Knowledge-driven AutoML

...~A future current-trend in AI?~...

by CORNELIU COFARU

= 10^{th} of May 2024 =

VUB

ARTIFICIAL
INTELLIGENCE
RESEARCH GROUP

- Outline

Outline

- AutoML - short introduction
- Knowledge-driven AutoML
  - The „Omina" project
  - Architecture & Motivation
  - Pipeline generation flow
- Open research challenges
  - mining & sharing experiments
  - wrapping ML-libs - inverting dependencies
  - optimization & evaluation

Auto ML

Auto ML

- The process of generating & tuning ML pipelines automatically

Auto ML

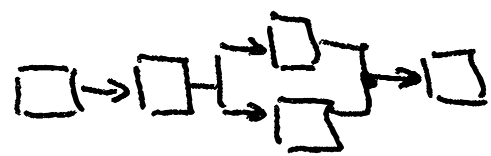
- The process of generating & tuning ML pipelines automatically

- Can be split into two parts:

Auto ML

- The process of generating & tuning ML pipelines automatically

- Can be split into two parts:

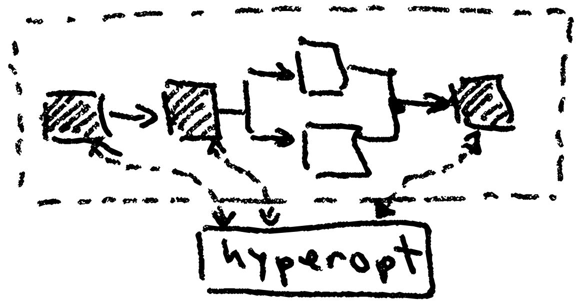
  1. pipe synthesis 

Auto ML

- The process of generating & tuning ML pipelines automatically

- Can be split into two parts:

  1. pipe synthesis
  2. hyperparameter optimization



Auto ML

- Most approaches consider the pipe structure fixed (auto-Weka, auto-sklearn)

Auto ML

- Most approaches consider the pipe structure fixed (auto-Weka, auto-sklearn)

- New research (~2016) considers pipe structure flexible : trees, rules, planning

trees → evolutionary

rules → logic

planning → ordering of blocks

Auto ML

- Most approaches consider the pipe structure fixed (auto-Weka, auto-sklearn)

- New research (~2016) considers pipe structure flexible : trees, rules, planning

  trees → evolutionary

  rules → logic

  planning → ordering of blocks

- Sophisticated and flexible use of knowledge structures still not explored !

Auto ML

- Note: Neural-networks case i.e neural arch. search - not covered here :)

- More info on: https://automl.org/book

Knowledge-driven AutoML

Knowledge-driven AutoML

- Data-driven technique for AutoML
- Available at:
  * Experiments: https://github.com/zgornel/knowledge-driven-automl
  * Library: https://github.com/zgornel/Kdautoml.jl

The Omina project

The Omina project

- Omina Technologies : AI company based in Antwerp developing ML solutions

The Omina project

- Omina Technologies: AI company based in Antwerp developing ML solutions

- Moving from manual to automatic pipeline building

The Omina project

- Omina Technologies: AI company based in Antwerp developing ML solutions

- Moving from manual to automatic pipeline building

WHY?

- improve development quality
- adapt to limited data access
- scale: performance, use-cases etc.

The Omina project

- The goal of the project: use knowledge to drive & speed up ML pipe building

The Omnia project

- The goal of the project: use knowledge to drive & speed up ML pipe building


Knowledge

- data science: code, pipes etc.
- processes: how entities interrelate and generate data
- world: open data, knowledge bases, wiki's ...

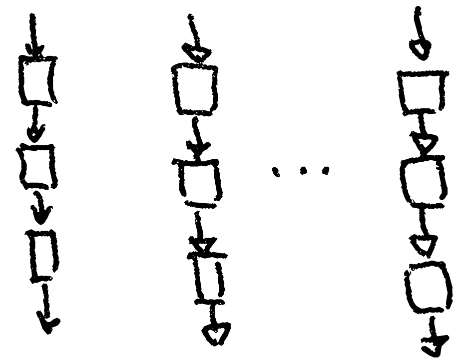
Architecture & Motivation

Architecture & Motivation

- Two approaches become apparent:

Architecture & Motivation

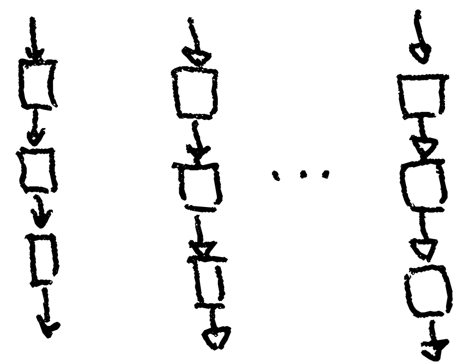
- Two approaches become apparent:
  - → generate complete pipelines first (a bunch of them, test, optimize, etc.)



