Simple Machine Learning Techniques For Binary Diffing (with Diaphora)

Joxean Koret (Activision)

Introduction

- Machine Learning/Artificial Intelligence is a hot topic in academia for Binary Code Similarity Analysis or, as the information security industry calls it, Binary Diffing (bindiffing for short).
- In this talk I will try to discuss how a machine learning (ML) engine was added to Diaphora, the initial steps, what problems were found, which dataset(s) were used, how they were built & cleaned up, how it works in Diaphora, how it enhanced Diaphora, etc.

Introduction

- Why?
- Prior knowledge: Industry vs Academia
- Ideas:
  1) Train a *local* model
  2) Train a **gigantic** model
  3) Train a highly specialized model
- Conclusions

Why?

- The basic idea of this research was to try to apply machine learning techniques with the aim to improve the results of Diaphora for real world use cases.
- My reasoning was: “Diaphora uses a myriad of heuristics and little ‘tricks’ here and there, but there is room for improvement, perhaps I can use ML techniques for this?”
- Disclaimer: I’m not an ML guy, just a reverse engineer.

Why?

- Binary diffing (for 2 binaries) can be summarized in the following steps:
  1) Mapping: Pairs of functions are matched between 2 binaries.
  2) Classification: The matches are classified into “good” or “bad” matches, and a similarity ratio is, optionally, calculated.
- Now, how can I improve any of these 2 steps?
  - I honestly don’t know how to apply ML techniques to improve mapping (in a way that scales, without doing, say, a Cartesian product of the adjacency matrices of the call graphs, for example).
  - But classification is actually one of the strongest point of ML techniques, isn’t it?
- Cool. Next step? Check prior knowledge in this field.
  - Spoiler: Meh.

Industry vs Academia

- In the reverse engineering industry most people use Diaphora, BinDiff or Ghidriff for doing binary diffing. No one of these tools apply ML techniques (until now, I guess), so nothing to learn from the industry here.
- In the academia, however, it’s the most researched topic.
- Apparently. It seems. I guess.

Academia

- Almost every single paper I have read lately coming from the academia regarding Binary Code Similarity Analysis (BCSA), BinDiffing, talks about applying Machine Learning/Artificial Intelligence techniques to this process (for classification, not for mapping).
- Tools like DeepBinDiff, αdiff or BinDiffNN, or techniques like asm2vec, are some simple examples.

Academia

- Unfortunately, most of the academic tools and techniques published about binary diffing are just proof-of-concepts (“Works For Me ™”), or don’t properly scale (they do Cartesian products or are based on adjacency matrices comparison), or are paper-ware (“write and forget”), or the dataset was a toy, or there is neither datasets nor code available, or are too “classical” (they only focus on graphs, assembly or even bytes!) for real world today’s use-cases, etc...
- In short: I have tried to use academic research, but I found almost nothing I consider, personally, useful. Unfortunately.
- Next step? Think about what could possibly work.

Ideas

The following is the list of ideas I wanted to give a try:

1) Training a *local* model.
2) Training a **gigantic** model.
3) Training a highly specialized model.

Any guess about which techniques work, won't work at all, or might work but require specific stuff?

Let's first discuss them a bit more in detail...

Training a local model

Training a local model

- This is how, on my mind, this idea could work:
  - Diaphora uses a number of heuristics that produce highly reliable results.
  - By using the matches produced by Diaphora that are good, I could train a classifier of what are good matches, and then use the model to determine if future matches found by not so reliable heuristics are good or bad.
- What do you think about this idea?
  - Yay or nay?

Training a local model

- This is how, on my mind, this idea could work:
  - Diaphora uses a number of heuristics that produce highly reliable results.
  - By using the matches produced by Diaphora that are good, I could train a classifier of what are good matches, and then use the model to determine if future matches found by not so reliable heuristics are good or bad.
- What do you think about this idea?
  - Yay or nay? **Answer: It's probably a bad idea. Let's see why.**

Training a local model

- Let’s see how it works and why it is a bad idea:
  - Diaphora finds reliable matches.
  - We train a classifier using these matches.
  - Then we use Diaphora to classify future new matches and use the classifier we trained using Diaphora as classifier to determine if the results are good.
- Do you see the problem already? I was trying to learn a model that I already have coded.
  - There is no need to re-learn what I already have properly coded.
  - Naturally, the model would only find whatever Diaphora was already finding.
  - And the model would also add false positives due to generalization.
- This is the worst idea ever. At least the worst if only 1 classifier is used.

Training a gigantic model

Training a gigantic model

- When I finally realised that it was probably a bad idea to train a local model, I decided that, perhaps, the best solution would be to train a model using lots of binaries for different targets, at least for “only”:
  - Gcc, clang and msvc compiled targets.
  - For Linux, Windows and Mac operating systems.
  - For x86, ARM, MIPS, PPC and Risc-V CPUs.
- After all, it’s a proved to work thing, right? And it’s as simple as...
  - Generating a huge amount of data.
  - Using some algorithm to generalize a model from the data.
  - Using the generalized model to make predictions.
- OK, now, what do you think about this idea?
  - Yay or nay?

Training a gigantic model

- When I finally realised that it was probably a bad idea to train a local model, I decided that, perhaps, the best solution would be to train a model using lots of binaries for different targets, at least for “only”:
  - Gcc, clang and msvc compiled targets.
  - For Linux, Windows and Mac operating systems.
  - For x86, ARM, MIPS, PPC and Risc-V CPUs.
- After all, it’s a proved to work thing, right? And it’s as simple as...
  - Generating a huge amount of data.
  - Using some algorithm to generalize a model from the data.
  - Using the generalized model to make predictions.
- OK, now, what do you think about this idea?
  - Yay or nay? **Answer: Maybe. Do you have *lots* of resources? Then maybe.**

Building Datasets

Finding datasets

- Let’s see the problems of this approach, step by step.
- The first thing one needs to train a model, naturally, is data.
- In this case, “data” are “binaries”. So one needs a binaries dataset to be able to start.
  - The biggest it’s, the better. I guess.
- Which datasets of binary programs are available?

Finding datasets

- There aren't so many datasets of binaries. The only 2 ones that I know (that doesn't include malicious and/or obfuscated software) are the following ones:
  - Johns Hopkins APL ALLSTAR: Public repo of Debian sources, binaries, compiler artefacts, etc...
    - https://allstar.jhuapl.edu/
  - Cisco Talos Binary Function Similarity: The dataset used for the paper "How Machine Learning Is Solving the Binary Function Similarity Problem".
    - https://github.com/Cisco-Talos/binary_function_similarity/tree/main/Binaries

Datasets: ALLSTAR

- This dataset is huge: 1.1 TB in total.
- It contains around 32,000 Debian packages.
- It has around ~20k executables for x86, amd64, ARM, PPC, MIPS and s390x.
- Which means that even if I use some disks with a few terabytes for downloading and analysing this dataset, I still need to run IDA (and export with Diaphora) some ~20k programs.
  - Sorry, I don’t have the required resources. At all.
  - So… I had to search for another option.
- BTW, naturally, it has no binary for Windows, Android or Macos/iOS, neither Risc-V support, nor support for, say, the MSVC compiler.

Datasets: Cisco Talos

- This dataset is considerably smaller. It's split in 3 subsets:
  - "Dataset Vulnerability".
  - "Dataset 1": Clamav, curl, nmap, openssl, unrar, z3 and zlib.
  - "Dataset 2": Binutils, diffutils, coreutils, findutils, ImageMagick, sqlite, putty, gmp, libmicrohttpd and tomcrypt.
- Without entering into details, I decided to just use "Dataset 2".
  - Covers enough binaries to be relevant and it's possible for me to export with Diaphora all of them in a reasonable amount of time.
  - It's around 4047 binaries exported to SQLite databases.
  - 45 gigabytes of data to analyse. Big enough for me.
- BTW, it doesn't have binaries for anything that isn't Linux, or CPUs other than ARM/x86/MIPS/PPC, or the MSVC compiler.
- Next step? Build a dataset out of the raw data.

Building a new dataset

- I cannot work properly with 4,047 SQLite database files, I need to have everything into a single “thing”.
- Also, I don’t need everything Diaphora exports, I just need a small subset of the artefacts exported by it for the “functions” table.
- But… which artefacts?

Building a new dataset: artefacts

- After some trial & tests, I decided to use the fields shown in the picture below.
- The fields don’t contain the real values, but a similarity degree, and the minimum and maximum values.
- Similarity ratios are, basically, calculated like this: 1 - (abs(x - y) / max(x, y))
- For text fields, the similarity ratio is calculated using Python’s SequenceMatcher.quick_ratio().

```
46  FIELDS = ["nodes", "edges", "indegree", "outdegree", "cc",
47    "primes_value", "clean_pseudo", "pseudocode_primes", "strongly_connected",
48    "strongly_connected_spp", "loops", "constants", "source_file"
49  ]
50
51  NUM_FIELDS = ["nodes", "edges", "indegree", "outdegree", "cc",
52    "strongly_connected", "loops"
53  ]
54
55  PRINT_FIELDS = ['ratio', 'nodes', 'min_nodes', 'max_nodes', 'edges', 'min_edges',
56    'max_edges', 'indegree', 'min_indegree', 'max_indegree', 'outdegree',
57    'min_outdegree', 'max_outdegree', 'cc', 'min_cc', 'max_cc', 'primes_value',
58    'clean_pseudo', 'pseudocode_primes', 'strongly_connected',
59    'min_strongly_connected', 'max_strongly_connected', 'strongly_connected_spp',
60    'loops', 'min_loops', 'max_loops', 'constants', 'source_file',
61  ]
```

Building a new dataset

- After choosing the proper fields and playing a bit generating small datasets it was time to play with bigger ones.
- Running a Python script I built to cross-compare all the Diaphora's exported SQLite databases and generate a dataset with the previously mentioned fields turned out to be… not very practical for an individual:
  - Tip: It means doing 4,047 x 4,047 operations (16,378,209 operations).
- Estimating the times it took for just 100 binaries doing so for the whole Cisco Talos Dataset-2, it would take me around 2 months for just generating the dataset. Doable. But not for an individual like me.

Building a new dataset

- I needed a partial solution, because “the best solution” (building a huge dataset) is absolutely not practical for me.
- As so, I thought about other approaches:
  - Building a subset of the dataset that is significant enough.
  - Building specialised datasets for specialised use-cases.
  - Building a dataset comparing only binaries that match around 99% of the functions (by address or by name, when there are symbols).
    - More details on this too later.

Building subsets of datasets

- I started trying to build datasets for:
  - Only diffutils.
  - Only coreutils.
  - Only putty binaries.
  - Only “ls” binaries.
  - Only another random subset of binaries.
- Guess what happened?
  - The generated datasets had good accuracy against the training data, but they were performing bad against the real data. Surprise.
  - Let’s see some examples...

Building subsets of datasets

- Example using a subset of the Cisco Talos dataset specific to diff-utils.
- At the top, the results against the training dataset.
- At the bottom, the results against a larger dataset involving multiple different binary types (from the same dataset)

```
Loading model test.pkl
Loading dataset diff-utils.csv
Evaluating the model against dataset diff-utils.csv...
Accuracy : 0.913242135342694
Precision: 0.9595621446410543
Recall   : 0.6566963352213323
F1       : 0.779752683935251
Done in 1.5063505172729492 second(s)
```

```
Loading model test.pkl
Loading dataset dataset.csv
Evaluating the model against dataset dataset.csv...
Accuracy : 0.8310285387990463
Precision: 0.599016790092377
Recall   : 0.553809948666227
F1       : 0.5755270009238262
Done in 4.850617170333862 second(s)
```

Building subsets of datasets

- Example using a subset of the Cisco Talos dataset specific to Putty binaries.
- At the top, the results against the training dataset.
- At the bottom, the results against a larger dataset involving multiple different binary types (from the same dataset)

```
Loading model test.pkl
Loading dataset small-psftp.csv
Evaluating the model against dataset small-psftp.csv...
Accuracy : 0.9767079437284137
Precision: 0.9472188388144539
Recall   : 0.9409155071587013
F1       : 0.9440566514921598
Done in 0.09028029441833496 second(s)
```

```
Loading model test.pkl
Loading dataset dataset.csv
Evaluating the model against dataset dataset.csv...
Accuracy : 0.812887057712804
Precision: 0.5529932397968347
Recall   : 0.49765959372615953
F1       : 0.5238693168090761
Done in 4.862972021102905 second(s)
```

Building subsets of datasets

- Example using a subset of the Cisco Talos dataset specific to Putty binaries.
- At the top, the results against the training dataset.
- At the bottom, the results against the diff-utils dataset previously generated.

```
Loading model test.pkl
Loading dataset small-psftp.csv
Evaluating the model against dataset small-psftp.csv...
Accuracy : 0.9767079437284137
Precision: 0.9472188388144539
Recall   : 0.9409155071587013
F1       : 0.9440566514921598
Done in 0.09028029441833496 second(s)
```

```
Loading model test.pkl
Loading dataset diff-utils.csv
Evaluating the model against dataset diff-utils.csv...
Accuracy : 0.7878073360945035
Precision: 0.5719570742728043
Recall   : 0.36825979835080414
F1       : 0.4480430057573294
Done in 1.450483798980713 second(s)
```

Choosing an algorithm

- You might think that, maybe, the problem wasn’t the data but the algorithm.
- This is something that I thought. So, just in case, I gave it a try and decided to try a lot of different classifiers (and even regressors).
- Results? No, the problem wasn’t the classifier at all.
- Let’s see some results...

