Model-Driven Engineering

Projects

RANDRIANAINA Georges Aaron

georges-aaron.randrianaina@irisa.fr

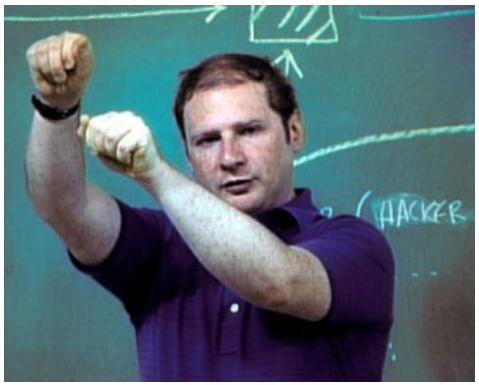
M2 IL, ISTIC, Université de Rennes

2023–2024

Structure and Interpretation of Computer Programs

Chapter 4 – Metalinguistic Abstraction

> *"We must constantly turn to **new languages** in order to **express our ideas more effectively**. Establishing **new languages** is a **powerful strategy** for **controlling complexity** in engineering design; we can often **enhance our ability to deal with a complex problem** by adopting a new language that **enables us to describe** (and hence to think about) the problem in a different way, using primitives, means of combination, and **means of abstraction that are particularly well suited to the problem** at hand."*