Architecture & Motivation

- Two approaches become apparent:

  → generate complete pipelines first (a bunch of them, test, optimize, etc.)

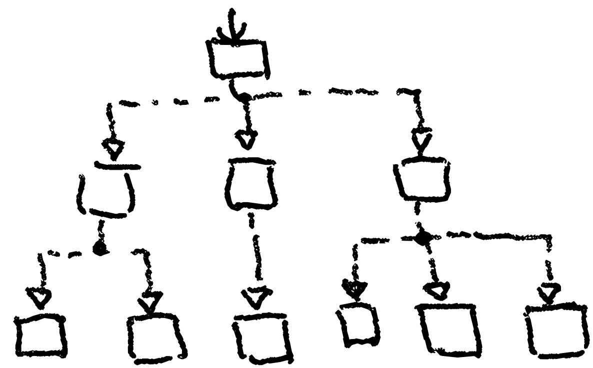


<u>Knowledge</u> used for:

- generating pipes
- post-process / combine them.
  i.e. optimization

Architecture & Motivation

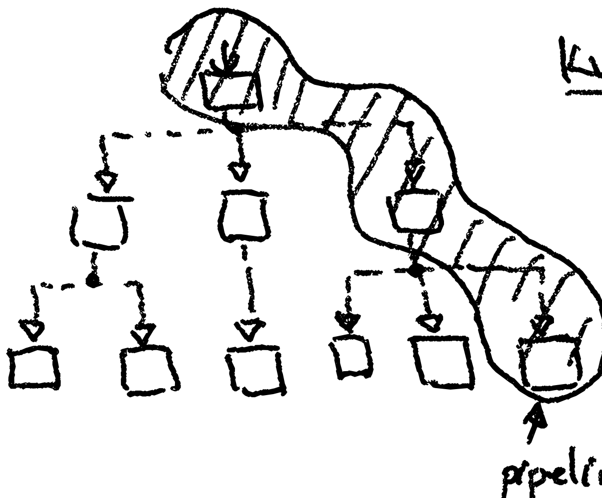
- Two approaches become apparent:
  - → generate and execute dynamically pipes



Architecture & Motivation

- Two approaches become apparent:

  → generate and execute dynamically pipes



<u>Knowledge</u> used to

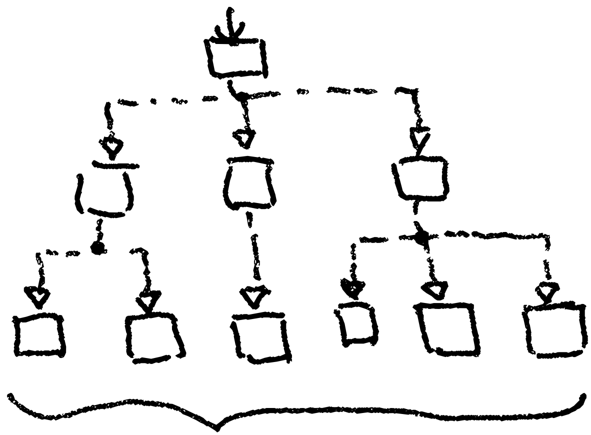
- decide what ops. are feasible based on:
  - → current operation
  - → partial pipe(s)
  - → execution results
  - → data science knowledge

Architecture & Motivation

- The latter „dynamic" approach chosen:
  - → better control over program space
    - → granularity in knowledge application
  - → more explanatory power
  - → allows for interactivity i.e. human intervention
    - → delays, partial execution etc.
  - → [potentially] more resource friendly
  - → the first one was already explored (by MIT)