Choosing an algorithm (dataset.csv)

Algorithm Name	Accuracy on training data	Accuracy on testing dataset	Deviation	Duration
ExtraTreesClassifier	0,97	0,91	0,01	~01:15:00
RandomForestClassifier	0,97	0,92	0,01	+2 hours
BaggingClassifier	0,96	0,91	0,01	~01:10:00
DecisionTreeClassifier	0,95	0,88	0,02	12 minutes
KNeighborsClassifier	0,94	0,88	0,01	~2:30:00
MLPClassifier	0,91	0,9	0,01	~5 hours
GradientBoostingClassifier	0,9	0,91	0,01	~3 hours
AdaBoostClassifier	0,89	0,9	0,02	~40 minutes
SGDClassifier	0,85	0,85	0,05	10 minutes
LinearDiscriminantAnalysis	0,85	0,88	0,02	2 minutes
LogisticRegression	0,85	0,88	0,01	3 minutes
BernoulliNB	0,83	0,8	0,02	1 minute
GaussianNB	0,82	0,86	0,02	~2 minutes
BayesianRidge	0,31	0,43	0,06	1 minute

Choosing an algorithm (Putty dataset)

Algorithm Name	Accuracy on testing data	Accuracy on training dataset	Deviation	Duration (sec)
MLPClassifier	0,87	0,91	0,01	21745
KNeighborsClassifier	0,86	0,88	0,01	17657
AdaBoostClassifier	0,85	0,9	0,02	13608
RandomForestClassifier	0,9	0,92	0,01	10326
ExtraTreesClassifier	0,9	0,91	0,01	9815
BaggingClassifier	0,89	0,91	0,02	2170
GradientBoostingClassifier	0,86	0,91	0,01	1894
LogisticRegression	0,83	0,88	0,01	1221
DecisionTreeClassifier	0,89	0,88	0,02	310
BernoulliNB	0,74	0,8	0,02	216
SGDClassifier	0,82	0,86	0,02	159
GaussianNB	0,73	0,86	0,02	131
LinearDiscriminantAnalysis	0,82	0,88	0,02	123

Choosing an algorithm (diff-utils against dataset.csv)

Algorithm Name	Accuracy on training dataset	Accuracy on testing data	Precision	Recall	F1	Training	Validation
MLPClassifier	0,88	0,864126110440244	0,72300897599	0,55617944838	0,62871540389	9000	2
GradientBoostingClassifier	0,87	0,869471201100845	0,79399703251	0,49820267142	0,61224506010	9301	6
RandomForestClassifier	0,91	0,8579833348712	0,70618835148	0,53669976854	0,60988775849	9056	50
BaggingClassifier	0,91	0,850788123387747	0,68025950341	0,52572184077	0,59308923817	4269	9
GaussianNB	0,81	0,816048377282897	0,54759693457	0,63659373908	0,58875109870	107	2
AdaBoostClassifier	0,86	0,865403208659127	0,81921531552	0,44818133622	0,57938769881	2342	12
DecisionTreeClassifier	0,91	0,842926948644604	0,65091175981	0,51890427609	0,57745984865	537	3
LogisticRegression	0,84	0,854233614027486	0,74283663432	0,45162406092	0,56173116241	5971	N/A
LinearDiscriminantAnalysis	0,84	0,846752894198765	0,70086052654	0,45205399743	0,54961022015	247	N/A
SGDClassifier	0,84	0,852440996056374	0,78310598513	0,39639823111	0,52635984341	1100	N/A
BernoulliNB	0,85	0,844958938952031	0,82583570262	0,31736749550	0,45852462461	119	N/A

Choosing an algorithm?

- After multiple other tests with multiple other datasets… I finally decided that the algorithm wasn't really the problem here.
- And, finally, decided that a decision tree classifier, actually, worked more than good enough for what I want:
  - I want it to be fast at validation and, maybe, training.
  - I want it to be reliable enough, but no need to be 100% bulletproof.
    - Remember that this ML engine won't be used alone: it will be used by Diaphora, and it will take care of removing potential FPs.
  - As I'm not really building huge datasets overfitting seems not to be a real problem for now.
  - A random forest classifier performs just slightly better but is considerably much slower. I don't want Diaphora to be significantly slower.

Building datasets

- So, after discussing the problems I had with *big* datasets, and briefly discussing that algorithms aren't actually a problem in this scenario…
- What else can I do?
- Does this testing mean that I cannot, as an independent RE tools developer train a model I can distribute that would work for every use-case?
  - Yes, it won't work. Unless I have the help of someone with huge resources.
  - And even in this case, the model to be distributed would probably be gigabytes in size, so probably not very practical (and I won't be able to just put it in Github).
- Does this mean that ML for Diaphora is not viable? Nope. Let's continue...

Training specialized models

Using specialized models

- This the 3rd idea I mentioned that I wanted to give a try.
- Actually, it wasn't an idea I had at first. This is an idea that arose at the time I decided that I don't have the resources to properly develop idea #2.
- Basically, train a model with 2 or more binaries with and without symbols, that are actually the same target, but for different versions, maybe, and then use the trained model to try to improve function matching for future later versions of that product.
- OK, now, what do you think about this idea?
  - Yay or nay?

Using specialized models

- This the 3rd idea I mentioned that I wanted to give a try.
- Actually, it wasn't an idea I had at first. This is an idea that arose at the time I decided that I don't have the resources to properly develop idea #2.
- Basically, train a model with 2 or more binaries with and without symbols, that are actually the same target, but for different versions, maybe, and then use the trained model to try to improve function matching for future later versions of that product.
- OK, now, what do you think about this idea?
  - Yay or nay? **This is the very first idea that I got properly working.**

Using specialized models

- The idea is the following:
  - A binary X version 1.0 has symbols.
  - A binary X version 1.1, however, doesn't have symbols anymore.
  - Generating a dataset (CSV) comparing functions with and without symbols for binary X version 1.0, we can train a model that will improve function matching when diffing version 1.1 against 1.0 to port symbols.
- Let's see an example with, for example, MPEngine.dll.

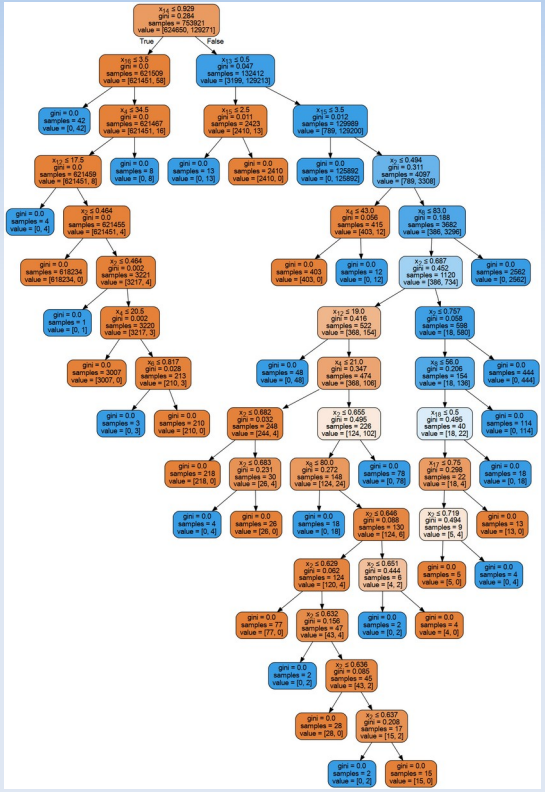
Using specialized models

- I have compared binaries for MPEngine.dll versions:
  - 1.1.19700 and 1.1.20000, no symbols.
  - 1.1.19700 and 1.1.20000, with PDB symbols.
- Then I generated a dataset and trained a model using a decision tree algorithm.

Using specialized models

- The dataset was generated like this:
  - Per each function pair that is good we generate one “good” labelled row (value 1), and 5 “bad” labelled rows (value 0).
- Then, we just train using a decision tree and, after some seconds, we get the following easy to understand tree...

Using specialized models



- The generated dataset has 1,636,376 rows, 308 MB.
- The final tree shown for it looks so easy that it looks like the model is probably correctly generalised.
- And, after some basic testing in Diaphora with a newer version of MPEngine.dll, it seems that it improved a bit the results.

Using specialized models

- So, after seeing the results I got with the specialized model for MPEngine.dll, I thought that, perhaps, I could give a try to using this model against the Cisco Talos Dataset-2 CSV.
- Guess what? The results were very bad. Surprise, right?

```
hora/public/ml$ ./train_dataset.py -v -i mpengine-model.pkl -c /mnt/new/datasets/diaphora_cisco/dataset-2.csv
Loading model mpengine-model.pkl
Loading dataset /mnt/new/datasets/diaphora_cisco/dataset-2.csv
Evaluating the model against dataset /mnt/new/datasets/diaphora_cisco/dataset-2.csv...
Accuracy : 0.21433218789792252
Precision: 0.18362145614994402
Recall   : 0.9875265565736481
F1       : 0.3096637868736923
Done in 16.351264238357544 second(s)
```

Using specialized models

- After this, the next step was *kind* of clear: I cannot train, with my resources, a gigantic model with millions of binaries for every single platform out there.
- The smaller datasets don’t really work with other datasets.
- But, perhaps, I can create a specialized dataset with relevant *enough* files for Diaphora? Like, the following ones:
  - All binaries in the Diaphora testing suite. Private files, sorry.
  - Together with the binaries from the aforementioned Dataset-2.
- Guess what? The results aren’t that bad.

Results before with a Haiku loader for Risc-V

IDA View-A | Best matches | Partial matches | Problematic matches | Unmatched in secondary