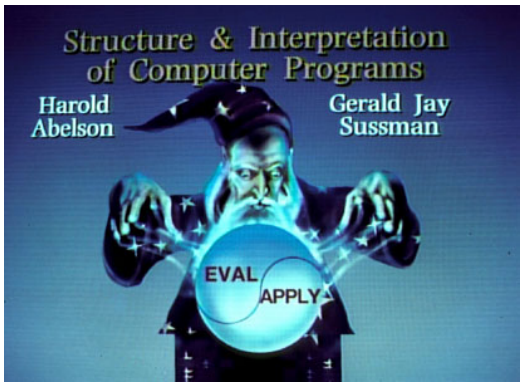


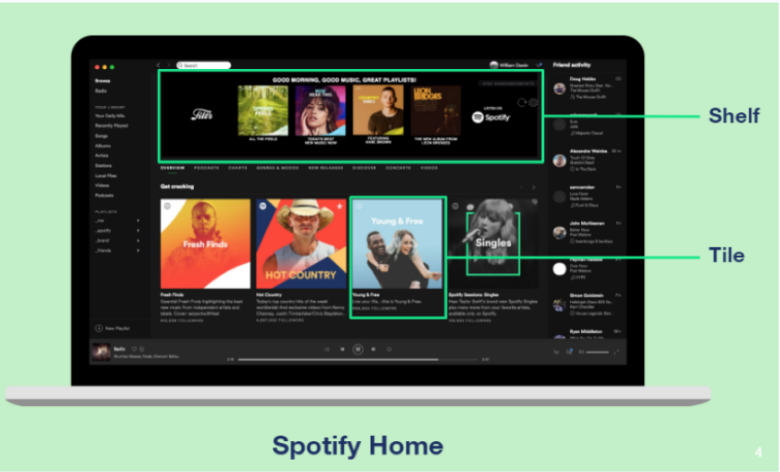
Table of contents

Table of contents

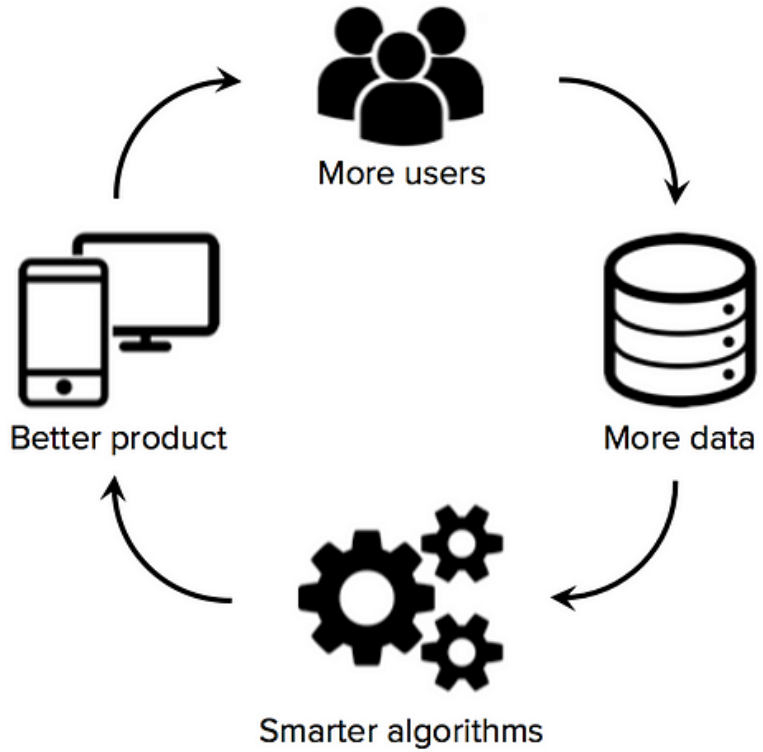
Context: ML-Enabled Software Systems[1]

For Your Ears Only: Personalizing Spotify Home with Machine Learning

January 16, 2020
Published by Spotify Engineering



This article is based on the keynote given by Tony Jebara at TensorFlow World in Santa Clara, California, October 2019. You can watch the presentation here.



[1] Inspired by Christian Kästner's course: Software Engineering for AI-Enabled Systems (SE4AI)

...

JSON

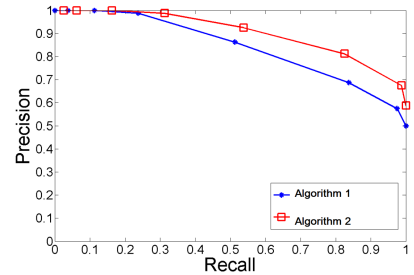
Web/IoT data

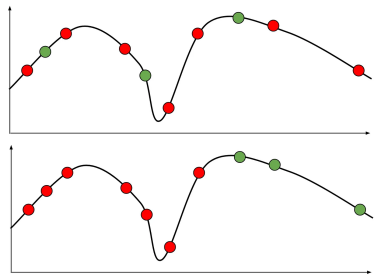
CSV

Tabular data

ML

Prediction model





...

