language/dialect.

check_substring. Given a user-specified text string, this function compares performance on instances with the substring to instances without.

str_len_analysis. This function outputs a histogram showing model performance on instances of different lengths (character or word count).

3.4 Adding datasets and analyses

Extensibility is a key principle guiding the design of OLEA, with the goal of providing an easy-to-use, uniform platform for error analysis in the context of offensive language detection. In addition to the two preloaded datasets, users can submit their own datasets using the process described in 4.2. To add a dataset, users need only define an interface (using the Dataset class) indicating where to find both target labels and features relevant for analysis.

OLEA has a helper function for preprocessing English text to remove user names and URLs and convert emoji to their textual descriptions.[7]

```
from olea.utils.preprocess_text import
    PreprocessText as pt
processed_text = pt.execute(user_data["
    raw_text"])
user_data["preprocessed_text"] =
    preprocessed_text
```

For example, the preprocessor converts "@username_1 Have you seen the video that @another_user made? 👀 🔥 https://fakelink.io" to "USER have you seen the video that USER made? eyes fire HTML".

Finally, users can write and share their own analysis functions, focusing on user-specified dimensions, as in 4.2.2. OLEA's code is modularized such that adding a new analysis requires enough Python knowledge to write the function, but not a detailed understanding of the entire codebase.

[7]Preprocessing scheme is described in Palmer et al. (2020).

4 Use Case Demonstrations

OLEA aids model development by providing an easy and comparable platform to test and build robust offensive language classifiers. We demonstrate three use cases: a) analysis on preloaded datasets (4.1), including model comparison (4.1.4), b) analysis on custom data (4.2), and c) sharing datasets and analysis functions (4.2).

4.1 User model performance evaluation using preloaded datasets

This section demonstrates how to use OLEA for detailed analysis of the strengths and weaknesses of existing offensive language detection models. For this demo, we use roBERTa-offensive (Barbieri et al., 2020), a pre-trained generic language model, fine-tuned on the SemEval2019 OffensEval dataset (Zampieri et al., 2019b). We use this model to make top-level predictions (offensive or not) for both COLD and HateCheck. Although they use different labels (offensive vs hateful), COLD and HateCheck align in their definitions by taking into account non-offensive uses of slurs and classifying derogatory text as offensive. Users explicitly run individual error analyses and specify whether a plot of the results should automatically be generated. These individual analysis functions show the model's performance with respect to a particular feature (i.e. an existing dataframe column, or a new one added by the function).

Each function returns two dataframes. The `metrics` dataframe contains a classification report for the analysis.[8] This dataframe uses OLEA's built in `Metrics` function, which is built upon and uses Scikit-learn's (Pedregosa et al., 2011) metrics library. The `plot_info` dataframe contains details of the analysis for the selected dimension, plus computed accuracy and the option to show textual examples. If `show_examples = True`, the function returns one randomly-selected incorrectly-classified instance for each value of the dimension being analyzed.[9] If the plot option is selected, the plots are built from the `plot_info` dataframe.

4.1.1 Generic analysis functions

Table 3 shows the classification report for roBERTa-offensive on COLD. Here, the classification report

[8]Appendix A provides more code examples for loading in data and starting generic analyses.

[9]The variable `show_examples` defaults to false to avoid accidental viewing of hateful or offensive language.

	N	Y	macro avg	weighted avg
precision	0.743	0.587	0.665	0.670
recall	0.502	0.803	0.652	0.643
f1-score	0.599	0.678	0.639	0.636
support	1072	944	2016	2016

Table 3: `Metrics` classification report for roBERTa-offensive on COLD, using `analyze_on` on the dimension of offensiveness. N=not offensive, Y=offensive.

Annotator Agreement	**Full**	**Partial**
Total	1431	585
Total Correct	1004	292
Accuracy	0.702	0.499
Incorrect Classification Example	an illegal is not an immigrant and illegals do take american jobs considering they are not americans.	USER yooo i was thinking bout that the other day lol.. you only really speaking of one person my n*ggah
Example's Predicted Label	N	Y
Example's Gold Label	Y	N

Table 4: `plot_info` report for roBERTa-offensive on COLD, using `check_anno_agreement` (full vs. partial), with randomly-selected examples.

provides F1, precision, and recall for the two categories of offensive and non-offensive, as well as the macros and weighted averages. This model performs better overall on offensive instances, with high recall, but shows much better precision for non-offensive instances. These reports can be easily modified to analyze subsets of the data.

