Report date 11/11/2024

Paper report:

**Overview:**

This is a very interesting paper for our project because it helps us to focus on the needs and common operators being used in ML pipelines. It is very fun and enjoyable to read, a lot of interesting big numbers.

The paper consists in an analysis of notebooks, namely: 6Mln from GitHub written in 2017 and 2019, 2Mln pipelines from Microsoft ML.NET and the source code of 900+ releases of 12 DS libraries.

The research focused on Python only files, it can be a good enough approximation since according to one of the authors' presentations (at DSDS) over 90% of DS pipelines written in 2019 are in Python.

The set has been cleaned both in terms of duplicated files and also in terms of empty lines.

The analysis included something like 7.68B AST entries, 13M functions, 1M classes, 10.7M if blocks and 15.4M for loops.

Regarding the dataset from GitHub 2019 interesting facts are the linear cells (88%) and completely linear cells(83%). Linear cells are cells that do not contain any conditional statement (if, for, while…) and completely linear are cells that do not contain any conditional statement nor classes or function. At a file level 29% of the notebooks are completely linear and 35% are linear. The researchers concluded that most of the content is about linear orchestration of data manipulation libraries.

An import analysis is made, which analysis the “import” and “import from ” statements in each file, be aware, the implicitly imported libraries are not counted (Pandas internally uses Numpy, but unless the uses also separately imports Numpy this will not be recorded).

The extensively most used libraries are numpy (70%), matplotlib (60%), pandas (50%), sklearn (30%) and scipy (18%). Apart from the first 4 libraries that are growing in usage, the growth of NLP and DL libraries is also seen. A libraries correlation analysis is also done but doesn't highlight anything that good sense would (Pandas is not used when image recognition libraries are used, libs to work on tabular data aren't used in correlations to libs that work on image data…). The coverage metric is interesting and shows that we'd need 10 different libs to cover 40% of the notebooks and 100 libs to cover 70% of the notebooks.

Now to the most important part, the pipelines analysis. This analysis has been performed on scikit-learn pipelines only due to their popularity and extensive usage and on Microsoft own pipelines. In total 140K pipelines have been harvested from the GitHub notebooks and 2M unique pipelines have been extracted from Microsoft code. Most of the pipelines aren't composed by more than 10 operators (when multiple sub pipelines are used their length isn't summed so the reported values are rather a lower bound). The author have also notice an increase in the pipelines complexity, both in terms of length and different operators usage.

The top 5 of scikit transformers are:

* StandardScaler
* CountVectorizer
* TfidfTransformer
* PolinomialFeatures
* TfidfVectorizer (PCA in GitHub 2017)

The top 5 learners are:

* LogisticRegression
* MultinomialNB
* SVC
* LinearRegression
* RandomFortestClassifier

In Microsoft pipelines the top 5 transformers are:

* OneHotEncoder
* TfidfVectorizer
* Imputer
* Tokenizer
* CountVectorizer

And the top learners are:

* Gradient Boosting
* Random Forest
* SDCA
* PoissonRegression
* Averaged Perceptron

Top 10 operators would cover 30% of notebooks found on GH and 80% of the analyzed pipelines from Microsoft. Top 100 operators would cover 80% of GH pipelines.

This tells us that we need to implement unlearning on relatively few components.

A release analysis of the most used libraries has been performed but it is not very useful for our purpose and expected results have been found:

* After an initial phase when many releases are made quite quickly the project settle to an stable release pace

The number of files, classes and functions has also been analyzed with no great surprises here too.

This paper is very interesting and useful for us.

Another paper is cited here: Software Engineering for Machine Learning: A Case Study. It should include an analysis of the same sector but conducted by speaking with data scientists and engineers rather than by analyzing the code. That paper will also be analyzed.

**Experimental metrics and scenarios:**

* # Files
* # Cells
* # Users
* Libraries usage
* Libraries correlation
* Pipeline operators usage (transformers and learner)
* Libraries release analysis
* Libraries # files, # classes and # functions

**Limitations:**

The analysis could be biased since it is only focusing on publicly available files, for instance they may include a lot of learning exercises, they've tried to limit this by also including Microsoft pipelines code.

**Repo:**

NA

**Authors:**

Fotis Psallidas et alii. (Microsoft GSL)

**Year:**

2019

Please note that the paper available and annotated is the 2019 version. The one discussed at SIGMOD has been released in 2022 instead.

The main differences are a distinct analysis of implicit vs explicit pipelines and the inclusion of GitHub data from 2020 and not only from 2017 and 2019. The final conclusions are the same as well as the figures.