Architecture & Motivation

Architecture & Motivation

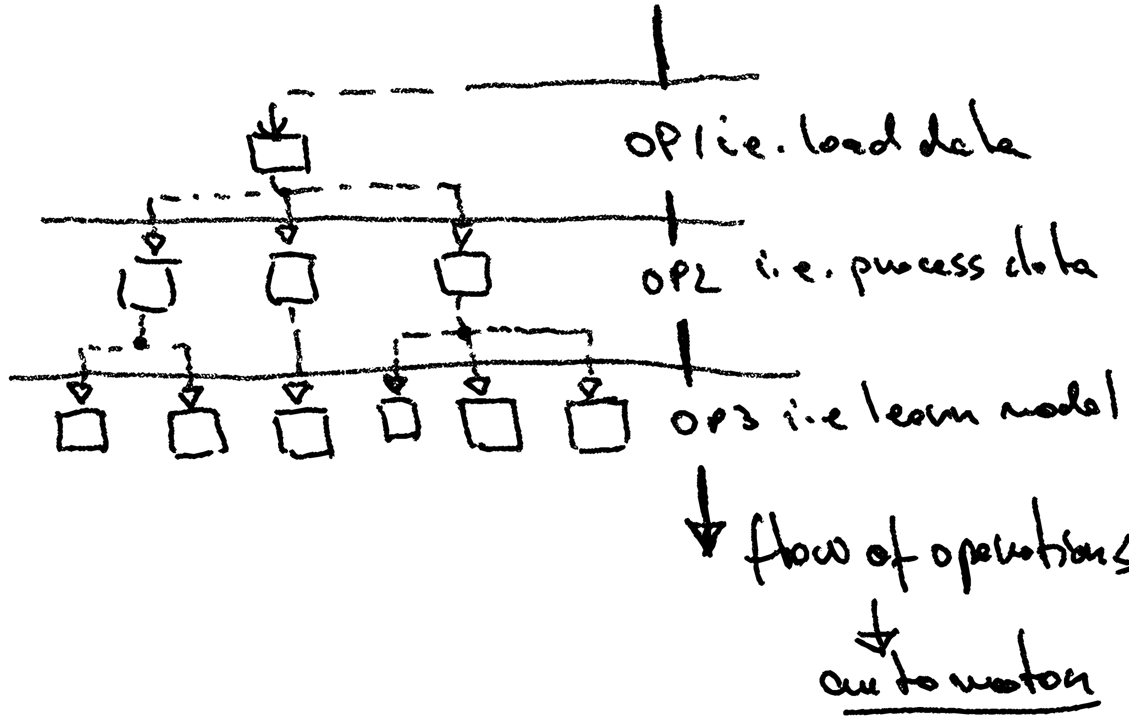


<u>program</u> <u>space</u> i.e all possible pipes

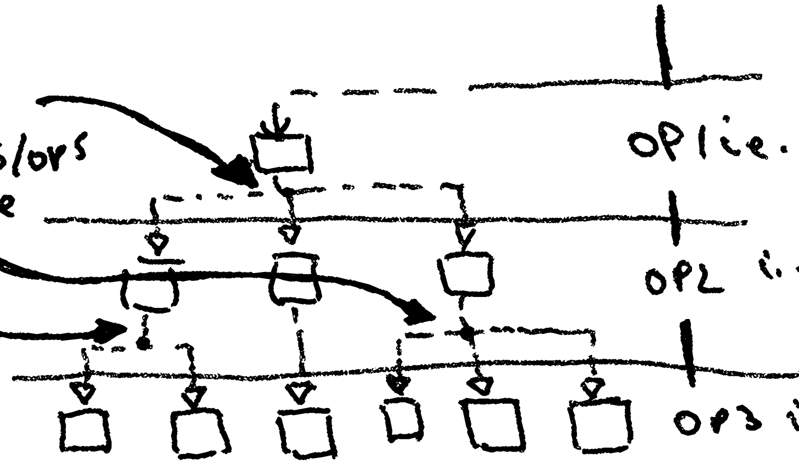
↓

<u>how</u> to construct it?