JSON

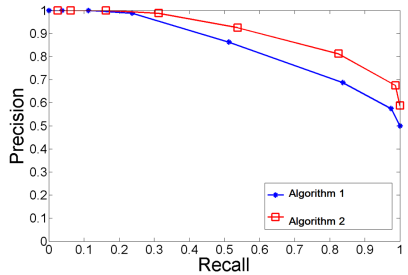
Web/IoT data

CSV

Tabular data

ML

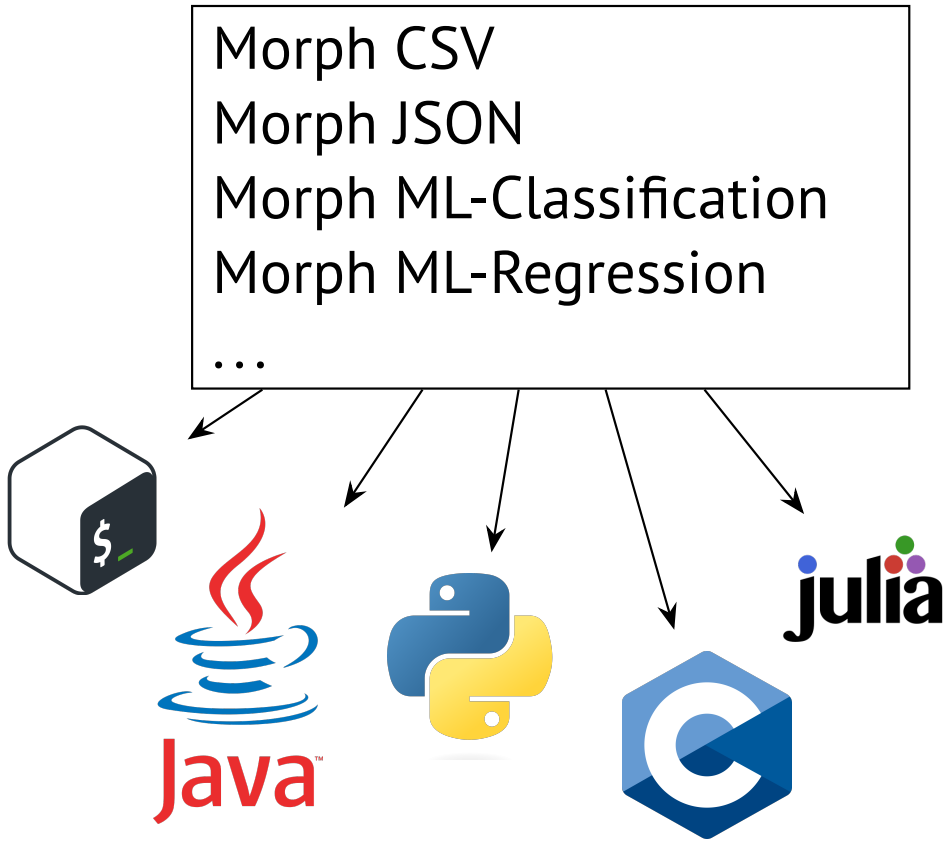
Prediction model



“En gros on va vous montrer des méthodes et des démarches qui permettent de construire des immeubles et des bâtiments complexes en vous demandant de les appliquer pour construire des niches.”
– Jean-Marc Jézéquel

Goals

- Finding functional/performance bugs
- Improve learning curve of APIs/libraries in a domain
- Choosing the good-enough/best morph for a given “task”
- Identifying good test suites/benchmarks for a domain
- Portoflio (“meta”) solution[a] *e.g.*, can be run in parallel



[a] e.g., *SATzilla: Portfolio-based Algorithm Selection for SAT*, Xu et al., Journal of Artificial Intelligence Research, 2008

1. DSL for JSON

1. Concepts
   - Load, store JSON files
   - Select subset of objects, Projection (slice of objet)
     a.k.a core relational algebra operators
   - Compute basics Σ, Π over fields
   - Print field value, #objects, #fields, depth, expressions…
   - Insert/modify/remove object/fields
2. Services
   - Export to CSV
   - Interpreter
   - Compiler to (Java or Python or Julia) + JQ[2] (for relevant subset)

[2] `https://jqlang.github.io/jq/`

2. DSL for CSV

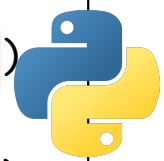
1. Concepts
   - Load, store CSV files
   - Select subset of lines/column (cut)
     a.k.a core relational algebra operators
   - Compute basics Σ, Π over fields
   - Print field value, #objects, #fields, depth, expressions…
   - Insert/modify/remove object/fields
2. Services
   - Export to JSON
   - Interpreter
   - Compiler to (Java or Python or Julia) + bash (using `grep`/`cut`/`awk`…)

Example: Polymorphic CSV

```
(with-open-csv (stream "/tmp/test.csv")
    (get-#rows stream))
```

```
wc -l /tmp/test.csv
```

```
import csv
a = open("/tmp/test.csv", 'rt')
a_read = csv.reader(a)
print(sum(1 for row in a_read))
```



```
#include <stdio.h>

int main() {
    const char* csv_file_path = "data.csv";
    FILE* file = fopen(csv_file_path, "r");
    if (file == NULL) {
        perror("Failed to open the CSV file");
        return 1;
    }
    int row_count = 0;
    char line[1024];
    while (fgets(line, sizeof(line), file)) {
        row_count++;
    }
    fclose(file);
    printf("%d\n", row_count);

    return 0;
}
```

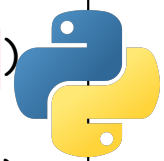
Example: Polymorphic CSV

Spot the Bug!

```
(with-open-csv (stream "/tmp/test.csv")
  (get-#rows stream))
```

```
wc -l /tmp/test.csv
```

```
import csv
a = open("/tmp/test.csv", 'rt')
a_read = csv.reader(a)
print(sum(1 for row in a_read))
```



```
#include <stdio.h>

int main() {
    const char* csv_file_path = "data.csv";
    FILE* file = fopen(csv_file_path, "r");
    if (file == NULL) {
        perror("Failed to open the CSV file");
        return 1;
    }
    int row_count = 0;
    char line[1024];
    while (fgets(line, sizeof(line), file)) {
        row_count++;
    }
    fclose(file);
    printf("%d\n", row_count);

    return 0;
}
```