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	00021286	sub_21286	0003aa30	error_description	0.9900000	386	291	Same rare constant
00001	00009620	sub_9620	0001a8f0	user_menu(BootVolume &,PathBl…	0.9545879	211	188	Loop count
00002	0000c198	sub_C198	0001e6a0	get_node_from(int)	0.9040000	5	5	Pseudo-code fuzzy AST hash
00003	0001022a	sub_1022A	00023a50	get_parent_partition	0.8978889	5	4	Pseudo-code fuzzy AST hash
00004	0000a836	sub_A836	0001c230	PackageVolumeState::DisplayNa…	0.8830000	3	1	Similar pseudo-code and names
00005	0000bdf8	sub_BDF8	0001e190	write_pos	0.8830000	6	6	Same rare KOKA hash
00006	0000bd20	sub_BD20	0001e060	read_pos	0.8820000	6	6	Same rare KOKA hash
00007	0001764a	sub_1764A	0002c4c0	FATFS::Directory::IsEmpty(void)	0.8770000	1	1	Local affinity
00008	0000be64	sub_BE64	0001e230	write	0.8559412	8	8	Same rare KOKA hash
00009	0000fb54	sub_FB54	00023000	boot::Partition::Lookup(int,Doubly…	0.8484848	12	13	Local affinity
00010	0001ee42	sub_1EE42	00037330	uncompress	0.8373333	1	1	Local affinity
00011	0000ce98	sub_CE98	0001f7f0	put_string(char **,long *,char *)	0.8371709	27	27	Loop count
00012	0000b506	sub_B506	0001d430	fprintf	0.8343333	1	3	Local affinity
00013	00001d8c	sub_1D8C	000117f0	dprintf	0.8333333	1	3	Local affinity
00014	0000b4d6	sub_B4D6	0001d390	printf	0.8333333	1	3	Local affinity
00015	0001c59c	sub_1C59C	00033ca0	BDataIO::ReadExactly(void *,uns…	0.8315294	11	10	Same rare constant
00016	0001c5fe	sub_1C5FE	00033d30	BDataIO::WriteExactly(void const…	0.8315294	11	10	Same rare constant
00017	0000b272	sub_B272	0001d030	RootFileSystem::LookupDontTra…	0.8292683	9	10	Loop count
00018	000089f4	sub_89F4	000195b0	Menu::FindItem(char const*)	0.8270870	6	7	Pseudo-code fuzzy AST hash
00019	00001dfe	sub_1DFE	00011940	platform_debug_get_log_buffer(u…	0.8191429	3	3	Local affinity
00020	0001ed46	sub_1ED46	00037170	uncompress2	0.8114783	27	23	Callee found diffing matches pseudo-code (iteration #1)
00021	00015b22	sub_15B22	0002a120	TarFS::Directory::~Directory()	0.8030000	9	9	Same nodes, edges, loops and strongly connected components
00022	000089a4	sub_89A4	00019500	Menu::ItemAt(int)	0.8020000	12	12	Same nodes, edges, loops and strongly connected components
00023	0000f4ea	sub_F4EA	00022cf0	add_safe_mode_settings(char co…	0.7994912	5	5	Same KOKA hash and MD-Index
00024	000108fe	sub_108FE	00025130	get_partition_type_string(unsigne…	0.7991026	11	11	Same nodes, edges, loops and strongly connected components
00025	0000bc78	sub_BC78	0001df80	open_node(Node *,int)	0.7988718	4	4	Same KOKA hash and MD-Index
00026	00001e0e	sub_1E0E	00011960	mmu_allocate_page	0.7957143	3	3	Same MD Index and constants
00027	0000c852	sub_C852	0001efd0	rewinddir	0.7896105	5	5	Same MD Index and constants
00028	00002f34	sub_2F34	00012ab0	quirks_init(void)	0.7887891	8	8	Same rare constant
00029	0000b44a	sub_B44A	0001d2f0	vfprintf	0.7876496	5	5	Callee found diffing matches pseudo-code (iteration #1)
00030	0000f5fe	sub_F5FE	000181d0	load_modules_from(BootVolume …	0.7828915	13	13	Same nodes, edges, loops and strongly connected components
00031	0001548a	sub_1548A	000297c0	utf8ToUnicode	0.7825625	12	12	Same nodes, edges, loops and strongly connected components
00032	00024c14	sub_24C14	0003fc60	BPackageKit::BHPKG::BPrivate::…	0.7812353	40	40	Switch structures
00033	00017098	sub_17098	0002be30	FATFS::Directory::_GetStreamSiz…	0.7798421	8	8	Same nodes, edges, loops and strongly connected components
00034	0001a2aa	sub_1A2AA	00030860	BPackageKit::BHPKG::BPrivate::…	0.7779338	13	13	Same rare MD Index
00035	00010e9c	sub_10E9C	000257c0	PrimaryPartition::AddLogicalParti…	0.7774444	6	6	Same rare KOKA hash
00036	00022206	sub_22206	0003bcd0	strerror	0.7772750	15	15	Same nodes, edges, loops and strongly connected components
00037	0001b1f2	sub_1B1F2	00031f20	BFdIO::ReadAt(long,void *,unsig…	0.7772424	3	3	Local affinity
00038	0001b1d2	sub_1B1D2	00031dd0	BFdIO::WriteAt(long,void const*,…	0.7772424	3	3	Local affinity
00039	0000b97e	sub_B97E	0001db70	Descriptor::Seek(long,int)	0.7762768	8	8	Same rare KOKA hash
00040	00018dca	sub_18DCA	0002e8a0	PackageFS::PackageSettingsIte…	0.7761701	9	9	Same nodes, edges, loops and strongly connected components
00041	0000c88e	sub_C88E	0001f020	free_parameter(driver_parameter *)	0.7757273	5	5	Same KOKA hash and MD-Index
00042	0000816a	sub_816A	00018990	debug_menu_save_previous_sy…	0.7740247	12	12	Same nodes, edges, loops and strongly connected components
00043	0000bc82	sub_BC82	0001df90	dup	0.7737641	4	4	Same KOKA hash and MD-Index
00044	00009500	sub_9500	0001a6f0	Menu::RemoveItem(MenuItem *)	0.7723793	7	7	Same rare KOKA hash
00045	00018d4e	sub_18D4E	0002e7f0	PackageFS::PackageSettingsIte…	0.7720540	9	9	Same nodes, edges, loops and strongly connected components
00046	0001c17c	sub_1C17C	000336a0	hash_hash_string_part	0.7713019	6	6	Same nodes, edges, loops and strongly connected components
00047	00023bf0	sub_23BF0	0003e330	void DoublyLinkedList<PackageV…	0.7704918	17	17	Local affinity
00048	000176a6	sub_176A6	0002c550	FATFS::Directory::LookupDontTr…	0.7698783	15	15	Same nodes, edges, loops and strongly connected components
00049	000137fa	sub_137FA	00028090	BFS::CachedNode::SetToHeader…	0.7686830	6	6	Same rare KOKA hash
00050	000101ae	sub_101AE	000239b0	create_child_partition	0.7679327	6	6	Same MD Index and constants
00051	00015db6	sub_15DB6	0002a500	TarFS::Directory::LookupDontTra…	0.7660319	4	4	Same KOKA hash and MD-Index
00052	00014b56	sub_14B56	000290d0	BFS::BPlusTree::Find(unsigned c…	0.7658465	24	24	Same nodes, edges, loops and strongly connected components

Line 3 of 433

Results after with a Haiku loader for Risc-V

IDA View-A | Recent scripts | Partial matches | ML matches | Unmatched in primary

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	00021286	sub_21286	0003aa30	error_description	0.9900000	386	291	Same rare constant
00001	00001dfe	sub_1DFE	00011940	platform_debug_get_log_buffer(u…	0.9900000	3	3	Local affinity
00002	00001d8c	sub_1D8C	000117f0	dprintf	0.9900000	1	3	Local affinity
00003	0000b506	sub_B506	0001d390	printf	0.9900000	1	3	Local affinity
00004	0001764a	sub_1764A	0002c4c0	FATFS::Directory::IsEmpty(void)	0.9900000	1	1	Local affinity
00005	0001ee42	sub_1EE42	00037330	uncompress	0.9900000	1	1	Local affinity
00006	0001b1f2	sub_1B1F2	00031f20	BFdIO::ReadAt(long,void *,unsig…	0.9772424	3	3	Local affinity
00007	0001b1d2	sub_1B1D2	00031dd0	BFdIO::WriteAt(long,void const*,…	0.9772424	3	3	Local affinity
00008	00017670	sub_17670	0002c500	FATFS::Directory::GetNextEntry(v…	0.9638392	3	3	Local affinity
00009	0001020e	sub_1020E	00023a30	get_child_partition	0.9620000	1	1	Local affinity
00010	000251f0	sub_251F0	000404f0	BPackageKit::BHPKG::BPrivate::…	0.9617601	6	6	Local affinity
00011	00006e22	sub_6E22	00017810	remove_range_index(addr_range…	0.9605592	3	3	Local affinity
00012	00007f1a	sub_7F1A	000186a0	MenuItem::~MenuItem()	0.9568462	4	4	Related compilation unit
00013	0001b188	sub_1B188	00031e00	BFdIO::SetSize(long)	0.9555862	3	3	Local affinity
00014	00009620	sub_9620	0001a8f0	user_menu(BootVolume &,PathBl…	0.9545879	211	188	Loop count
00015	00007fe4	sub_7FE4	000187c0	user_menu_reboot(Menu *,Menu…	0.9530000	1	1	Local affinity
00016	000124ee	sub_124EE	000272c0	BFS::Stream::Stream(BFS::Volu…	0.9530000	2	2	Local affinity
00017	0001b160	sub_1B160	00031e80	BFdIO::Read(void *,unsigned long)	0.9522424	3	3	Local affinity
00018	0000e84c	sub_E84C	00021c70	list_insert_item_before	0.9498286	3	3	Local affinity
00019	0000b88e	sub_B88E	0001d9e0	Node::Release(void)	0.9490588	4	4	Local affinity
00020	00020fdc	sub_20FDC	0003a6d0	strdup	0.9479073	5	5	Callee found diffing matches pseudo-code (iteration #1)
00021	0000b938	sub_B938	0001db00	Descriptor::Write(void const*,unsi…	0.9446816	3	3	Local affinity
00022	0000b8f8	sub_B8F8	0001daa0	Descriptor::Read(void *,unsigned…	0.9446816	3	3	Local affinity
00023	0001227a	sub_1227A	00027010	BFS::Directory::Close(void *)	0.9440526	3	3	Local affinity
00024	000211a8	sub_211A8	0003a910	strcasecmp	0.9422924	5	5	Callee found diffing matches pseudo-code (iteration #1)
00025	0001c202	sub_1C202	00033730	BPositionIO::Read(void *,unsigne…	0.9418669	3	3	Local affinity
00026	00021274	sub_21274	0003aa10	strcpy	0.9330000	3	3	Callee found diffing matches pseudo-code (iteration #1)
00027	00009580	sub_9580	0001a7b0	Menu::AddShortcut(char,void (*)(…	0.9260769	3	3	Local affinity
00028	0000fbb4	sub_FBB4	00023090	boot::Partition::SetParent(boot::P…	0.9242857	1	1	Local affinity
00029	00022fb0	sub_22FB0	0003d430	BlacklistRootMenu::~BlacklistRo…	0.9240526	4	4	Local affinity
00030	00022f4c	sub_22F4C	0003d3a0	BlacklistMenu::~BlacklistMenu()	0.9240526	4	4	Local affinity
00031	0001a0f4	sub_1A0F4	00030610	BPackageKit::BHPKG::BPrivate::…	0.9239971	7	7	Local affinity
00032	0000162c	sub_162C	00010fc0	console_check_boot_keys(void)	0.9202721	5	5	Local affinity
00033	00019570	sub_19570	0002f420	BPackageKit::BHPKG::BPrivate::…	0.9111166	6	6	Local affinity
00034	00018022	sub_18022	0002d330	FATFS::Stream::FindBlock(long,l…	0.9109208	3	3	Local affinity
00035	0001bbea	sub_1BBEA	00032e20	BPackageKit::BHPKG::BPrivate::…	0.9094327	4	4	Local affinity
00036	00016e30	sub_16E30	0002bac0	FATFS::CachedBlock::SetTo(long…	0.9083333	17	18	Local affinity
00037	000131dc	sub_131DC	00027ca0	BFS::Stream::ReadLink(char *,un…	0.9074684	3	3	Local affinity
00038	0000c958	sub_C958	0001f110	put_chars(char **,long *,char con…	0.9073478	3	3	Local affinity
00039	00010cf8	sub_10CF8	00025630	PrimaryPartition::SetTo(long,long…	0.9055397	3	3	Local affinity
00040	0000873e	sub_873E	00019170	MenuItem::SetLabel(char const*)	0.9040526	3	3	Local affinity
00041	0000c198	sub_C198	0001e6a0	get_node_from(int)	0.9040000	5	5	Pseudo-code fuzzy AST hash
00042	0000e8ca	sub_E8CA	000115b0	video_mode_hook(Menu *,MenuI…	0.9037407	3	3	Local affinity
00043	000082c0	sub_82C0	00018b00	MenuItem::~MenuItem()	0.9030000	2	2	Local affinity
00044	00008456	sub_8456	00018d50	Menu::~Menu()	0.9030000	2	2	Local affinity
00045	0000a81a	sub_A81A	0001c200	PackageVolumeInfo::~PackageV…	0.9030000	2	2	Local affinity
00046	0000b256	sub_B256	0001d000	RootFileSystem::~RootFileSystem()	0.9030000	2	2	Local affinity
00047	0000fa2c	sub_FA2C	00022e60	boot::Partition::~Partition()	0.9030000	2	2	Local affinity
00048	0001225e	sub_1225E	00026fe0	BFS::Directory::~Directory()	0.9030000	2	2	Local affinity
00049	00015c0a	sub_15C0A	0002a2b0	TarFS::Directory::~Directory()	0.9030000	2	2	Local affinity
00050	0001b144	sub_1B144	00031da0	BFdIO::~BFdIO()	0.9030000	2	2	Local affinity
00051	00010d4a	sub_10D4A	000256a0	PrimaryPartition::SetTo(partition_…	0.9025614	3	3	Local affinity
00052	00022f78	sub_22F78	0003d350	BlacklistMenuItem::~BlacklistMe…	0.8997778	4	4	Local affinity

Line 42 of 501

The amalgamation model

- The dataset I built using...
  - The Diaphora testing suite binaries.
  - And all the binaries in Dataset-2.
- ...is what forms the dataset I called the “Diaphora’s amalgamation”.
  - https://github.com/joxeankoret/diaphora-ml/blob/main/datasets/diaphora-amalgamation.csv.bz2
- This dataset isn’t huge, I can distribute it. And it seems to be performing quite well, to be honest.
- Let’s discuss it a bit...

