SAFRAN

Release v1.0.0

Simon Ott, Matthias Samwald

Apr 26, 2022

CONTENTS

SAFRAN (Scalable and fast non-redundant rule application) is a framework for fast inference of groundings and aggregation of predictions of logical rules in the context of knowledge graph completion/link prediction. It uses rules learned by AnyBURL (Anytime Bottom Up Rule Learning), a highly-efficient approach for learning logical rules from knowledge graphs.

Warning: Currently only rules learned with the `AnyBURL-RE` version are supported. Further information can be found at the *Previous and Special Versions* section at the AnyBURL Homepage. You can use the version `AnyBURL-JUNO`, however you have to set *ZERO_RULES_ACTIVE = false* in the properties file for learning the rules.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

You can find download links to prebuilt executables (recommended) for your operating system below. If you want the latest development version you can build SAFRAN from source, which is described below.

1.1.1 Download

Prebuild binaries can be found here (only x64):

- Linux
- Windows (MSVC)
- Windows (MinGW)
- MacOS

1.1.2 Build from source

1. Clone the SAFRAN repository
2. Download and extract boost 1.76.0 to project root directory
    1. Windows: https://boostorg.jfrog.io/artifactory/main/release/1.76.0/source/boost_1_76_0.zip
    2. Unix: https://boostorg.jfrog.io/artifactory/main/release/1.76.0/source/boost_1_76_0.tar.gz
3. Have cmake installed (> 9.6.0)
4. Create and change to directory **build**
5. Run `cmake ../`
6. Run `make` (Unix) or `cmake --build .` (Windows)

CHAPTER

TWO

MANUAL

2.1 Command lines

- Windows

```
SAFRAN.exe {action} {path_to_config}
```

- Linux

```
./SAFRAN {action} {path_to_config}
```

where `action` is one of

applymax Application of rules using the MaxPlus aggregation

applynoisy Application of rules using the Noisy-OR aggregation

applynrnoisy Application of rules using the Non-redundant Noisy-OR aggregation

learnnrnoisy Clustering of rules for the Non-redundant Noisy-OR aggregation

calcjacc Calculation of similarity matrices of rules for each relation (needed for clustering)

and `path_to_config` is a path to a file containing the configuration of an action as key-value pairs. Details to the configuration of a specific action can be seen on the Actions page. The properties file should have the following format:

```
{KEY} = {VALUE}
{KEY} = {VALUE}
...
```

2.2 Action *applymax*

Application of rules using the Maximum aggregation approach:

$$score(e) = \max\{conf(r_1), \ldots, \text{conf}(r_k)\}$$

where score(e) is the maximum confidence of all rules $r_1, \ldots, r_k$ that predict entity e. If the maximum confidences of two or more entities are the same, these entities are further ranked by their second best confidence and so on, until all top-k candidates can be distinguished or all rules are processed.

2.2.1 Configuration file

Input :

- **PATH_TRAINING** [Valid path (file)] Path to training file (absolute or relative), **default: train.txt**
- **PATH_TEST** [Valid path (file)] Path to test file (absolute or relative), **default: test.txt**
- **PATH_VALID** [Valid path (file)] Path to validation file (absolute or relative), **default: valid.txt**
- **PATH_RULES** [Valid path (file)] Path to rules file (absolute or relative), **default: rules.txt**

Properties :

- **WORKER_THREADS** [int] Number of threads that are used for computation. (-1 means all threads are used), **default: -1**
- **DISCRIMINATION_BOUND** [int] Discriminates (omits) rules which predict more elements than this, 0 means no limit., **default: 4000**
- **UNSEEN_NEGATIVE_EXAMPLES** [int] The number of negative examples for which we assume that they exist, however, we have not seen them. Rules with high coverage are favoured the higher the chosen number, **default: 5**
- **REFLEXIV_TOKEN** [string] Token used for substitution of reflexive rules. (Used if AnyBURL ruleset was trained with REWRITE_REFLEXIV = TRUE), **default: me_myself_i**
- **TOP_K_OUTPUT** [int] The top-k results that are after filtering kept in the results, **default: 10**
- **PREDICT_UNKNOWN** [int] If set to 1, does not skip triples containing unkwown entities in the training set. F.e. generates predictions for *john speaks UNKOWN* if *UNKNOWN* is not in the training set. **default 0**
- **ONLY_XY** [int] If set to 1, only cyclic (XY) rules are read from the rules file, **default: 0**