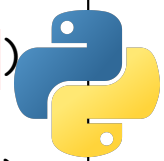
Example: Polymorphic CSV

Spot the Bug!

```
(with-open-csv (stream "/tmp/test.csv")
  (get-#rows stream))
```

```
wc -l /tmp/test.csv
```

```
import csv
a = open("/tmp/test.csv", 'rt')
a_read = csv.reader(a)
print(sum(1 for row in a_read))
```



Memory leak! `a.close()`

```
#include <stdio.h>

int main() {
    const char* csv_file_path = "data.csv";
    FILE* file = fopen(csv_file_path, "r");
    if (file == NULL) {
        perror("Failed to open the CSV file");
        return 1;
    }
    int row_count = 0;
    char line[1024];
    while (fgets(line, sizeof(line), file)) {
        row_count++;
    }
    fclose(file);
    printf("%d\n", row_count);

    return 0;
}
```

3. DSL for ML Classification

1. Concepts
   - Statistical classification[3]
     Typically uses CSV file as input *e.g.*, Iris flower data set[4]
   - Evaluation strategy, either:
     - dataset is split in two (training/test), with user defined % training;
     - cross-validation (provide means to parameterize it).
   - Predictive variables and target variable can be specified
     By default, all variables are predictive except last column of the CSV (target)
   - Specify what to calculate: accuracy and/or recall and/or f1
   - Which algorithm(s) to use (*e.g.*, classification tree or SVM for scikit-learn)
2. Services
   - Interpreter (*i.e.* classification using "random" algorithm) usefull to provide baselines
   - Compiler to Python/scikit-learn + (R or Julia)

[3] `https://en.wikipedia.org/wiki/Statistical_classification`
[4] `https://en.wikipedia.org/wiki/Iris_flower_data_set`

Example of scikit-learn code to be generated

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
# Using pandas to import the dataset
df = pd.read_csv("iris.csv")

# Spliting dataset between features (X) and label (y)
X = df.drop(columns=["variety"])
y = df["variety"]

# Spliting dataset into training set and test set
test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)

# Set algorithm to use
clf = tree.DecisionTreeClassifier()

# Use the algorithm to create a model with the training set
clf.fit(X_train, y_train)

# Compute and display the accuracy
accuracy = accuracy_score(y_test, clf.predict(X_test))
print(accuracy)
```

RTFM![5]

[5] https://pandas.pydata.org/docs/, https://scikit-learn.org/stable/,...

4. DSL for ML Regression

1. Concepts
   - Evaluation strategy, either:
     - dataset is split in two (training/test), with user defined % training;
     - cross-validation (provide means to parameterize it).
   - Predictive variables and target variable can be specified
     By default, all variables are predictive except last column of the CSV (target)
   - Specify what to calculate: mean relative error…
   - Which algorithm(s) to use (*e.g.*, classification tree or SVM for scikit-learn)
2. Services
   - Interpreter (*i.e.* regression using “random” algorithm)
     usefull to provide baselines
   - Compiler to Python/scikit-learn + (R or Julia)

Table of contents

Tasks

1. Chose a subproject **<now!>**
   - Each subproject should be taken at least once: JSON/CSV/ML-C/ML-R
   - Working in group of 2
   - Insert group composition into the following spreedsheet **<tbd>**
2. Build a first version of your metamodel **<tbd>**
   - Scan of handwritten or pdf diagram is enough at this stage
3. Build concrete syntax + parser **<tbd>**
4. Build an interpreter **<tbd>**
5. Build a compiler #1 **<tbd>**
6. Build a compiler #2 **<tbd>**
7. Make sure to interoperate with 2 complementary subprojects **<tbd>**
   - Show test case, if ok $\Rightarrow$ bonus for all 3 teams.
8. Project defense