The amalgamation model

- After building the amalgamation model I did some manual tests in Diaphora to see if it looked that it was performing better.
- Over all, it was finding more partial results and removing multimatches, which is an indicative of it working properly.
- After manually testing some new results with a couple of binaries, it turned out that almost no false positive was added, because Diaphora is pretty good trying to remove them (for functions that aren't tiny).
- Next step? Try to, first, re-validate the model by taking the dataset and splitting it into the usual "train, validation & testing" datasets. Let's see its results...

The amalgamation model

- The amalgamation model was randomly split using the following trivial Python script into 3 smaller datasets: 60% for training, 20% for testing and 20% for validation.

```
#!/usr/bin/python3

import pandas as pd
import numpy as np

df = pd.read_csv("amalgamation.csv")
train, validate, test = np.split(df.sample(frac=1, random_state=0x29a), [int(0.6 * len(df)), int(0.8 * len(df))])
with open("amalgamation-train.csv", "w") as f:
    f.write(train.to_csv())

with open("amalgamation-validate.csv", "w") as f:
    f.write(validate.to_csv())

with open("amalgamation-test.csv", "w") as f:
    f.write(test.to_csv())

```

The amalgamation model

```
/diaphora_ml_2024$                          /diaphora/public/ml/train_dataset.py -v -u 17 -o ./diaphora-amalgamation-model.pkl -t -d amalgamation-train.csv
2162040] Loading data from amalgamation-train.csv...
2162040] Fitting data with DecisionTreeClassifier()...
2162040] Evaluating the model against training dataset...
2162040] Accuracy : 0.9938056021621613
2162040] Precision: 0.9876784879231781
2162040] Recall   : 0.9768217262705028
2162040] F1       : 0.9822201072849974
2162040] Saving model...
2162040] Done in 44.14934825897217 second(s)
/diaphora_ml_2024$                          /diaphora/public/ml/train_dataset.py -v -u 17 -i ./diaphora-amalgamation-model.pkl -c amalgamation-test.csv
2162828] Loading model ./diaphora-amalgamation-model.pkl
2162828] Loading dataset amalgamation-test.csv
2162828] Evaluating the model against dataset amalgamation-test.csv...
2162828] Accuracy : 0.9863384241147274
2162828] Precision: 0.9729267710900851
2162828] Recall   : 0.9483330965266924
2162828] F1       : 0.9604725243817086
2162828] Done in 3.6835227012634277 second(s)
/diaphora_ml_2024$                          /diaphora/public/ml/train_dataset.py -v -u 17 -i ./diaphora-amalgamation-model.pkl -c amalgamation-validate.csv
2162849] Loading model ./diaphora-amalgamation-model.pkl
2162849] Loading dataset amalgamation-validate.csv
2162849] Evaluating the model against dataset amalgamation-validate.csv...
2162849] Accuracy : 0.9863033518751081
2162849] Precision: 0.972000475015504
2162849] Recall   : 0.9492728877663448
2162849] F1       : 0.9605022540803254
2162849] Done in 3.673429489135742 second(s)
```

The amalgamation model

- Now, in Diaphora, this model is used when comparing a functions match increasing its similarity ratio by a configurable value when the ML classifier considers the pair good.
- After explaining how this ML engine was plugged into Diaphora, let’s take a look to the results of some example diffing projects:
  - Busybox x86-64 with symbols against Busybox PowerPC.
  - Haiku loader x86-64 with symbols against Haiku loader Risc-V.
  - iOS kernel 10.3.1 ARM64 vs Macos kernel 10.12.4 x86-64.

Busybox x86-64 vs PowerPC, no ML

IDA View-A | Partial matches | Unmatched in secondary | Unmatch

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	10037e44	sub_10037E44	00553d32	rtnl_rtntype_n2a	0.9900000	16	16	Switch structures
00001	1000d518	sub_1000D518	0052623a	identify	0.9900000	354	285	Same rare constant
00002	1000f270	sub_1000F270	00527b5e	process_dev	0.9787537	245	160	Same rare constant
00003	1008d324	sub_1008D324	005a0221	chgrp_main	0.8433333	6	6	Pseudo-code fuzzy AST hash
00004	10002d7c	sub_10002D7C	0051d80d	bb_copyfd_size	0.8363333	4	4	Local affinity
00005	1002f5e8	sub_1002F5E8	0054b47b	sanitize_string	0.8107295	4	4	Same KOKA hash and MD-Index
00006	10043964	sub_10043964	005af92f	hashidx	0.8033221	4	4	Same KOKA hash and MD-Index
00007	100236f4	sub_100236F4	0053c846	print_bytes_scaled	0.7996667	7	5	Same rare constant
00008	100c0ed8	sub_100C0ED8	005cd08c	open_or_warn_stdin	0.7943333	6	6	Pseudo-code fuzzy AST hash
00009	100096cc	sub_100096CC	005229d0	unescape	0.7888974	18	18	Same nodes, edges, loops and strongly connected components
00010	100aa994	sub_100AA994	005b9728	move_to_col	0.7861852	11	11	Same nodes, edges, loops and strongly connected components
00011	100bbd24	sub_100BBD24	005c8613	fast_strtoul_10	0.7820000	4	4	Local affinity
00012	100aaf30	sub_100AAF30	005b9342	find_line	0.7817879	4	4	Same KOKA hash and MD-Index
00013	10021428	sub_10021428	0053a914	echo_stream	0.7796667	4	4	Same KOKA hash and MD-Index
00014	1009f9f8	sub_1009F9F8	005af991	nextarg_0	0.7796275	4	4	Same KOKA hash and MD-Index
00015	10022ec0	sub_10022EC0	0053c8ce	get_hwtype	0.7767059	5	5	Callee found diffing matches pseudo-code (iteration #1)
00016	1009fbc8	sub_1009FBC8	005af9e5	iamarray	0.7748992	6	6	Same nodes, edges, loops and strongly connected components
00017	100a01a4	sub_100A01A4	005afe71	getvar_i_int	0.7747816	9	9	Callee found diffing matches pseudo-code (iteration #1)
00018	100485bc	sub_100485BC	0056561e	isdigit_str9	0.7737143	5	5	Same KOKA hash and MD-Index
00019	100c0f54	sub_100C0F54	005cd07a	xfopen_stdin	0.7733636	3	3	Callee found diffing matches pseudo-code (iteration #1)
00020	1009ecb4	sub_1009ECB4	005aef47	wait_many	0.7688076	4	4	Local affinity
00021	100319ac	sub_100319AC	0054db23	ip_parse_common_args	0.7676628	17	17	Same nodes, edges, loops and strongly connected components
00022	1008d390	sub_1008D390	005a0321	chmod_main	0.7666207	14	12	Same rare constant
00023	100bb01c	sub_100BB01C	005c79ad	bb_mode_string	0.7653828	9	9	Same nodes, edges, loops and strongly connected components
00024	1008c97c	sub_1008C97C	0057b06d	build_row	0.7635385	11	11	Same nodes, edges, loops and strongly connected components
00025	100bfac0	sub_100BFAC0	0051da0a	bb_signals	0.7616491	6	6	Same nodes, edges, loops and strongly connected components
00026	1008f994	sub_1008F994	005a2170	du_main	0.7609668	22	22	Same nodes, edges, loops and strongly connected components
00027	100433b8	sub_100433B8	00561366	edir	0.7608821	24	24	Same nodes, edges, loops and strongly connected components
00028	1001de10	sub_1001DE10	005383ce	network_ioctl	0.7602222	4	4	Callee found diffing matches pseudo-code (iteration #1)
00029	100d4a7c	sub_100D4A7C	00442280	sscanf	0.7594481	3	3	Callee found diffing matches pseudo-code (iteration #1)
00030	100a6600	sub_100A6600	005b5345	bad_nums	0.7593283	5	5	Same KOKA hash and MD-Index
00031	100bffc0	sub_100BFFC0	00558a28	udhcp_end_option	0.7592799	6	6	Same nodes, edges, loops and strongly connected components
00032	10001f64	sub_10001F64	0051d1ca	xsendto	0.7592424	3	3	Same KOKA hash and constants
00033	100912a0	sub_100912A0	005a3526	fsync_main	0.7589805	13	13	Same nodes, edges, loops and strongly connected components
00034	100037e0	sub_100037E0	0051e100	validate_tm_time	0.7588571	3	3	Same KOKA hash and constants
00035	1009e7d8	sub_1009E7D8	005aeb90	kill_all_if_got_signal	0.7581515	9	9	Same nodes, edges, loops and strongly connected components
00036	100759bc	sub_100759BC	0058b96f	resolve_mount_spec	0.7564476	9	9	Same rare KOKA hash
00037	100c1bc8	sub_100C1BC8	005cd833	xconnect	0.7562333	5	5	Same KOKA hash and MD-Index
00038	1006a2d0	sub_1006A2D0	0057fbcf	add_zone	0.7560769	3	3	Callee found diffing matches pseudo-code (iteration #1)
00039	100586c4	sub_100586C4	005711c2	parse_numeric_argv1	0.7560443	6	6	Same rare KOKA hash
00040	10093958	sub_10093958	005a5652	mkfifo_main	0.7557391	7	7	Same nodes, edges, loops and strongly connected components
00041	1000273c	sub_1000273C	0051cc46	open3_or_warn	0.7550476	3	3	Same KOKA hash and constants
00042	10002650	sub_10002650	0051ccc8	rename_or_warn	0.7541818	3	3	Same KOKA hash and constants
00043	10048d10	sub_10048D10	005659f2	__lookupalias	0.7537071	8	8	Same nodes, edges, loops and strongly connected components
00044	10043be4	sub_10043BE4	0056186c	open_trunc_or_warn	0.7533913	3	3	Same KOKA hash and constants
00045	100014f0	sub_100014F0	0051c0d8	skip_whitespace	0.7533534	5	5	Callee found diffing matches pseudo-code (iteration #1)
00046	10026de0	sub_10026DE0	0053f7d8	my_SQRT	0.7530000	1	1	Pseudo-code fuzzy AST hash
00047	10093fc8	sub_10093FC8	005a5de8	get_lcm	0.7526653	9	9	Same nodes, edges, loops and strongly connected components
00048	10048778	sub_10048778	005656b8	hashvar	0.7518413	5	5	Local affinity
00049	100376f8	sub_100376F8	005539d2	rtnl_rtscope_initialize	0.7515714	4	4	Same KOKA hash and MD-Index
00050	1009d3ec	sub_1009D3EC	005ad62f	bb_alphasort	0.7510000	3	1	Pseudo-code fuzzy AST hash
00051	10090160	sub_10090160	005a280f	null	0.7510000	6	7	Pseudo-code fuzzy AST hash
00052	10058750	sub_10058750	0057120e	builtin_return	0.7497778	4	4	Pseudo-code fuzzy AST hash

Line 1 of 1212

Output

```
[Diaphora: Fri Sep 13 10:08:03 2024] Finding unmatched functions
[Diaphora: Fri Sep 13 10:08:03 2024] Final results: Best 0, Partial 1212, Unreliable 0, Multimatches 55
[Diaphora: Fri Sep 13 10:08:03 2024] Matched 37.91% of main binary functions (1212 out of 3197)
[Diaphora: Fri Sep 13 10:08:03 2024] Done, time taken: 0:05:07.773074.
```