Output :

- **PATH_OUTPUT** [Valid path (file)] Path to file used for storing predictions, **default: predictions.txt**

2.3 Action *applynoisy*

Application of rules using noisy-or aggregation approach:

$$score(e) = 1 - \prod_{i=1}^{k}(1 - conf(r_i))$$

where $r_1, \ldots, r_k$ are rules that predict entity e.

2.3.1 Configuration file

Input :

- **PATH_TRAINING** [Valid path (file)] Path to training file (absolute or relative), **default: train.txt**
- **PATH_TEST** [Valid path (file)] Path to test file (absolute or relative), **default: test.txt**
- **PATH_VALID** [Valid path (file)] Path to validation file (absolute or relative), **default: valid.txt**
- **PATH_RULES** [Valid path (file)] Path to rules file (absolute or relative), **default: rules.txt**

Properties :

- **WORKER_THREADS** [int] Number of threads that are used for computation. (-1 means all threads are used), **default: -1**
- **DISCRIMINATION_BOUND** [int] Discriminates (omits) rules which predict more elements than this, 0 means no limit., **default: 4000**
- **UNSEEN_NEGATIVE_EXAMPLES** [int] The number of negative examples for which we assume that they exist, however, we have not seen them. Rules with high coverage are favoured the higher the chosen number, **default: 5**
- **REFLEXIV_TOKEN** [string] Token used for substitution of reflexive rules. (Used if AnyBURL ruleset was trained with REWRITE_REFLEXIV = TRUE), **default: me_myself_i**
- **TOP_K_OUTPUT** [int] The top-k results that are after filtering kept in the results, **default: 10**
- **PREDICT_UNKNOWN** [int] If set to 1, does not skip triples containing unkwown entities in the training set. F.e. generates predictions for *john speaks UNKOWN* if *UNKNOWN* is not in the training set. **default 0**
- **ONLY_XY** [int] If set to 1, only cyclic (XY) rules are read from the rules file, **default: 0**

Output :

- **PATH_OUTPUT** [Valid path (file)] Path to file used for storing predictions, **default: predictions.txt**

2.4 Action *applynrnoisy*

Caution: Needs clusters from action *learnnrnoisy*

Application of Non-redundant Noisy-OR (NRNO) to the test set (requires learned clusters). NRNO tries to overcome the problem of redundancy when using the Noisy-OR aggregation method by clustering rules based on their redundancy degree prior to aggregation. Predictions of rules in a cluster are aggregated using the Maximum approach, as this approach is not susceptible to redundancies. Predictions of the different clusters are then further aggregated using the Noisy-OR approach. As a metric for redundancy between two rules r_i, r_j the Jaccard Index $sim(r_i, r_j) = |\hat{H}_{r_i} \cap \hat{H}_{r_j}|/|\hat{H}_{r_i} \cup \hat{H}_{r_j}|$ of the sets of inferred triples is used. As the calculation of the Jaccard coefficient is very inefficient for large sets, the Jaccard coefficient is estimated using the MinHash scheme, which makes time complexity linear and memory usage constant.

2.4.1 Configuration file

Input :

- **PATH_TRAINING** [Valid path (file)] Path to training file (absolute or relative), **default: train.txt**
- **PATH_TEST** [Valid path (file)] Path to test file (absolute or relative), **default: test.txt**
- **PATH_VALID** [Valid path (file)] Path to validation file (absolute or relative), **default: valid.txt**
- **PATH_RULES** [Valid path (file)] Path to rules file (absolute or relative), **default: rules.txt**
- **PATH_CLUSTER** [Valid path (file)] Path to clustering file, **default: cluster.txt**

Properties :

- **WORKER_THREADS** [int] Number of threads that are used for computation. (-1 means all threads are used), **default: -1**

