ferret Documentation

Release 0.2.4

Giuseppe Attanasio

Feb 05, 2023

CONTENTS:

CHAPTER

ONE

FERRET

A python package for benchmarking interpretability techniques.

- Free software: MIT license
- Documentation: https://ferret.readthedocs.io.

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from ferret import Benchmark

model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased")
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

bench = Benchmark(model, tokenizer)
explanations = bench.explain("You look stunning!")
evaluations = bench.evaluate_explanations(explanations)

print(evaluations)
```

1.1 Features

ferret builds on top of the transformers library. The library supports explanations using:

- Gradients
- Integrated Gradinets
- Gradient x Input word embeddings
- SHAP
- LIME

and evaluate explanations via:

Faithfulness measures.

- AOPC Comprehensiveness
- AOPC Sufficiency
- Kendall's tau correlation with leave-one-feature out

Plausibility measures.

- AUPRC soft score plausibility
- Token f1 hard score plausibility
- Token IOU hard score plausibility

TODOs

- Possibility to run on select device ("cpu", "cuda")
- Sample-And-Occlusion explanations
- Discretized Integrated Gradients: https://arxiv.org/abs/2108.13654

1.2 Visualization

```
bench = Benchmark(...)

explanations = ...
bench.show_table(explanations)

evaluations = bench.evaluate_explanations(explanations)
bench.show_evaluation_table(evaluations)
```

1.3 Datasets evaluations

```
bench = Benchmark(...)

hatexdata = bench.load_dataset("hatexplain")
dataset_explanations = bench.generate_dataset_explanations(hatexdata)
dataset_evaluations = bench.evaluate_dataset_explanations(dataset_explanations)
bench.show_dataset_evaluation_table(dataset_evaluations)
```

1.4 Credits

This package was created with Cookiecutter and the *audreyr/cookiecutter-pypackage* project template.

- Cookiecutter: https://github.com/audreyr/cookiecutter
- *audreyr/cookiecutter-pypackage*: https://github.com/audreyr/cookiecutter-pypackage

Logo and graphical assets made by Luca Attanasio.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install ferret, run this command in your terminal:

```
$ pip install -U ferret-xai
```

This is the preferred method to install ferret, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for ferret can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/g8a9/ferret
```

Or download the tarball:

```
$ curl -OJL https://github.com/g8a9/ferret/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use ferret in a project:

```
import ferret
```

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/g8a9/ferret/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

ferret could always use more documentation, whether as part of the official ferret docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/g8a9/ferret/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *ferret* for local development.

1. Fork the *ferret* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/ferret.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv ferret
$ cd ferret/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 ferret tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/g8a9/ferret/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_ferret
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Giuseppe Attanasio <giuseppeattanasio6@gmail.com>
- Eliana Pastor <eliana.pastor@centai.eu>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 0.1.0 (2022-05-30)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search