Table 4 shows `plot_info` for roBERTa-offensive on COLD, using `check_anno_agreement`. The table shows accuracy for each category (full vs. partial) and one example incorrect prediction. Accuracy is much higher for instances with full agreement than for those with some disagreement. Offensiveness can be subjective, so it is useful to examine model performance on these different cases. Showing examples allows users to review difficult cases and may provide insights for model improvement.

4.1.2 COLD analysis

The next analysis (Fig. 2) is on the fine-grained COLD categories. roBERTa performs well on offensive tweets containing slurs and poorly on the reclaimed category (non-offensive tweets containing slurs). The model also performs poorly on offensive tweets containing adjectival nominalization. These insights suggest that the model relies too much on slurs for identifying offensive language.

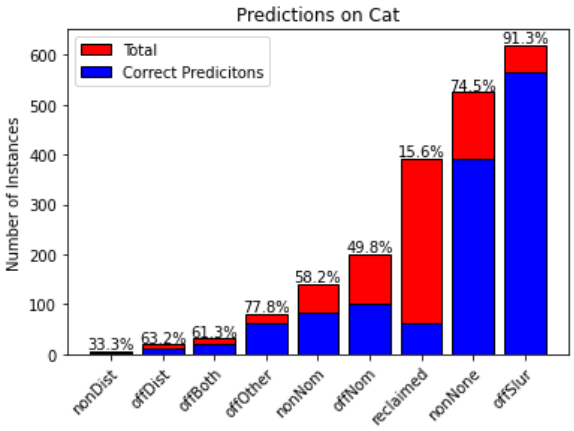


Figure 2: Results for roBERTa-offensive on COLD, focusing on fine-grained categories. Percent value above a bar shows percent accuracy for that category.

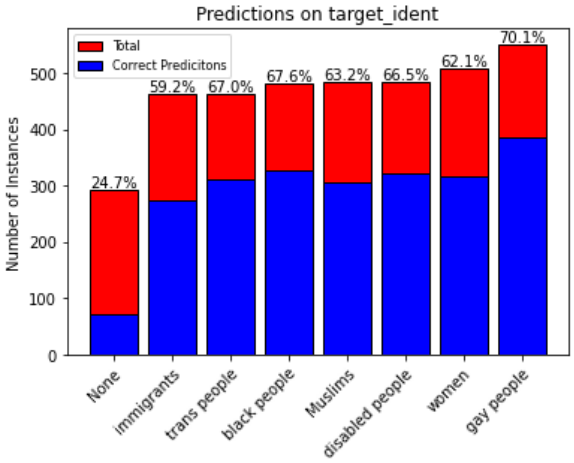


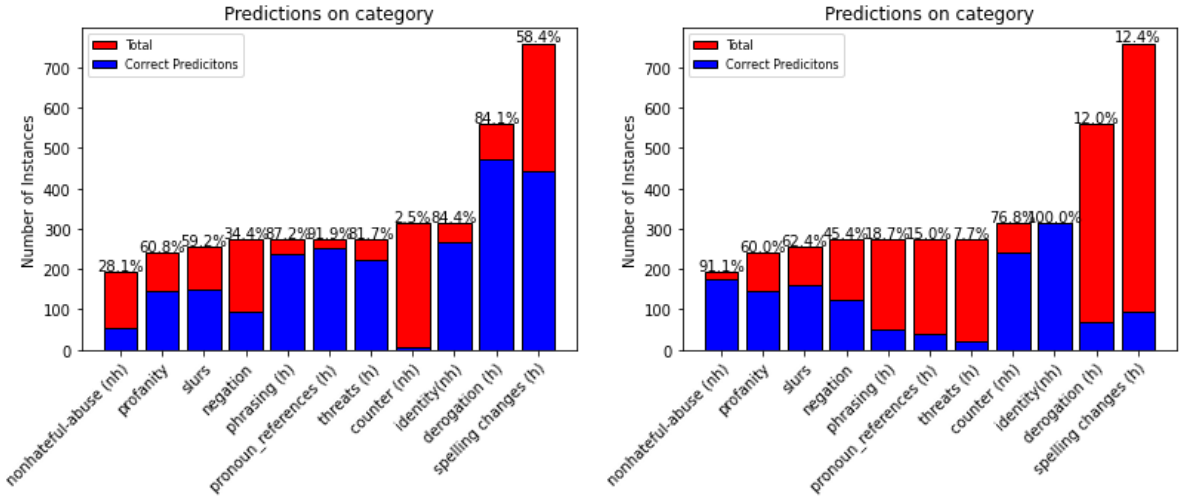
Figure 3: Results for roBERTa-offensive on HateCheck, focusing on identity of the target.