- **DISCRIMINATION_BOUND** [int] Discriminates (omits) rules which predict more elements than this, 0 means no limit., **default: 4000**
- **UNSEEN_NEGATIVE_EXAMPLES** [int] The number of negative examples for which we assume that they exist, however, we have not seen them. Rules with high coverage are favoured the higher the chosen number, **default: 5**
- **REFLEXIV_TOKEN** [string] Token used for substitution of reflexive rules. (Used if AnyBURL ruleset was trained with REWRITE_REFLEXIV = TRUE), **default: me_myself_i**
- **TOP_K_OUTPUT** [int] The top-k results that are after filtering kept in the results, **default: 10**
- **PREDICT_UNKNOWN** [int] If set to 1, does not skip triples containing unkwown entities in the training set. F.e. generates predictions for *john speaks UNKOWN* if *UNKNOWN* is not in the training set. **default 0**
- **ONLY_XY** [int] If set to 1, only cyclic (XY) rules are read from the rules file, **default: 0**

Output :

- **PATH_OUTPUT** [Valid path (file)] Path to file used for storing predictions, **default: predictions.txt**

2.5 Action *learnnrnoisy*

Caution: Needs similarity matrices from action *calcjacc*

Learning of the optimal thresholds for the clustering used by non redundant noisy or (requires similarity matrices)

2.5.1 Configuration file

Input :

- **PATH_TRAINING** [Valid path (file)] Path to training file (absolute or relative), **default: train.txt**
- **PATH_TEST** [Valid path (file)] Path to test file (absolute or relative), **default: test.txt**
- **PATH_VALID** [Valid path (file)] Path to validation file (absolute or relative), **default: valid.txt**
- **PATH_RULES** [Valid path (file)] Path to rules file (absolute or relative), **default: rules.txt**
- **PATH_JACCARD** [Valid path (directory)] Path to directory containing jaccard files, **default: jaccard/**

Properties :

- **WORKER_THREADS** [int] Number of threads that are used for computation. (-1 means all threads are used), **default: -1**
- **DISCRIMINATION_BOUND** [int] Discriminates (omits) rules which predict more elements than this, 0 means no limit., **default: 4000**
- **UNSEEN_NEGATIVE_EXAMPLES** [int] The number of negative examples for which we assume that they exist, however, we have not seen them. Rules with high coverage are favoured the higher the chosen number, **default: 5**
- **REFLEXIV_TOKEN** [string] Token used for substitution of reflexive rules. (Used if AnyBURL ruleset was trained with REWRITE_REFLEXIV = TRUE), **default: me_myself_i**

- **BUFFER_SIZE** [int] Buffer size (in amount of integers, 4 byte) used to limit memory consumption of buffering previously inferred rules. Should only be set if running out of memory. (2500000000 –> ~10 GB), **default: Maximum unsigned long long**
- **TOP_K_OUTPUT** [int] Top-K predictions that are used to calculate the MRR for hyperparameter search, **default: 10**
- **RESOLUTION** [int] Sets the accuracy of the Jaccard estimation. The number of hash functions used in MinHash (f.e. RESOLUTION = 200 –> 200 hash functions –> Max resolution of Jaccard 1/200), **default: 200**
- **STRATEGY** [[grid|random]] Sets the search strategy to be used for finding optimal clustering, **default: grid**
- **ITERATIONS** [int] Amount of iterations used in random search strategy, **default: 10000**
- **SEED** [int] Seed for the sampling of thresholds used in random search strategy, **default: 0**
- **VERBOSE** [int] If set to 1, writes the MRR and settings (thresholds) of each iteration of the hyperparameter search to seperate files named {relation}_chk.txt, **default: 0**
- **ONLY_XY** [int] If set to 1, only cyclic (XY) rules are read from the rules file, **default: 0**

Output :

- **PATH_CLUSTER** [Valid path (file)] Path to clustering file, **default: cluster.txt**

2.6 Action *calcjacc*

Calculates the similarity matrices (Jaccard index) used by Non-redundant Noisy-OR for each relation.