Busybox x86-64 vs PowerPC, using ML

IDA View-A | Partial matches | Problematic matches | Unmatched in secondary

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	10037e44	sub_10037E44	00553d32	rtnl_rtntype_n2a	0.9900000	16	16	Switch structures
00001	1000d518	sub_1000D518	0052623a	identify	0.9900000	354	285	Same rare constant
00002	10002d7c	sub_10002D7C	0051d80d	bb_copyfd_size	0.9863333	4	4	Local affinity
00003	1000f270	sub_1000F270	00527b5e	process_dev	0.9787537	245	160	Same rare constant
00004	100bbd24	sub_100BBD24	005c8613	fast_strtoul_10	0.9320000	4	4	Local affinity
00005	10022ec0	sub_10022EC0	0053c8ce	get_hwtype	0.9267059	5	5	Callee found diffing matches pseudo-code (iteration #1)
00006	100a01a4	sub_100A01A4	005afe71	getvar_i_int	0.9247816	9	9	Callee found diffing matches pseudo-code (iteration #1)
00007	100c0f54	sub_100C0F54	005cd07a	xfopen_stdin	0.9233636	3	3	Callee found diffing matches pseudo-code (iteration #1)
00008	1009ecb4	sub_1009ECB4	005aef47	wait_many	0.9188076	4	4	Local affinity
00009	1001de10	sub_1001DE10	005383ce	network_ioctl	0.9102222	4	4	Callee found diffing matches pseudo-code (iteration #1)
00010	100d4a7c	sub_100D4A7C	00442280	sscanf	0.9094481	3	3	Callee found diffing matches pseudo-code (iteration #1)
00011	1006a2d0	sub_1006A2D0	0057fbcf	add_zone	0.9060769	3	3	Callee found diffing matches pseudo-code (iteration #1)
00012	100014f0	sub_100014F0	0051c0d8	skip_whitespace	0.9033534	5	5	Callee found diffing matches pseudo-code (iteration #1)
00013	10048778	sub_10048778	005656b8	hashvar	0.9018413	5	5	Local affinity
00014	100028c8	sub_100028C8	0051cb2a	xmalloc	0.8985591	4	4	Callee found diffing matches pseudo-code (iteration #1)
00015	10002ef8	sub_10002EF8	0051d89d	safe_read	0.8981111	4	4	Callee found diffing matches pseudo-code (iteration #1)
00016	10033758	sub_10033758	0055018b	flush_update_1	0.8965185	4	4	Local affinity
00017	10063c20	sub_10063C20	0057cc20	read_nonempty	0.8962424	3	3	Local affinity
00018	1000287c	sub_1000287C	0051cb49	xrealloc	0.8953333	4	4	Callee found diffing matches pseudo-code (iteration #1)
00019	1002ac00	sub_1002AC00	00542ebc	tcsetattr_serial_or_warn	0.8912051	3	3	Callee found diffing matches pseudo-code (iteration #1)
00020	1001a1bc	sub_1001A1BC	00534cc8	ftp_send	0.8908791	8	8	Callee found diffing matches pseudo-code (iteration #1)
00021	10001d1c	sub_10001D1C	0051d3fa	bb_xioctl	0.8878065	3	3	Callee found diffing matches pseudo-code (iteration #1)
00022	1000ef8c	sub_1000EF8C	005278d9	seek_to_zero	0.8863333	2	2	Related compilation unit
00023	1000218c	sub_1000218C	0051d0df	xchdir	0.8862424	3	3	Callee found diffing matches pseudo-code (iteration #1)
00024	100c0dd8	sub_100C0DD8	005ccece	spawn	0.8847778	6	6	Callee found diffing matches pseudo-code (iteration #1)
00025	100aac10	sub_100AAC10	005b931c	next_line	0.8842222	3	3	Callee found diffing matches pseudo-code (iteration #1)
00026	100b4e20	sub_100B4E20	005c260a	fclose_if_not_stdin	0.8830978	5	5	Callee found diffing matches pseudo-code (iteration #1)
00027	10048678	sub_10048678	0056566d	is_number	0.8822682	4	4	Local affinity
00028	100bb698	sub_100BB698	005c7c83	config_open2	0.8817368	3	3	Callee found diffing matches pseudo-code (iteration #1)
00029	10095ddc	sub_10095DDC	005a742b	get_width_prec	0.8806956	3	3	Local affinity
00030	100a0130	sub_100A0130	005afe32	istrue	0.8801186	5	5	Local affinity
00031	10029ec8	sub_10029EC8	0054230f	set_flags	0.8786966	4	4	Related compilation unit
00032	10073ad8	sub_10073AD8	00588985	pivot_root_main	0.8777143	5	5	Local affinity
00033	10002144	sub_10002144	0051cd0e	xdup2	0.8770000	3	3	Local affinity
00034	10036798	sub_10036798	00553393	addattr32	0.8738954	3	3	Callee found diffing matches pseudo-code (iteration #1)
00035	100331a8	sub_100331A8	0054efb5	get_ctl_fd	0.8738182	5	5	Local affinity
00036	100901b8	sub_100901B8	005a283c	tostring	0.8729617	3	3	Local affinity
00037	100a65b0	sub_100A65B0	005b5320	setCurNum	0.8721216	3	3	Callee found diffing matches pseudo-code (iteration #1)
00038	1009f9c0	sub_1009F9C0	005af972	nextword	0.8696667	3	3	Callee found diffing matches pseudo-code (iteration #1)
00039	1002f60c	sub_1002F60C	0054b4c4	open_socket	0.8695105	3	3	Local affinity
00040	1002be0c	sub_1002BE0C	00543dad	setConMode	0.8670964	8	8	Local affinity
00041	100ba04c	sub_100BA04C	0051be76	llist_pop	0.8656729	3	3	Callee found diffing matches pseudo-code (iteration #1)
00042	100e2964	sub_100E2964	004c7bb0	unlockpt	0.8646853	4	4	Callee found diffing matches pseudo-code (iteration #1)
00043	100d250c	sub_100D250C	004565b0	strchr	0.8644921	3	3	Callee found diffing matches pseudo-code (iteration #1)
00044	10091604	sub_10091604	005a38ce	hostid_main	0.8637460	4	4	Local affinity
00045	1004184c	sub_1004184C	0055fd48	mult_lvl_cmp	0.8636616	4	4	Local affinity
00046	100b29f4	sub_100B29F4	005c09b8	uid2uname	0.8627348	3	3	Callee found diffing matches pseudo-code (iteration #1)
00047	100b2974	sub_100B2974	005c09cd	gid2group	0.8627348	3	3	Callee found diffing matches pseudo-code (iteration #1)
00048	10001fe0	sub_10001FE0	0051d19a	xbind	0.8621681	3	3	Callee found diffing matches pseudo-code (iteration #1)
00049	100454ac	sub_100454AC	00562f65	fail	0.8600000	2	2	Local affinity
00050	10045488	sub_10045488	00562f52	warn	0.8600000	2	2	Local affinity
00051	10058848	sub_10058848	00571290	builtin_continue	0.8540000	2	2	Local affinity
00052	1001a06c	sub_1001A06C	00534a79	xconnect_ftpdata	0.8536667	6	6	Callee found diffing matches pseudo-code (iteration #1)

Line 1 of 1301

Output

```
[Diaphora: Fri Sep 13 10:29:44 2024] Finding unmatched functions
[Diaphora: Fri Sep 13 10:29:44 2024] Final results: Best 0, Partial 1301, Unreliable 0, Multimatches 52
[Diaphora: Fri Sep 13 10:29:44 2024] Matched 40.69% of main binary functions (1301 out of 3197)
[Diaphora: Fri Sep 13 10:29:44 2024] Done, time taken: 0:06:58.188661.
```

Haiku x86-64 vs Risc-V, no ML

IDA View-A | Partial matches | Unmatched in secondary | Problematic matches

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	00021286	sub_21286	0003aa30	error_description	0.9900000	386	291	Same rare constant
00001	00009620	sub_9620	0001a8f0	user_menu(BootVolume &,PathBl…	0.9545879	211	188	Loop count
00002	0000c198	sub_C198	0001e6a0	get_node_from(int)	0.9040000	5	5	Pseudo-code fuzzy AST hash
00003	0001022a	sub_1022A	00023a50	get_parent_partition	0.8978889	5	4	Pseudo-code fuzzy AST hash
00004	0000a836	sub_A836	0001c230	PackageVolumeState::DisplayNa…	0.8830000	3	1	Similar pseudo-code and names
00005	0000bdf8	sub_BDF8	0001e190	write_pos	0.8830000	6	6	Same rare KOKA hash
00006	0000bd20	sub_BD20	0001e060	read_pos	0.8820000	6	6	Same rare KOKA hash
00007	0001764a	sub_1764A	0002c4c0	FATFS::Directory::IsEmpty(void)	0.8770000	1	1	Local affinity
00008	0000be64	sub_BE64	0001e230	write	0.8559412	8	8	Same rare KOKA hash
00009	0000fb54	sub_FB54	00023000	boot::Partition::Lookup(int,Doubly…	0.8484848	12	13	Local affinity
00010	0001ee42	sub_1EE42	00037330	uncompress	0.8373333	1	1	Local affinity
00011	0000ce98	sub_CE98	0001f7f0	put_string(char **,long *,char *)	0.8371709	27	27	Loop count
00012	0000b506	sub_B506	0001d430	fprintf	0.8343333	1	3	Local affinity
00013	00001d8c	sub_1D8C	000117f0	dprintf	0.8333333	1	3	Local affinity
00014	0000b4d6	sub_B4D6	0001d390	printf	0.8333333	1	3	Local affinity
00015	0001c59c	sub_1C59C	00033ca0	BDataIO::ReadExactly(void *,uns…	0.8315294	11	10	Same rare constant
00016	0001c5fe	sub_1C5FE	00033d30	BDataIO::WriteExactly(void const…	0.8315294	11	10	Same rare constant
00017	0000b272	sub_B272	0001d030	RootFileSystem::LookupDontTra…	0.8292683	9	10	Loop count
00018	000089f4	sub_89F4	000195b0	Menu::FindItem(char const*)	0.8270870	6	7	Pseudo-code fuzzy AST hash
00019	00001dfe	sub_1DFE	00011940	platform_debug_get_log_buffer(u…	0.8191429	3	3	Local affinity
00020	0001ed46	sub_1ED46	00037170	uncompress2	0.8114783	27	23	Callee found diffing matches pseudo-code (iteration #1)
00021	00015b22	sub_15B22	0002a120	TarFS::Directory::~Directory()	0.8030000	9	9	Same nodes, edges, loops and strongly connected components
00022	000089a4	sub_89A4	00019500	Menu::ItemAt(int)	0.8020000	12	12	Same nodes, edges, loops and strongly connected components
00023	0000f4ea	sub_F4EA	00022cf0	add_safe_mode_settings(char co…	0.7994912	5	5	Same KOKA hash and MD-Index
00024	000108fe	sub_108FE	00025130	get_partition_type_string(unsigne…	0.7991026	11	11	Same nodes, edges, loops and strongly connected components
00025	0000bc78	sub_BC78	0001df80	open_node(Node *,int)	0.7988718	4	4	Same KOKA hash and MD-Index
00026	00001e0e	sub_1E0E	00011960	mmu_allocate_page	0.7957143	3	3	Same MD Index and constants
00027	0000c852	sub_C852	0001efd0	rewinddir	0.7896105	5	5	Same MD Index and constants
00028	00002f34	sub_2F34	00012ab0	quirks_init(void)	0.7887891	8	8	Same rare constant
00029	0000b44a	sub_B44A	0001d2f0	vfprintf	0.7876496	5	5	Callee found diffing matches pseudo-code (iteration #1)
00030	0000f5fe	sub_F5FE	000181d0	load_modules_from(BootVolume …	0.7828915	13	13	Same nodes, edges, loops and strongly connected components
00031	0001548a	sub_1548A	000297c0	utf8ToUnicode	0.7825625	12	12	Same nodes, edges, loops and strongly connected components
00032	00024c14	sub_24C14	0003fc60	BPackageKit::BHPKG::BPrivate::…	0.7812353	40	40	Switch structures
00033	00017098	sub_17098	0002be30	FATFS::Directory::_GetStreamSiz…	0.7798421	8	8	Same nodes, edges, loops and strongly connected components
00034	0001a2aa	sub_1A2AA	00030860	BPackageKit::BHPKG::BPrivate::…	0.7779338	13	13	Same rare MD Index
00035	00010e9c	sub_10E9C	000257c0	PrimaryPartition::AddLogicalParti…	0.7774444	6	6	Same rare KOKA hash
00036	00022206	sub_22206	0003bcd0	strerror	0.7772750	15	15	Same nodes, edges, loops and strongly connected components
00037	0001b1f2	sub_1B1F2	00031f20	BFdIO::ReadAt(long,void *,unsig…	0.7772424	3	3	Local affinity
00038	0001b1d2	sub_1B1D2	00031dd0	BFdIO::WriteAt(long,void const*,…	0.7772424	3	3	Local affinity
00039	0000b97e	sub_B97E	0001db70	Descriptor::Seek(long,int)	0.7762768	8	8	Same rare KOKA hash
00040	00018dca	sub_18DCA	0002e8a0	PackageFS::PackageSettingsIte…	0.7761701	9	9	Same nodes, edges, loops and strongly connected components
00041	0000c88e	sub_C88E	0001f020	free_parameter(driver_parameter *)	0.7757273	5	5	Same KOKA hash and MD-Index
00042	0000816a	sub_816A	00018990	debug_menu_save_previous_sy…	0.7740247	12	12	Same nodes, edges, loops and strongly connected components
00043	0000bc82	sub_BC82	0001df90	dup	0.7737641	4	4	Same KOKA hash and MD-Index
00044	00009500	sub_9500	0001a6f0	Menu::RemoveItem(MenuItem *)	0.7723793	7	7	Same rare KOKA hash
00045	00018d4e	sub_18D4E	0002e7f0	PackageFS::PackageSettingsIte…	0.7720540	9	9	Same nodes, edges, loops and strongly connected components
00046	0001c17c	sub_1C17C	000336a0	hash_hash_string_part	0.7713019	6	6	Same nodes, edges, loops and strongly connected components
00047	00023bf0	sub_23BF0	0003e330	void DoublyLinkedList<PackageV…	0.7704918	17	17	Local affinity
00048	000176a6	sub_176A6	0002c550	FATFS::Directory::LookupDontTr…	0.7698783	15	15	Same nodes, edges, loops and strongly connected components
00049	000137fa	sub_137FA	00028090	BFS::CachedNode::SetToHeader…	0.7686830	6	6	Same rare KOKA hash
00050	000101ae	sub_101AE	000239b0	create_child_partition	0.7679327	6	6	Same MD Index and constants
00051	00015db6	sub_15DB6	0002a500	TarFS::Directory::LookupDontTra…	0.7660319	4	4	Same KOKA hash and MD-Index
00052	00014b56	sub_14B56	000290d0	BFS::BPlusTree::Find(unsigned c…	0.7658465	24	24	Same nodes, edges, loops and strongly connected components