4.1.3 HateCheck analysis

HateCheck includes a column identifying the targeted group of the offensive language. This analysis can help identify which groups a model is more or less sensitive to. Figure 3 shows that the model does a better job of recognizing hateful speech when an identity is targeted than when one is not.

Figure 4a analyzes the performance of roBERTa-offensive on the fine-grained HateCheck categories; complete table is in Appendix B. Most instances of counter-speech (when hate speech is quoted or is referred to with a non-hateful effect) are incorrectly classified. An example of counter-speech is "I lost all respect for you when you said women are pathetic". Instances belonging to the categories of derogation, identity, threats, pronoun references, and phrasing are mostly correctly classified.

Figure 4: Comparison of HateXplain and roBERTa models on the fine-grained HateCheck categories. The category labels are followed by either (h) to denote that it is comprised of only hateful instances, (nh) for only non-hateful instances, or nothing to denote a mix of hateful and non-hateful instances



(a) Results for roBERTa-offensive on HateCheck, focusing on fine-grained categories.

(b) Results for HateXplain on HateCheck, focusing on fine-grained categories.

4.1.4 Model comparison

So far we have compared one model's performance on two different datasets. Next, we compare two different models on the same dataset. The second model is HateXplain (Mathew et al., 2021). Both models of course suffer when classifying data sourced from outside their training domain.[10,11] But we get a much truer sense of the models' out-of-domain capabilities when we look at the side-by-side analysis of the two models for the fine-grained HateCheck categories (Figures 4a and 4b). While roBERTa-offensive does not perform well on counter-speech, HateXplain correctly classifies most counter-speech instances. And while HateXplain struggles to recognize hateful expressions with spelling changes, roBERTa does much better.

4.2 OLEA as infrastructure: Extending functionality

OLEA is open-source[12] and has been designed to be extensible with new datasets and new analyses.

4.2.1 Analysis on custom data

The analysis methods described above can be easily applied to new corpora. The code below shows the process of loading the OLID dataset (Zampieri et al., 2019a) as a pandas dataframe. The user only needs to specify a path to the data and the relevant column headings. The `Dataset` class acts as a wrapper for the data loaded from disk and allows the user to access class utilities such as `generator()`, which in turn is helpful for accessing data in batches.

```
olid = pd.read_csv('data/olid/
    olid_levela.csv')
olid_dataset = Dataset(data = olid,
                features = 'Text',
                gold_column = 'label',
                text_column = 'Text')
data_gen = olid_dataset.generator(
    batch_size=64)
data = next(data_gen)
```

We can now submit model predictions, returning a `DatasetSubmissionObject` which can be used to conduct the generic analyses previously described; code in Appendix A.

4.2.2 Sharing datasets and analysis patterns

With just a bit of coding, interfaces new datasets can be added to the OLEA library more permanently, and for the benefit of all users.[13] We demonstrate again using OLID,[14] establishing the new `OLIDDataset` class which inherits from `Dataset`.

```
class OLIDDataset(Dataset) :
    text_column = 'Text'
```

[10] roBERTa-offensive reports an F1 of 0.78 on OLID but drops to 0.62 on HateCheck. HateXplain reports F1 of 0.69 on the HateXplain dataset, and drops to 0.37 on HateCheck.

[11] We map HateXplain's "offensive" and "hate speech" labels both to HateCheck's "hateful".

[12] https://github.com/alexispalmer/olea, Licensed under MIT License

[13] OLEA is not currently hosting datasets. The preloaded datasets are hosted via HuggingFace's datasets library.