2.6.1 Configuration file

Input :

- **PATH_TRAINING** [Valid path (file)] Path to training file (absolute or relative), **default: train.txt**
- **PATH_TEST** [Valid path (file)] Path to test file (absolute or relative), **default: test.txt**
- **PATH_VALID** [Valid path (file)] Path to validation file (absolute or relative), **default: valid.txt**
- **PATH_RULES** [Valid path (file)] Path to rules file (absolute or relative), **default: rules.txt**

Properties :

- **WORKER_THREADS** [int] Number of threads that are used for computation. (-1 means all threads are used), **default: -1**
- **DISCRIMINATION_BOUND** [int] Discriminates (omits) rules which predict more elements than this, 0 means no limit., **default: 4000**
- **REFLEXIV_TOKEN** [string] Token used for substitution of reflexive rules. (Used if AnyBURL ruleset was trained with REWRITE_REFLEXIV = TRUE), **default: me_myself_i**
- **SEED** [int] Seed for generating hash functions used in MinHash, **default: 0**
- **RESOLUTION** [int] Sets the accuracy of the Jaccard estimation. The number of hash functions used in MinHash (f.e. RESOLUTION = 200 –> 200 hash functions –> Max resolution of Jaccard 1/200), **default: 200**
- **ONLY_XY** [int] If set to 1, only cyclic (XY) rules are read from the rules file, **default: 0**

Output :

- **PATH_JACCARD** [Valid path (directory)] Path to directory for storing binary files containing similarity matrices for each relation. *0_jacc.bin* f.e. is the similarity matrix of relation with ID 0., **default: jaccard/**

2.7 Evaluate predictions

Following scripts can be found in the *python* folder.

2.7.1 Evaluate a single prediction file

Use this script to evaluate a single prediction file.

Script name: eval.py

Requires:

```
pip install scipy tqdm
```

Usage:

```
python eval.py {path to file containing predictions} {path to testset file}
```

Example output:

```
MRR: 0.389
Hits@1: 0.298
Hits@3: 0.371
Hits@10: 0.537
```

2.7.2 Evaluate an experiment

Use this script if you want to evaluate multiple datasets containing multiple prediction files at once (Multiple datasets -> Multiple prediction files).

Script name: eval_experiment.py

Requires:

```
pip install scipy tqdm
```

Usage:

```
python eval_experiment.py --datasets {list of datasets} --predictions {list of␣
↪prediction file names}
```

File structure:

Each dataset should have its own folder. Evaluations are run

```
for each {dataset} in {list of datasets}:
   for each {prediction file name} in {list of prediction file name}:
      Path to prediction file: f"./{dataset}/predictions/{prediction file name}"
      Path to testset file: f"./{dataset}/data/test.txt"
```

Example:

```
python eval_experiment.py --datasets OBL WN18RR --predictions predfile1.txt predfile2.txt
```

```
---- OBL
    |
    ---- predictions
        |
        ---- predfile1.txt
        |
        ---- predfile2.txt
    |
    ---- data
        |
        ---- test.txt
---- WN18RR
    |
    ---- predictions
        |
        ---- predfile1.txt
        |
        ---- predfile2.txt
    |
    ---- data
        |
        ---- test.txt
```

Output:

```
OBL
predfile1.txt MRR: 0.389 Hits@1: 0.298 Hits@3: 0.371 Hits@10: 0.537
predfile2.txt MRR: 0.389 Hits@1: 0.298 Hits@3: 0.371 Hits@10: 0.537

WN18RR
predfile1.txt MRR: 0.389 Hits@1: 0.298 Hits@3: 0.371 Hits@10: 0.537
predfile2.txt MRR: 0.389 Hits@1: 0.298 Hits@3: 0.371 Hits@10: 0.537
```

2.8 Format of the rule file

Caution: Whitespaces in entity and relation names are not supported!