Line 1 of 433

Output

```
[Diaphora: Fri Sep 13 10:58:52 2024] Finding with heuristic 'Related compilation unit'
[Diaphora: Fri Sep 13 10:58:52 2024] Finding locally affine functions
[Diaphora: Fri Sep 13 10:58:52 2024] Finding unmatched functions
[Diaphora: Fri Sep 13 10:58:52 2024] Final results: Best 3, Partial 433, Unreliable 0, Multimatches 29
[Diaphora: Fri Sep 13 10:58:52 2024] Matched 59.32% of main binary functions (436 out of 735)
[Diaphora: Fri Sep 13 10:58:52 2024] Done, time taken: 0:00:19.754841.
```

Haiku x86-64 vs Risc-V, using ML

IDA View-A | Partial matches | Best matches | Problematic matches | Unmat

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	0000fb54	sub_FB54	00023000	boot::Partition::Lookup(int,Doubly…	0.9984848	12	13	Local affinity
00001	00021286	sub_21286	0003aa30	error_description	0.9900000	386	291	Same rare constant
00002	0001ee42	sub_1EE42	00037330	uncompress	0.9873333	1	1	Local affinity
00003	00001dfe	sub_1DFE	00011940	platform_debug_get_log_buffer(u…	0.9691429	3	3	Local affinity
00004	00009620	sub_9620	0001a8f0	user_menu(BootVolume &,PathBl…	0.9545879	211	188	Loop count
00005	0001b1f2	sub_1B1F2	00031f20	BFdIO::ReadAt(long,void *,unsig…	0.9272424	3	3	Local affinity
00006	0001b1d2	sub_1B1D2	00031dd0	BFdIO::WriteAt(long,void const*,…	0.9272424	3	3	Local affinity
00007	00017670	sub_17670	0002c500	FATFS::Directory::GetNextEntry(v…	0.9138392	3	3	Local affinity
00008	0001020e	sub_1020E	00023a30	get_child_partition	0.9120000	1	1	Local affinity
00009	000251f0	sub_251F0	000404f0	BPackageKit::BHPKG::BPrivate::…	0.9117601	6	6	Local affinity
00010	00006e22	sub_6E22	00017810	remove_range_index(addr_range…	0.9105592	3	3	Local affinity
00011	00007f1a	sub_7F1A	000186a0	MenuItem::~MenuItem()	0.9068462	4	4	Related compilation unit
00012	0001b188	sub_1B188	00031e00	BFdIO::SetSize(long)	0.9055862	3	3	Local affinity
00013	0000c198	sub_C198	0001e6a0	get_node_from(int)	0.9040000	5	5	Pseudo-code fuzzy AST hash
00014	00007fe4	sub_7FE4	000187c0	user_menu_reboot(Menu *,Menu…	0.9030000	1	1	Local affinity
00015	000124ee	sub_124EE	000272c0	BFS::Stream::Stream(BFS::Volu…	0.9030000	2	2	Local affinity
00016	0001b160	sub_1B160	00031e80	BFdIO::Read(void *,unsigned long)	0.9022424	3	3	Local affinity
00017	0000e84c	sub_E84C	00021c70	list_insert_item_before	0.8998286	3	3	Local affinity
00018	0000b88e	sub_B88E	0001d9e0	Node::Release(void)	0.8990588	4	4	Local affinity
00019	00020fdc	sub_20FDC	0003a6d0	strdup	0.8979073	5	5	Callee found diffing matches pseudo-code (iteration #1)
00020	0001022a	sub_1022A	00023a50	get_parent_partition	0.8978889	5	4	Pseudo-code fuzzy AST hash
00021	0000b938	sub_B938	0001db00	Descriptor::Write(void const*,unsi…	0.8946816	3	3	Local affinity
00022	0000b8f8	sub_B8F8	0001daa0	Descriptor::Read(void *,unsigned…	0.8946816	3	3	Local affinity
00023	0001227a	sub_1227A	00027010	BFS::Directory::Close(void *)	0.8940526	3	3	Local affinity
00024	00020e4e	sub_20E4E	0003a4c0	memcmp	0.8936129	5	5	Callee found diffing matches pseudo-code (iteration #1)
00025	000211a8	sub_211A8	0003a910	strcasecmp	0.8922924	5	5	Callee found diffing matches pseudo-code (iteration #1)
00026	0000bd20	sub_BD20	0001e060	read_pos	0.8920000	6	6	Callee found diffing matches pseudo-code (iteration #1)
00027	0001c202	sub_1C202	00033730	BPositionIO::Read(void *,unsigne…	0.8918669	3	3	Local affinity
00028	0000a836	sub_A836	0001c230	PackageVolumeState::DisplayNa…	0.8830000	3	1	Similar pseudo-code and names
00029	0000bdf8	sub_BDF8	0001e190	write_pos	0.8830000	6	6	Same rare KOKA hash
00030	00021274	sub_21274	0003aa10	strcpy	0.8830000	3	3	Callee found diffing matches pseudo-code (iteration #1)
00031	0001764a	sub_1764A	0002c4c0	FATFS::Directory::IsEmpty(void)	0.8770000	1	1	Local affinity
00032	00009580	sub_9580	0001a7b0	Menu::AddShortcut(char,void (*)(…	0.8760769	3	3	Local affinity
00033	0000fbb4	sub_FBB4	00023090	boot::Partition::SetParent(boot::P…	0.8742857	1	1	Local affinity
00034	00022fb0	sub_22FB0	0003d430	BlacklistRootMenu::~BlacklistRo…	0.8740526	4	4	Local affinity
00035	00022f4c	sub_22F4C	0003d3a0	BlacklistMenu::~BlacklistMenu()	0.8740526	4	4	Local affinity
00036	0001a0f4	sub_1A0F4	00030610	BPackageKit::BHPKG::BPrivate::…	0.8739971	7	7	Local affinity
00037	0000162c	sub_162C	00010fc0	console_check_boot_keys(void)	0.8702721	5	5	Local affinity
00038	00019570	sub_19570	0002f420	BPackageKit::BHPKG::BPrivate::…	0.8611166	6	6	Local affinity
00039	00018022	sub_18022	0002d330	FATFS::Stream::FindBlock(long,l…	0.8609208	3	3	Local affinity
00040	0001bbea	sub_1BBEA	00032e20	BPackageKit::BHPKG::BPrivate::…	0.8594327	4	4	Local affinity
00041	00016e30	sub_16E30	0002bac0	FATFS::CachedBlock::SetTo(long…	0.8583333	17	18	Local affinity
00042	000131dc	sub_131DC	00027ca0	BFS::Stream::ReadLink(char *,un…	0.8574684	3	3	Local affinity
00043	0000c958	sub_C958	0001f110	put_chars(char **,long *,char con…	0.8573478	3	3	Local affinity
00044	0000be64	sub_BE64	0001e230	write	0.8559412	8	8	Same rare KOKA hash
00045	00010cf8	sub_10CF8	00025630	PrimaryPartition::SetTo(long,long…	0.8555397	3	3	Local affinity
00046	0000873e	sub_873E	00019170	MenuItem::SetLabel(char const*)	0.8540526	3	3	Local affinity
00047	0000e8ca	sub_E8CA	000115b0	video_mode_hook(Menu *,MenuI…	0.8537407	3	3	Local affinity
00048	000082c0	sub_82C0	00018b00	MenuItem::~MenuItem()	0.8530000	2	2	Local affinity
00049	00008456	sub_8456	00018d50	Menu::~Menu()	0.8530000	2	2	Local affinity
00050	0000a81a	sub_A81A	0001c200	PackageVolumeInfo::~PackageV…	0.8530000	2	2	Local affinity
00051	0000b256	sub_B256	0001d000	RootFileSystem::~RootFileSystem()	0.8530000	2	2	Local affinity
00052	0000fa2c	sub_FA2C	00022e60	boot::Partition::~Partition()	0.8530000	2	2	Local affinity

Line 1 of 488

Output

```
[Diaphora: Fri Sep 13 11:02:02 2024] Finding with heuristic 'Related compilation unit'
[Diaphora: Fri Sep 13 11:02:02 2024] Finding locally affine functions
[Diaphora: Fri Sep 13 11:02:02 2024] Finding unmatched functions
[Diaphora: Fri Sep 13 11:02:02 2024] Final results: Best 3, Partial 488, Unreliable 0, Multimatches 29
[Diaphora: Fri Sep 13 11:02:02 2024] Matched 66.80% of main binary functions (491 out of 735)
[Diaphora: Fri Sep 13 11:02:02 2024] Done, time taken: 0:00:36.627234.
```