Architecture & Motivation



Architecture & Motivation



decisions on what blocks/ops are feasible

↓

Knowledge

↓

graph db

OP1 i.e. load data

OP2 i.e. process data

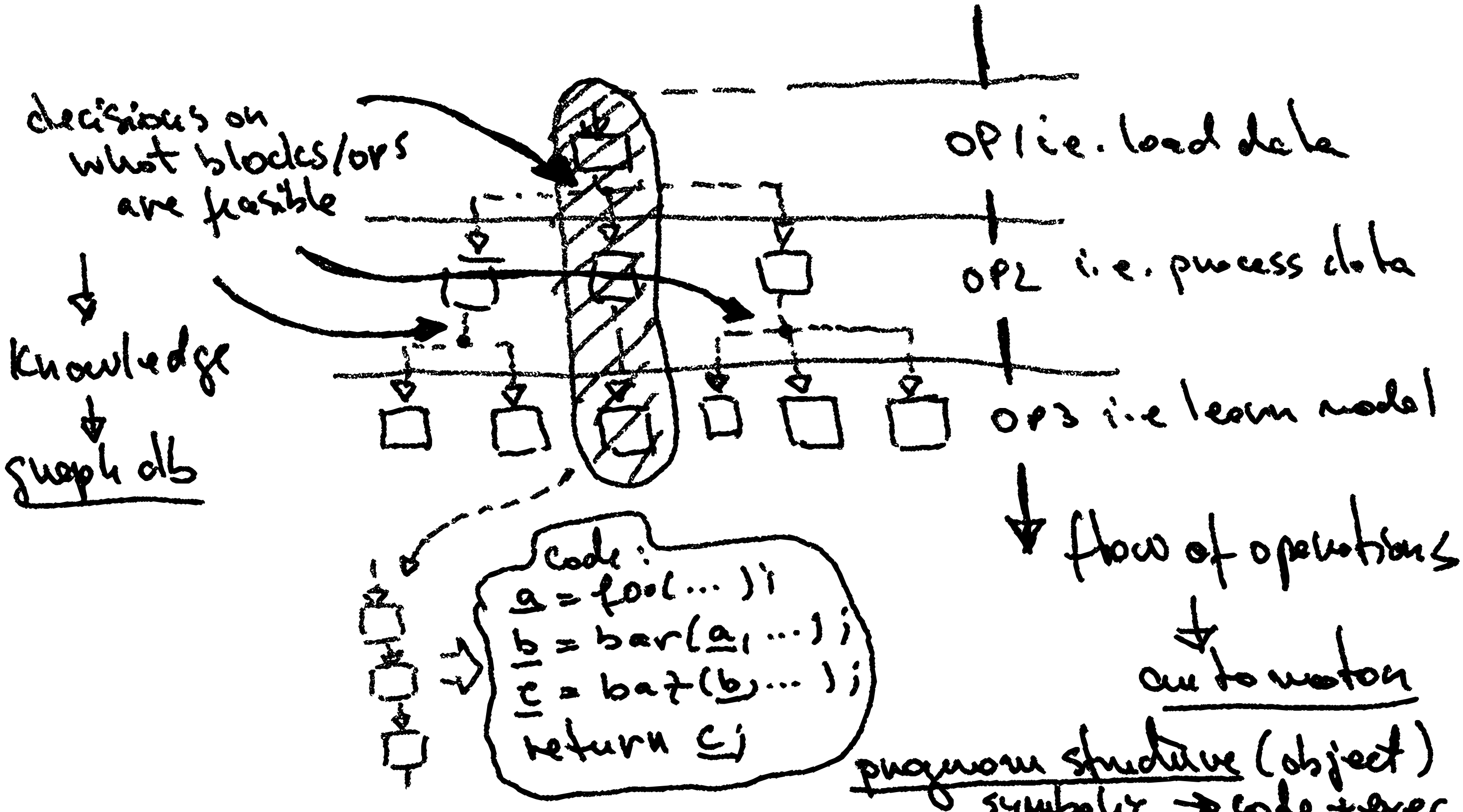
OP3 i.e. learn model

↓ flow of operations

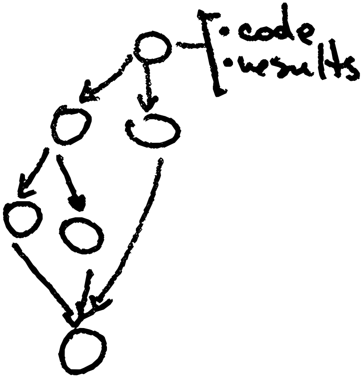
↓

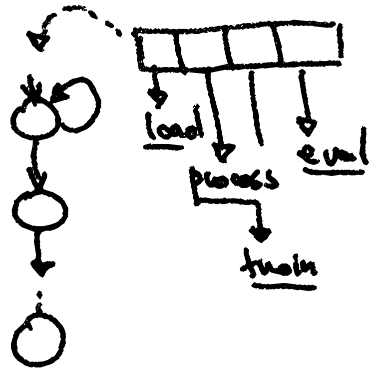