Safran currently only supports rules in the format of AnyBURL:

```
{predicted: int}\t{correctlyPredicted: int}\t{confidence: double}\t{rule}
```

where `predicted` is the absolute number of all predictions made by the rule and `correctlyPredicted` is the absolute number of all predictions that are correct. `confidence` is redundant and never is used within SAFRAN as it gets calculated as

$$confidence = \frac{correctlyPredicted}{predicted + Properties.UNSEEN\_NEGATIVE\_EXAMPLES}$$

2.8.1 AnyBURL rule format

`rule` is a rule in the format of AnyBURL:

```
{headatom} <= {(bodyatom)+}
```

where the seperator of `headatom` and *(bodyatom)+* is " `<=` " (notice the whitespaces).

`headatom` is the head atom of a rule:

- Cyclic rules: can be only `{r}(X,Y)`, with `r` being a valid relation
- Acyclic rules: can be either `{r}(X,{c})` or `{r}({c},Y)`, with `r` being a valid relation and `c` being a valid constant entity.

`(bodyatom)+` is a list of bodyatoms:

- Bodyatoms are ordered from 'X' to 'Y'
- Adjacent bodyatoms have to be connected by a single variable
- If `head` contains 'X': First body atom has to contain 'X'
- If `head` contains 'Y': Last body atom has to contain 'Y'
- Bodyatoms are seperated by ", " (notice the whitespace)

Example of valid rules

```
1302        165     0.126728111     hasNeighbor(X,Y) <= dealsWith(Y,X)
1302        164     0.125960061     hasNeighbor(X,Y) <= dealsWith(X,Y)
2000        216     0.108   GENE_GO(X,Y) <= GENE_CATALYSIS_GENE(X,A), GENE_CATALYSIS_
↪GENE(A,B), GENE_GO(B,Y)
2000    73      0.0365  GENE_REACTION_GENE(X,Y) <= GENE_CATALYSIS_GENE(A,X), GENE_PTMOD_
↪GENE(B,A), GENE_BINDING_GENE(Y,B)
42      3       0.07142857142857142     P106(X,Q520549) <= P101(X,Q413)
29  2       0.06896551724137931     P27(Q75612,Y) <= P30(Y,Q49)
194 3       0.015463917525773196    P27(Q75612,Y) <= P30(Y,A)
50  2       0.04    P27(Q154756,Y) <= P551(A,Y)
959 7       0.0072992700729927005   P27(X,Q27) <= P27(X,A)
238 6       0.025210084033613446    P136(X,Q676) <= P737(A,X)
```

2.9 Creating explanation file

Safran currently supports the creation of explanations for predictions, as needed f.e. for **`https://github.com/OpenBioLink/Explorer/ LinkExplorer`_**.

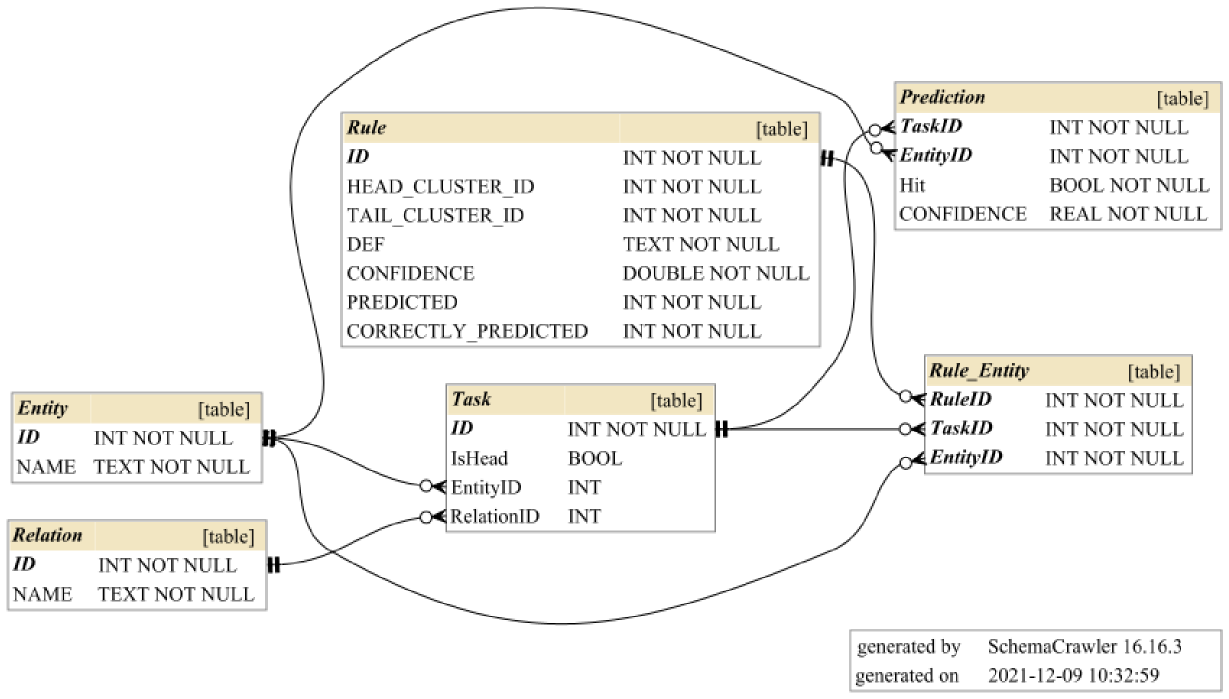
2.9.1 Building SAFRAN with explanation capabilities