iOS kernel 10.3.1 vs Macos kernel 10.12.4, no ML

IDA View-A | Partial matches | Problematic matches | Unmatched in secondary

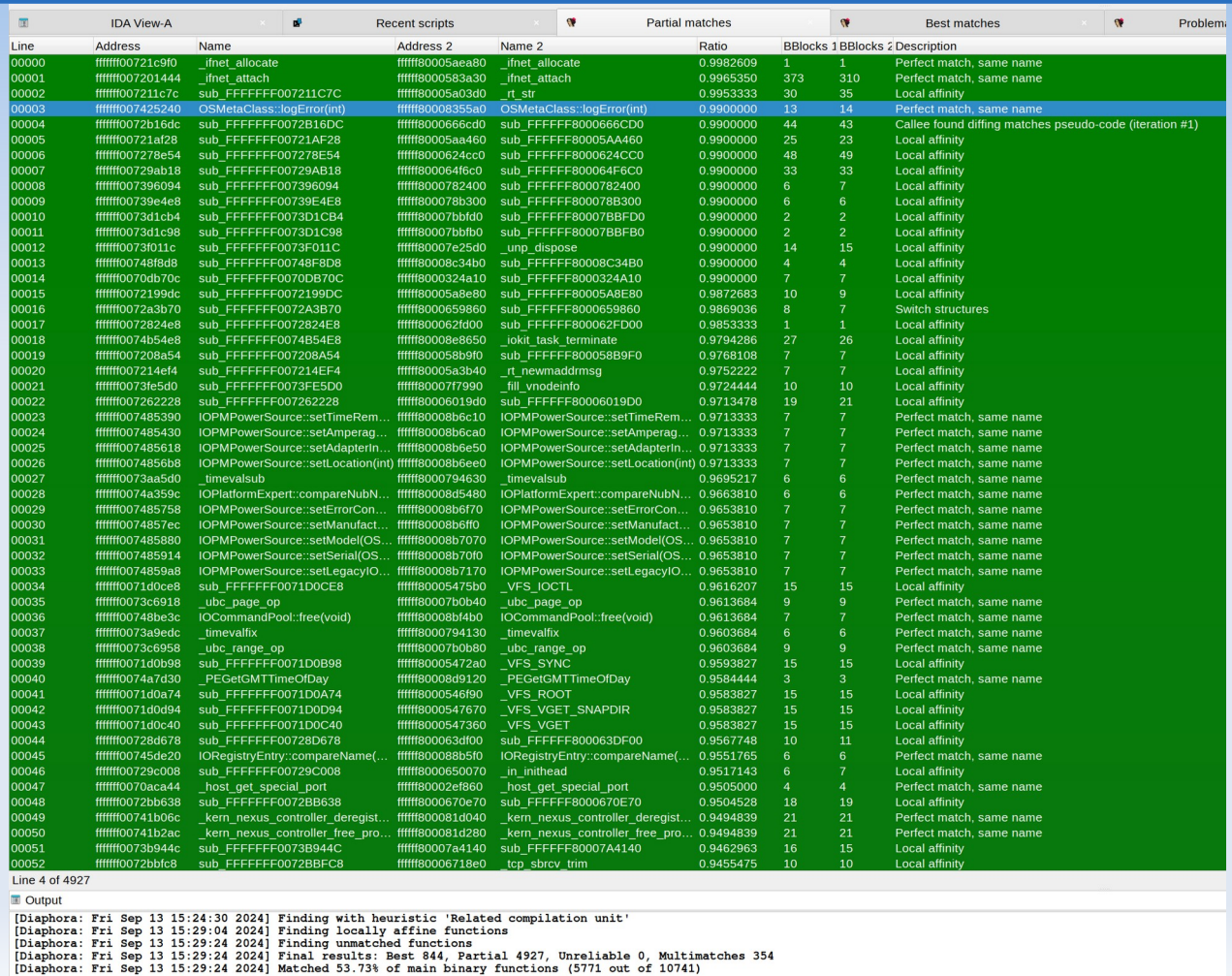
Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	ffffff00721c9f0	_ifnet_allocate	ffffff80005aea80	_ifnet_allocate	0.9982609	1	1	Perfect match, same name
00001	ffffff007201444	_ifnet_attach	ffffff8000583a30	_ifnet_attach	0.9965350	373	310	Perfect match, same name
00002	ffffff00721af28	sub_FFFFFF00721AF28	ffffff80005aa460	sub_FFFFFF80005AA460	0.9901312	25	23	Local affinity
00003	ffffff007425240	OSMetaClass::logError(int)	ffffff80008355a0	OSMetaClass::logError(int)	0.9900000	13	14	Perfect match, same name
00004	ffffff0072a3b70	sub_FFFFFF0072A3B70	ffffff8000659860	sub_FFFFFF8000659860	0.9869036	8	7	Switch structures
00005	ffffff007485390	IOPMPowerSource::setTimeRem…	ffffff80008b6c10	IOPMPowerSource::setTimeRem…	0.9713333	7	7	Perfect match, same name
00006	ffffff007485430	IOPMPowerSource::setAmperag…	ffffff80008b6ca0	IOPMPowerSource::setAmperag…	0.9713333	7	7	Perfect match, same name
00007	ffffff007485618	IOPMPowerSource::setAdapterIn…	ffffff80008b6e50	IOPMPowerSource::setAdapterIn…	0.9713333	7	7	Perfect match, same name
00008	ffffff074856b8	IOPMPowerSource::setLocation(int)	ffffff80008b6ee0	IOPMPowerSource::setLocation(int)	0.9713333	7	7	Perfect match, same name
00009	ffffff0073aa5d0	_timevalsub	ffffff8000794630	_timevalsub	0.9695217	6	6	Perfect match, same name
00010	ffffff0074a359c	IOPlatformExpert::compareNubN…	ffffff80008d5480	IOPlatformExpert::compareNubN…	0.9663810	6	6	Perfect match, same name
00011	ffffff007485758	IOPMPowerSource::setErrorCon…	ffffff80008b6f70	IOPMPowerSource::setErrorCon…	0.9653810	7	7	Perfect match, same name
00012	ffffff0074857ec	IOPMPowerSource::setManufact…	ffffff80008b6ff0	IOPMPowerSource::setManufact…	0.9653810	7	7	Perfect match, same name
00013	ffffff007485880	IOPMPowerSource::setModel(OS…	ffffff80008b7070	IOPMPowerSource::setModel(OS…	0.9653810	7	7	Perfect match, same name
00014	ffffff007485914	IOPMPowerSource::setSerial(OS…	ffffff80008b70f0	IOPMPowerSource::setSerial(OS…	0.9653810	7	7	Perfect match, same name
00015	ffffff0074859a8	IOPMPowerSource::setLegacyIO…	ffffff80008b7170	IOPMPowerSource::setLegacyIO…	0.9653810	7	7	Perfect match, same name
00016	ffffff0073c6918	_ubc_page_op	ffffff80007b0b40	_ubc_page_op	0.9613684	9	9	Perfect match, same name
00017	ffffff00748be3c	IOCommandPool::free(void)	ffffff80008bf4b0	IOCommandPool::free(void)	0.9613684	7	7	Perfect match, same name
00018	ffffff0073a9edc	_timevalfix	ffffff8000794130	_timevalfix	0.9603684	6	6	Perfect match, same name
00019	ffffff0073c6958	_ubc_range_op	ffffff80007b0b80	_ubc_range_op	0.9603684	9	9	Perfect match, same name
00020	ffffff0074a7d30	_PEGetGMTTimeOfDay	ffffff80008d9120	_PEGetGMTTimeOfDay	0.9584444	3	3	Perfect match, same name
00021	ffffff00745de20	IORegistryEntry::compareName(…	ffffff800088b5f0	IORegistryEntry::compareName(…	0.9551765	6	6	Perfect match, same name
00022	ffffff0070aca44	_host_get_special_port	ffffff80002ef860	_host_get_special_port	0.9505000	4	4	Perfect match, same name
00023	ffffff00741b06c	_kern_nexus_controller_deregist…	ffffff800081d040	_kern_nexus_controller_deregist…	0.9494839	21	21	Perfect match, same name
00024	ffffff00741b2ac	_kern_nexus_controller_free_pro…	ffffff800081d280	_kern_nexus_controller_free_pro…	0.9494839	21	21	Perfect match, same name
00025	ffffff0074b4300	IOMachPort::dictForType(unsign…	ffffff80008e6980	IOMachPort::dictForType(unsign…	0.9433333	8	7	Perfect match, same name
00026	ffffff007220430	_ifnet_report_issues	ffffff80005b2c10	_ifnet_report_issues	0.9425714	4	4	Perfect match, same name
00027	ffffff007220ac8	_ifnet_set_packetpreamblelen	ffffff80005b32b0	_ifnet_set_packetpreamblelen	0.9415714	4	4	Perfect match, same name
00028	ffffff007220b28	_ifnet_get_fastlane_capable	ffffff80005b3320	_ifnet_get_fastlane_capable	0.9415714	4	4	Perfect match, same name
00029	ffffff007484d94	IOPMPowerSource::setPSProper…	ffffff80008b66b0	IOPMPowerSource::setPSProper…	0.9415714	7	7	Perfect match, same name
00030	ffffff007489b40	IOInterruptEventSource::enable(v…	ffffff80008bd210	IOInterruptEventSource::enable(v…	0.9415714	4	4	Perfect match, same name
00031	ffffff007489b84	IOInterruptEventSource::disable(…	ffffff80008bd250	IOInterruptEventSource::disable(…	0.9415714	4	4	Perfect match, same name
00032	ffffff0074ad448	IODMAController::notifyDMACom…	ffffff80008df6a0	IODMAController::notifyDMACom…	0.9395714	1	1	Perfect match, same name
00033	ffffff0074656bc	_IOOpenServiceIterator::free(void)	ffffff8000892bc0	_IOOpenServiceIterator::free(void)	0.9370769	5	5	Perfect match, same name
00034	ffffff0074653b4	sub_FFFFFF0074653B4	ffffff8000892780	sub_FFFFFF8000892780	0.9370769	3	3	Callee found diffing matches pseudo-code (iteration #1)
00035	ffffff007375320	sub_FFFFFF007375320	ffffff800075b610	_kevent	0.9363333	1	1	Pseudo-code fuzzy AST hash
00036	ffffff00742865c	OSBoolean::serialize(OSSerializ…	ffffff8000838a50	OSBoolean::serialize(OSSerializ…	0.9360909	3	6	Perfect match, same name
00037	ffffff007116aec	_memory_object_page_op	ffffff80003650d0	_memory_object_page_op	0.9360769	6	6	Perfect match, same name
00038	ffffff0074b34b0	IOUserClient::getTargetAndMeth…	ffffff80008e42f0	IOUserClient::getTargetAndMeth…	0.9360769	3	3	Perfect match, same name
00039	ffffff0074b34e8	IOUserClient::getAsyncTargetAn…	ffffff80008e4320	IOUserClient::getAsyncTargetAn…	0.9360769	3	3	Perfect match, same name
00040	ffffff0074b9f64	sub_FFFFFF0074B9F64	ffffff80008ee450	IOKitDiagnostics::serialize(OSSe…	0.9360000	12	12	Same MD Index and constants
00041	ffffff00746f930	IOService::unregisterAllInterest(v…	ffffff800089d690	IOService::unregisterAllInterest(v…	0.9340000	2	2	Perfect match, same name
00042	ffffff007486a34	IOPMPowerSourceList::addToLis…	ffffff80008b8050	IOPMPowerSourceList::addToLis…	0.9330000	6	6	Perfect match, same name
00043	ffffff007393f50	sub_FFFFFF007393F50	ffffff800077f230	sub_FFFFFF800077F230	0.9325714	5	5	Same KOKA hash and MD-Index
00044	ffffff00490da0	IODMACommand::prepareWithS…	ffffff80008c4760	IODMACommand::prepareWithS…	0.9320526	1	1	Perfect match, same name
00045	ffffff007483cfc	IOCatalogue::terminateDriversFo…	ffffff80008b5720	IOCatalogue::terminateDriversFo…	0.9296667	4	4	Perfect match, same name
00046	ffffff0071ff2c8	_ifnet_event	ffffff80005817f0	_ifnet_event	0.9286667	4	4	Perfect match, same name
00047	ffffff007456e74	_IORecursiveLockSleep	ffffff8000884b10	_IORecursiveLockSleep	0.9286667	1	1	Perfect match, same name
00048	ffffff007456ebc	_IORecursiveLockSleepDeadline	ffffff8000884b60	_IORecursiveLockSleepDeadline	0.9286667	1	1	Perfect match, same name
00049	ffffff00748bfa8	IOCommandPool::gatedGetCom…	ffffff80008bf620	IOCommandPool::gatedGetCom…	0.9252857	10	9	Perfect match, same name
00050	ffffff0073facf8	sub_FFFFFF0073FACF8	ffffff80007f4de0	_psynch_cvsignal	0.9250769	1	1	Pseudo-code fuzzy AST hash
00051	ffffff007302df4	sub_FFFFFF007302DF4	ffffff80006c2fb0	_icmp6_dgram_ctloutput	0.9241988	17	14	Switch structures
00052	ffffff0073fa5b0	_sock_inject_data_out	ffffff80007f4580	_sock_inject_data_out	0.9230909	4	4	Perfect match, same name

Line 1 of 4134

Output

```
[Diaphora: Fri Sep 13 14:02:39 2024] Finding with heuristic 'Related compilation unit'
[Diaphora: Fri Sep 13 14:06:25 2024] Finding locally affine functions
[Diaphora: Fri Sep 13 14:06:45 2024] Finding unmatched functions
[Diaphora: Fri Sep 13 14:06:45 2024] Final results: Best 844, Partial 4134, Unreliable 0, Multimatches 349
[Diaphora: Fri Sep 13 14:06:45 2024] Matched 46.35% of main binary functions (4978 out of 10741)
[Diaphora: Fri Sep 13 14:06:45 2024] Done, time taken: 0:16:30.413260.
```