[14] Note that we only consider OLID's "level-A" annotations.

```
    gold_column = 'label'
    features = ['Text','label_id']
def __init__(self, olid_csv_path:str) :
    self.olid_csv_path = olid_csv_path
    self._data = self._load_data()
def _load_data(self) -> pd.DataFrame:
    return pd.read_csv(self.
    olid_csv_path)
```

To accommodate the properties of the new dataset, we need to override some attributes of the `Dataset` class and to modify the method for loading data.

OLEA's scaffolding minimizes the amount of new code needed to add a new dataset, as well as automatically handling helper utilities such as mapping model predictions to the custom `Dataset` object. The library also runs sanity checks on submitted predictions before returning a `DatasetSubmissionObject`.

The advantage of using a native `DatasetSubmissionObject` is that users may run `Generic` analyses on it. However, if authors have a unique analysis that they wish to couple with their dataset, they may specify a special `Analysis` class that can operate on submissions. The class methods can be modified to accommodate patterns or properties specific to the dataset.

```
class OLIDAnalysis(object) :
    @classmethod
    def analyze_on(cls, submission:
    DatasetSubmissionObject, on:str) :
        '''
        Unique OLID analysis goes here!
        '''
        return get_metrics(submission,
    on)
```

5 Conclusion and Future Directions

This paper introduces OLEA, a tool for easy, in-depth error analysis and an infrastructure for sharing new datasets and analysis methods. OLEA helps researchers understand the strengths and weaknesses of their offensive language detection models. In the near term, we will continue to add new analysis methods and datasets, including methods for corpus exploration, and providing automatic trends and insights of model performance without users needing to run explicit analyses. Mid term, we plan to extend OLEA to additional languages, and eventually we would like to expand OLEA into a general error analysis library for a range of language classification tasks. Because OLEA is a convenient way for authors to share datasets and analyses, it is our hope that a community will develop around the library, and that models ultimately will improve as we learn more about what they can and cannot do.

Ethical Considerations. We acknowledge the ethical implications of releasing a tool that encourages accessing hate speech datasets where tweet author anonymity may not be ensured. This tool is to help researchers identify how their offensive language model can improve, with the intended benefit of more accurate identification of language that negatively affects vulnerable populations, and should not be used for any task that promotes or spreads the usage of hate speech or unnecessarily exposes people to hate speech. Even when used as intended and functioning correctly, users may react negatively to seeing offensive language, so we take steps to minimize exposure by defaulting `show_examples` to `False` during analysis, though researchers belonging to vulnerable populations might be more negatively affected. If the tool gives incorrect results, researchers might overestimate their model performance, which could directly hurt vulnerable populations depending on the deployments of the model. This tool relies on datasets that often categorize people based on their identity, and it supports analyses based on these categories. We believe that these categorizations are necessary for a granular understanding and examination of offensive language classification.

Limitations. OLEA is restricted in the languages that it can be applied to. Currently, for the substring analyses, it assumes that the language is delimited by spaces. Additionally, the library is primarily focused on providing error analyses for offensive language applications. Its use outside of this domain is not known or well-defined. Though we focus on error analysis for offensive language identification, the system makes no binding assumptions as to the proper definitions of offensive language and hate speech, nor does it assert (or assume) any difference between these two categories which can complicate cross-model comparison. Furthermore, this tool is just an analysis tool; it does not address concerns regarding language drift and other sociolinguistic biases that may be present within a user's dataset, nor does it address any annotator biases present in original datasets.

Acknowledgments. Thanks to the anonymous reviewers for useful feedback. Thanks also to Cutter Dalton for testing, and to members of the 2021-2023 CLASIC cohort for helpful discussions!

References

Francesco Barbieri, José Camacho-Collados, Leonardo Neves, and Luis Espinosa Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *CoRR*, abs/2010.12421.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130, Austin, Texas. Association for Computational Linguistics.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, Inga Kartoziya, and Michael Granitzer. 2020. I feel offended, don't be abusive! implicit/explicit messages in offensive and abusive language. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6193–6202, Marseille, France. European Language Resources Association.

Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. 2021. Latent hatred: A benchmark for understanding implicit hate speech. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 345–363, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Komal Florio, Valerio Basile, Marco Polignano, Pierpaolo Basile, and Viviana Patti. 2020. Time of your hate: The challenge of time in hate speech detection on social media. *Applied Sciences*, 10(12):4180.