1. Checkout the branch *explanation* of the SAFRAN repository
2. Download and extract boost 1.76.0 to project root directory
    1. Windows: https://boostorg.jfrog.io/artifactory/main/release/1.76.0/source/boost_1_76_0.zip
    2. Unix: https://boostorg.jfrog.io/artifactory/main/release/1.76.0/source/boost_1_76_0.tar.gz
3. Have cmake installed (> 9.6.0)
4. Create and change to directory **build**
5. Run `cmake ../`
6. Run `make` (Unix) or `cmake --build .` (Windows)

2.9.2 Create explanations

You can now run the built SAFRAN executable with the actions *applynoisy*, *applymax*, *applynrnoisy* and *EXPLAIN = 1* in the properties file. This creates a sqlite database file that stores the dataset, predictions and their explanations.

2.9.3 Explanation DB schema



CHAPTER

THREE

BENCHMARK RESULTS

3.1 Benchmark Results

3.1.1 FB15k-237

- *Dataset <https://github.com/TimDettmers/ConvE/blob/master/FB15k-237.tar.gz>*
- *Ruleset <https://zenodo.org/record/5517166/files/fb15k237-1000.zip?download=1>*

Table 1: FB15k-237

	MRR	Hits@1	Hits@10
Maximum	.355	.270	.519
Noisy-OR	.342	.277	.502
Non-redundant Noisy-OR	.389	.298	.537

3.1.2 WN18RR

- *Dataset <https://github.com/TimDettmers/ConvE/blob/master/WN18RR.tar.gz>*
- *Ruleset <https://zenodo.org/record/5517166/files/wn18rr-1000.zip?download=1>*

Table 2: WN18RR

	MRR	Hits@1	Hits@10
Maximum	.494	.452	.572
Noisy-OR	.454	.399	.562
Non-redundant Noisy-OR	.502	.459	.578

3.1.3 YAGO3-10

- *Dataset <https://github.com/TimDettmers/ConvE/blob/master/YAGO3-10.tar.gz>*
- *Ruleset <https://zenodo.org/record/5517166/files/yago310-1000.zip?download=1>*
- [Dataset](https://github.com/TimDettmers/ConvE/blob/master/YAGO3-10.tar.gz)
- [Ruleset](https://zenodo.org/record/5517166/files/yago310-1000.zip?download=1)

Table 3: YAGO3-10

	MRR	Hits@1	Hits@10
Maximum	.559	.487	.686
Noisy-OR	.524	.444	.672
Non-redundant Noisy-OR	.564	.491	.693

3.1.4 OpenBioLink

- *Dataset <https://github.com/OpenBioLink/OpenBioLink> (ready to use with AnyBURL <https://zenodo.org/record/5517166/files/obl-dataset.zip?download=1>)*
- *Ruleset <https://zenodo.org/record/5517166/files/obl-1000.zip?download=1>*

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search