iOS kernel 10.3.1 vs Macos kernel 10.12.4, using ML



IDA View-A | Recent scripts | Partial matches | Best matches | Problem

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	ffffff00721c9f0	_ifnet_allocate	ffffff80005aea80	_ifnet_allocate	0.9982609	1	1	Perfect match, same name
00001	ffffff007201444	_ifnet_attach	ffffff8000583a30	_ifnet_attach	0.9965350	373	310	Perfect match, same name
00002	ffffff007211c7c	sub_FFFFFFF007211C7C	ffffff80005a03d0	_rt_str	0.9953333	30	35	Local affinity
00003	ffffff007425240	OSMetaClass::logError(int)	ffffff80008355a0	OSMetaClass::logError(int)	0.9900000	13	14	Perfect match, same name
00004	ffffff0072b16dc	sub_FFFFFFF0072B16DC	ffffff8000666cd0	sub_FFFFFF8000666CD0	0.9900000	44	43	Callee found diffing matches pseudo-code (iteration #1)
00005	ffffff00721af28	sub_FFFFFFF00721AF28	ffffff80005aa460	sub_FFFFFF80005AA460	0.9900000	25	23	Local affinity
00006	ffffff007278e54	sub_FFFFFFF007278E54	ffffff8000624cc0	sub_FFFFFF8000624CC0	0.9900000	48	49	Local affinity
00007	ffffff00729ab18	sub_FFFFFFF00729AB18	ffffff800064f6c0	sub_FFFFFF800064F6C0	0.9900000	33	33	Local affinity
00008	ffffff007396094	sub_FFFFFFF007396094	ffffff8000782400	sub_FFFFFF8000782400	0.9900000	6	7	Local affinity
00009	ffffff00739e4e8	sub_FFFFFFF00739E4E8	ffffff800078b300	sub_FFFFFF800078B300	0.9900000	6	6	Local affinity
00010	ffffff0073d1cb4	sub_FFFFFFF0073D1CB4	ffffff80007bbfd0	sub_FFFFFF80007BBFD0	0.9900000	2	2	Local affinity
00011	ffffff0073d1c98	sub_FFFFFFF0073D1C98	ffffff80007bbfb0	sub_FFFFFF80007BBFB0	0.9900000	2	2	Local affinity
00012	ffffff0073f011c	sub_FFFFFFF0073F011C	ffffff80007e25d0	_unp_dispose	0.9900000	14	15	Local affinity
00013	ffffff00748f8d8	sub_FFFFFFF00748F8D8	ffffff80008c34b0	sub_FFFFFF80008C34B0	0.9900000	4	4	Local affinity
00014	ffffff0070db70c	sub_FFFFFFF0070DB70C	ffffff8000324a10	sub_FFFFFF8000324A10	0.9900000	7	7	Local affinity
00015	ffffff0072199dc	sub_FFFFFFF0072199DC	ffffff80005a8e80	sub_FFFFFF80005A8E80	0.9872683	10	9	Local affinity
00016	ffffff0072a3b70	sub_FFFFFFF0072A3B70	ffffff8000659860	sub_FFFFFF8000659860	0.9869036	8	7	Switch structures
00017	ffffff0072824e8	sub_FFFFFFF0072824E8	ffffff800062fd00	sub_FFFFFF800062FD00	0.9853333	1	1	Local affinity
00018	ffffff0074b54e8	sub_FFFFFFF0074B54E8	ffffff80008e8650	_iokit_task_terminate	0.9794286	27	26	Local affinity
00019	ffffff007208a54	sub_FFFFFFF007208A54	ffffff800058b9f0	sub_FFFFFF800058B9F0	0.9768108	7	7	Local affinity
00020	ffffff007214ef4	sub_FFFFFFF007214EF4	ffffff80005a3b40	_rt_newmaddrmsg	0.9752222	7	7	Local affinity
00021	ffffff0073fe5d0	sub_FFFFFFF0073FE5D0	ffffff80007f7990	_fill_vnodeinfo	0.9724444	10	10	Local affinity
00022	ffffff007262228	sub_FFFFFFF007262228	ffffff80006019d0	sub_FFFFFF80006019D0	0.9713478	19	21	Local affinity
00023	ffffff007485390	IOPMPowerSource::setTimeRem…	ffffff80008b6c10	IOPMPowerSource::setTimeRem…	0.9713333	7	7	Perfect match, same name
00024	ffffff007485430	IOPMPowerSource::setAmperag…	ffffff80008b6ca0	IOPMPowerSource::setAmperag…	0.9713333	7	7	Perfect match, same name
00025	ffffff007485618	IOPMPowerSource::setAdapterIn…	ffffff80008b6e50	IOPMPowerSource::setAdapterIn…	0.9713333	7	7	Perfect match, same name
00026	ffffff0074856b8	IOPMPowerSource::setLocation(int)	ffffff80008b6ee0	IOPMPowerSource::setLocation(int)	0.9713333	7	7	Perfect match, same name
00027	ffffff0073aa5d0	_timevalsub	ffffff8000794630	_timevalsub	0.9695217	6	6	Perfect match, same name
00028	ffffff0074a359c	IOPlatformExpert::compareNubN…	ffffff80008d5480	IOPlatformExpert::compareNubN…	0.9663810	6	6	Perfect match, same name
00029	ffffff007485758	IOPMPowerSource::setErrorCon…	ffffff80008b6f70	IOPMPowerSource::setErrorCon…	0.9653810	7	7	Perfect match, same name
00030	ffffff0074857ec	IOPMPowerSource::setManufact…	ffffff80008b6ff0	IOPMPowerSource::setManufact…	0.9653810	7	7	Perfect match, same name
00031	ffffff007485880	IOPMPowerSource::setModel(OS…	ffffff80008b7070	IOPMPowerSource::setModel(OS…	0.9653810	7	7	Perfect match, same name
00032	ffffff007485914	IOPMPowerSource::setSerial(OS…	ffffff80008b70f0	IOPMPowerSource::setSerial(OS…	0.9653810	7	7	Perfect match, same name
00033	ffffff0074859a8	IOPMPowerSource::setLegacyIO…	ffffff80008b7170	IOPMPowerSource::setLegacyIO…	0.9653810	7	7	Perfect match, same name
00034	ffffff0071d0ce8	sub_FFFFFFF0071D0CE8	ffffff80005475b0	_VFS_IOCTL	0.9616207	15	15	Local affinity
00035	ffffff0073c6918	_ubc_page_op	ffffff80007b0b40	_ubc_page_op	0.9613684	9	9	Perfect match, same name
00036	ffffff0074 8be3c	IOCommandPool::free(void)	ffffff80008bf4b0	IOCommandPool::free(void)	0.9613684	7	7	Perfect match, same name
00037	ffffff0073a9edc	_timevalfix	ffffff8000794130	_timevalfix	0.9603684	6	6	Perfect match, same name
00038	ffffff0073c6958	_ubc_range_op	ffffff80007b0b80	_ubc_range_op	0.9603684	9	9	Perfect match, same name
00039	ffffff0071d0b98	sub_FFFFFFF0071D0B98	ffffff80005472a0	_VFS_SYNC	0.9593827	15	15	Local affinity
00040	ffffff0074a7d30	_PEGetGMTTimeOfDay	ffffff80008d9120	_PEGetGMTTimeOfDay	0.9584444	3	3	Perfect match, same name
00041	ffffff0071d0a74	sub_FFFFFFF0071D0A74	ffffff8000546f90	_VFS_ROOT	0.9583827	15	15	Local affinity
00042	ffffff0071d0d94	sub_FFFFFFF0071D0D94	ffffff8000547670	_VFS_VGET_SNAPDIR	0.9583827	15	15	Local affinity
00043	ffffff0071d0c40	sub_FFFFFFF0071D0C40	ffffff8000547360	_VFS_VGET	0.9583827	15	15	Local affinity
00044	ffffff00728d678	sub_FFFFFFF00728D678	ffffff800063df00	sub_FFFFFF800063DF00	0.9567748	10	11	Local affinity
00045	ffffff00745de20	IORegistryEntry::compareName(…	ffffff800088b5f0	IORegistryEntry::compareName(…	0.9551765	6	6	Perfect match, same name
00046	ffffff00729c008	sub_FFFFFFF00729C008	ffffff8000650070	_in_inithead	0.9517143	6	7	Local affinity
00047	ffffff0070aca44	_host_get_special_port	ffffff80002ef860	_host_get_special_port	0.9505000	4	4	Perfect match, same name
00048	ffffff0072bb638	sub_FFFFFFF0072BB638	ffffff8000670e70	sub_FFFFFF8000670E70	0.9504528	18	19	Local affinity
00049	ffffff00741b06c	_kern_nexus_controller_deregist…	ffffff800081d040	_kern_nexus_controller_deregist…	0.9494839	21	21	Perfect match, same name
00050	ffffff00741b2ac	_kern_nexus_controller_free_pro…	ffffff800081d280	_kern_nexus_controller_free_pro…	0.9494839	21	21	Perfect match, same name
00051	ffffff0073b944c	sub_FFFFFFF0073B944C	ffffff80007a4140	sub_FFFFFF80007A4140	0.9462963	16	15	Local affinity
00052	ffffff0072bbfc8	sub_FFFFFFF0072BBFC8	ffffff80006718e0	_tcp_sbrcv_trim	0.9455475	10	10	Local affinity

Line 4 of 4927

Output

```
[Diaphora: Fri Sep 13 15:24:30 2024] Finding with heuristic 'Related compilation unit'
[Diaphora: Fri Sep 13 15:29:04 2024] Finding locally affine functions
[Diaphora: Fri Sep 13 15:29:24 2024] Finding unmatched functions
[Diaphora: Fri Sep 13 15:29:24 2024] Final results: Best 844, Partial 4927, Unreliable 0, Multimatches 354
[Diaphora: Fri Sep 13 15:29:24 2024] Matched 53.73% of main binary functions (5771 out of 10741)
```

Training a specialized model yourself

Specialized models

- Diaphora since September 2024 distributes an already trained model and, also, in a different repository, the tools so you can train yourself your own models:
  - https://github.com/joxeankoret/diaphora-ml
- Basically, you will need to do the following:
  - Analyse all the binaries with IDA.
  - Export all of them with Diaphora.
  - Run a tool that parses Diaphora exported SQLite databases to generate a CSV file.
  - Use the generated CSV file to train a model.
  - Tell Diaphora in diaphora_config.py the path of your own model.

Specialized models

diaphora-ml / example / **run.sh**

joxeankoret Added example mini-dataset and script — d26e594 · 1 minute ago — History

Code | Blame — Executable File · 23 lines (19 loc) · 1.02 KB — Code 55% faster with GitHub Copilot — Raw

```
1   #!/bin/sh
2
3   #
4   # Run this script to generate a dataset out of the binaries in the directory with
5   # the name 'mini-dataset'. Remember to configure the IDA and Diaphora scripts
6   # paths in ../tools/diaphora-export.sh
7   #
8
9   # Run a maximum of ${CPUS} processes
10  CPUS=4
11
12  # Find all binaries and export them with Diaphora
13  find mini-dataset/ -type f | parallel -u -j ${CPUS} ../tools/diaphora-export.sh {}
14  # Create a CSV dataset cross comparing the generated SQLite databases
15  ../tools/create_dataset.py mini-dataset -o mini-dataset-example.csv
16  # Split the dataset into training, validation and testing
17  ../tools/split-dataset.py mini-dataset-example.csv
18  # Then, train a model using adecision tree classifier and output it to mini-dataset.pkl
19  ../tools/train_dataset.py -u 17 -v -o mini-dataset.pkl -t -d mini-dataset-example-train.csv
20  # And validate it against both the test & training subsets
21  ../tools/train_dataset.py -v -i mini-dataset.pkl -c mini-dataset-example-test.csv
22  ../tools/train_dataset.py -v -i mini-dataset.pkl -c mini-dataset-example-validate.csv
```

- https://github.com/joxeankoret/diaphora-ml/blob/main/example/run.sh

Specialized models

- After your model is generated, you will need to specify in your diaphora_config.py file the path of the model to be used:

```
203  #-----------------------------------------------------------------------------
204  # Diaphora can use a local mode, enable this configuration directive to use it.
205  ML_USE_TRAINED_MODEL = True
206  # Model trained with a decision tree classifier: fast and accurate enough
207  ML_TRAINED_MODEL = "/path/to/your/model/goes/here.pkl"
208  # The value added to the similarity ratio for a positive match using the model.
209  ML_TRAINED_MODEL_MATCH_SCORE = 0.15
210
211  # Show a chooser with all the matches that the classifier think are good ones?
212  ML_DEBUG_SHOW_MATCHES = True
```

Specialized models

- Now, let's suppose that you want to build your model but you also want to use the model supplied by Diaphora at the same time.
- What can you do? Here is the solution:
- Simply, concatenate your CSV with the Diaphora's supplied CSV file.
  - The Diaphora's CSV is in https://github.com/joxeankoret/diaphora-ml
- And then train your model with the concatenation of them.
- As simple as it sounds.

Conclusions

Conclusions

- It's not obvious at first sight what ML techniques might or might not work to apply to bindiffing, to be honest.
  - And this talk has only been about classification, not mapping.
- Building huge models from huge datasets, unfortunately, doesn't look appropriate for everybody. At least, it is not for me.
- However, building highly specialised datasets for very specific use-cases is easy to do and significantly improves results.
- It's more important, in my opinion, to release the tools to allow reversers to train their own models than actually distributing an already trained model, anyway.
- And this is why I now distribute with Diaphora the model, and also the tools (separately).

Download

- Everything is public already and, naturally, Open Source:
  - The ML tools: https://github.com/joxeankoret/diaphora-ml
  - The binary diffing tool: https://github.com/joxeankoret/diaphora
- Enjoy! Hopefully, it makes your RE life a bit better.
- Also, if you want, come join me for the Diaphora workshop right after this talk.

Questions anyone?

Thanks for your time!