Eduard Hovy. 2022. Rediscovering the need for representation and knowledge. ACL 2022 - 60th Annual Meeting of the Association for Computational Linguistics.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Chris J Kennedy, Geoff Bacon, Alexander Sahn, and Claudia von Vacano. 2020. Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application. *arXiv preprint arXiv:2009.10277*.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14867–14875.

Angelina McMillan-Major, Amandalynne Paullada, and Yacine Jernite. 2022. An interactive exploratory tool for the task of hate speech detection. In *Proceedings of the Second Workshop on Bridging Human–Computer Interaction and Natural Language Processing*, pages 11–20, Seattle, Washington. Association for Computational Linguistics.

Alexis Palmer, Christine Carr, Melissa Robinson, and Jordan Sanders. 2020. COLD: Annotation scheme and evaluation data set for complex offensive language in English. *Journal for Language Technology and Computational Linguistics*, 34(1):1–28.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Paul Röttger, Bertram Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet B Pierrehumbert. 2020. Hatecheck: Functional tests for hate speech detection models. *arXiv preprint arXiv:2012.15606*.

Maarten Sap, Swabha Swayamdipta, Laura Vianna, Xuhui Zhou, Yejin Choi, and Noah A Smith. 2021. Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. *arXiv preprint arXiv:2111.07997*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Bertie Vidgen, Dong Nguyen, Helen Margetts, Patricia Rossini, and Rebekah Tromble. 2021. Introducing CAD: the contextual abuse dataset. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2289–2303, Online. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017a. Understanding abuse: A typology of abusive language detection subtasks. *arXiv preprint arXiv:1705.09899*.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017b. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of Abusive Language: the Problem of Biased Datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 602–608, Minneapolis, Minnesota. Association for Computational Linguistics.

Wenjie Yin and Arkaitz Zubiaga. 2021. Towards generalisable hate speech detection: a review on obstacles and solutions. *PeerJ Computer Science*, 7:e598.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

A Code Examples

In this appendix, we provide code snippets corresponding to Section 4.

A.1 Preliminaries

Installation

```
pip install olea
```

Import Statements

```
import pandas as pd
from olea.data.cold import COLD,
    COLDSubmissionObject
from olea.data.hatecheck import
    HateCheck
from olea.analysis.cold import
    COLDAnalysis
from olea.analysis.generic import
    Generic
from olea.analysis.hatecheck import
    HateCheckAnalysis
from olea.utils import preprocess_text
```

A.2 Code examples for Section 4.1

Download and Preprocess COLD text:

```
cold = COLD()
pt = preprocess_text.PreprocessText()
processed_text = pt.execute(cold.data()[
    "Text"])
cold.data()["Text"] =processed_text
```

Create predictions using roBERTa-offensive

```
from transformers import AutoTokenizer,
    AutoModelForSequenceClassification
from transformers import
    TextClassificationPipeline

 link = "cardiffnlp/twitter-roberta-base
    -offensive"
 tokenizer = AutoTokenizer.
    from_pretrained(link)
        model =
    AutoModelForSequenceClassification.
    from_pretrained(link)
        #Create Pipeline for Predicting
        pipe =
    TextClassificationPipeline(model=
    model, tokenizer=tokenizer)
        preds = pd.DataFrame(pipe(list(
    cold.data()["Text"])))["label"]
```

Create Submission Objects:

```
cold_so = cold.submit(
    batch = cold.data(),
    predictions = preds,
    map = {"LABEL_0": 'N', 'LABEL_1': "Y
    "})
```

```
hc_so = hc.submit(
    batch = hc.data(),
    predictions = preds,
    map = {"LABEL_0": 'non-hateful', '
    LABEL_1': "hateful"})
```

Generate Table 3:

```
plot_info, metrics = Generic.analyze_on(
    cold_so,
    'Cat',
    show_examples = False,
    plot = False)
```

Generate Table 3 and Save plot to file:

```
plot_info, metrics = Generic.analyze_on(
    cold_so,
    'Cat',
    show_examples = False,
    plot = False,
    savePlotToFile= "cold_cats.png")
```

Generate Table 4:

```
plot_info, metrics = Generic.
    check_anno_agreement(cold_so, ["Off1
    ","Off2","Off3"],show_examples =
    True, plot = False)
```

Generate Figure 2:

```
plot_info, metrics =COLDAnalysis.
    analyze_on(
    cold_so,
    'Cat',
    show_examples = False,
    plot = True)
```

Generate Figure 3:

```
plot_info, metrics = Generic.analyze_on(
    hc_so,
    'target_ident')
```

Generate Figure 4a, Figure 4b, and Table 5:

```
plot_info, metrics = HateCheckAnalysis.
    analyze_on(
    hc_so,
    'category')
```

A.3 Code examples for Section 4.2

Run analysis functions on local custom data:

```
predictions = model.predict(data)
submission = olid_dataset.submit(
            batch = data,
            predictions = predictions,
            map = {1:'OFF', 0:'NOT'})
# performance on AAVE
Generic.aave(submission)
# performance on texts containing
    substring 'female'
Generic.check_substring(submission, "
    female")
```

Submit predictions for newly-established dataset class `OLIDDataset`:

```
olid = OLIDDataset('data/olid.csv')
datagen = olid.generator(64)
data = next(datagen)
preds = model.predict(data)
map = {'OFF' : 1.0 , 'NOT' : 0.0}
submission = olid.submit(batch = data,
               predictions = preds,
               map = map)
```

B Full Results Table for Figure 4a

category	Metrics	precision	recall	f1-score	support
counter (nh)	hateful	0.000	0.000	0.000	0
counter (nh)	non-hateful	1.000	0.038	0.074	314
counter (nh)	macro avg	0.500	0.019	0.037	314
counter (nh)	weighted avg	1.000	0.038	0.074	314
derogation (h)	hateful	1.000	0.805	0.892	560
derogation (h)	non-hateful	0.000	0.000	0.000	0
derogation (h)	macro avg	0.500	0.403	0.446	560
derogation (h)	weighted avg	1.000	0.805	0.892	560
identity(nh)	hateful	0.000	0.000	0.000	0
identity(nh)	non-hateful	1.000	0.892	0.943	315
identity(nh)	macro avg	0.500	0.446	0.471	315
identity(nh)	weighted avg	1.000	0.892	0.943	315
negation	hateful	0.295	0.236	0.262	140
negation	non-hateful	0.335	0.406	0.367	133
negation	macro avg	0.315	0.321	0.315	273
negation	weighted avg	0.315	0.319	0.313	273
nonhateful-abuse (nh)	hateful	0.000	0.000	0.000	0
nonhateful-abuse (nh)	non-hateful	1.000	0.339	0.506	192
nonhateful-abuse (nh)	macro avg	0.500	0.169	0.253	192
nonhateful-abuse (nh)	weighted avg	1.000	0.339	0.506	192
phrasing (h)	hateful	1.000	0.868	0.929	273
phrasing (h)	non-hateful	0.000	0.000	0.000	0
phrasing (h)	macro avg	0.500	0.434	0.465	273
phrasing (h)	weighted avg	1.000	0.868	0.929	273
profanity	hateful	0.601	1.000	0.751	140
profanity	non-hateful	1.000	0.070	0.131	100
profanity	macro avg	0.800	0.535	0.441	240
profanity	weighted avg	0.767	0.613	0.492	240
pronoun-references (h)	hateful	1.000	0.908	0.952	273
pronoun-references (h)	non-hateful	0.000	0.000	0.000	0
pronoun-references (h)	macro avg	0.500	0.454	0.476	273
pronoun-references (h)	weighted avg	1.000	0.908	0.952	273
slurs	hateful	0.593	0.778	0.673	144
slurs	non-hateful	0.515	0.306	0.384	111
slurs	macro avg	0.554	0.542	0.528	255
slurs	weighted avg	0.559	0.573	0.547	255
spelling changes (h)	hateful	1.000	0.549	0.709	760
spelling changes (h)	non-hateful	0.000	0.000	0.000	0
spelling changes (h)	macro avg	0.500	0.274	0.354	760
spelling changes (h)	weighted avg	1.000	0.549	0.709	760
threats (h)	hateful	1.000	0.810	0.895	273
threats (h)	non-hateful	0.000	0.000	0.000	0
threats (h)	macro avg	0.500	0.405	0.447	273
threats (h)	weighted avg	1.000	0.810	0.895	273

Table 5: The `Metrics` classification report for roBERTa-offensive on HateCheck