automaton

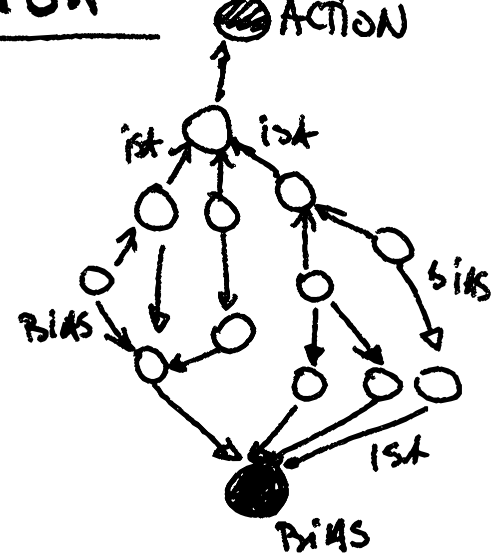
Architecture & Motivation



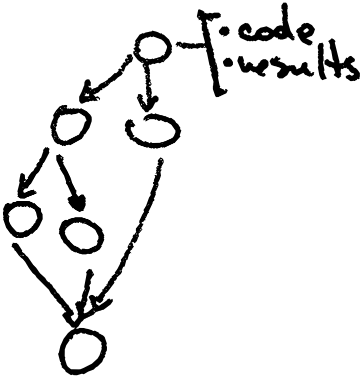
Architecture & Motivation

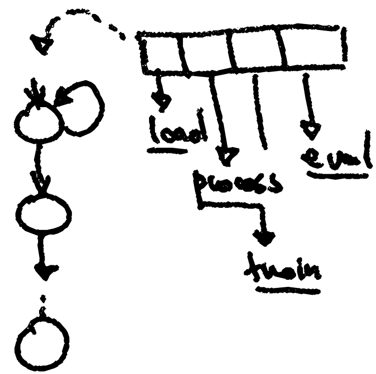


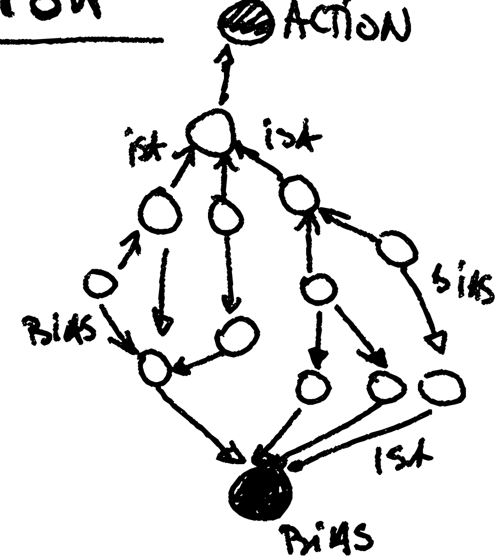




Architecture & Motivation



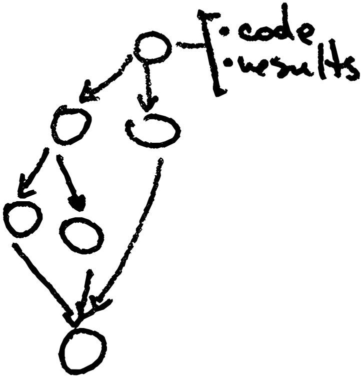


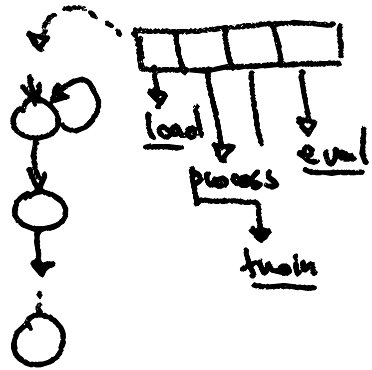


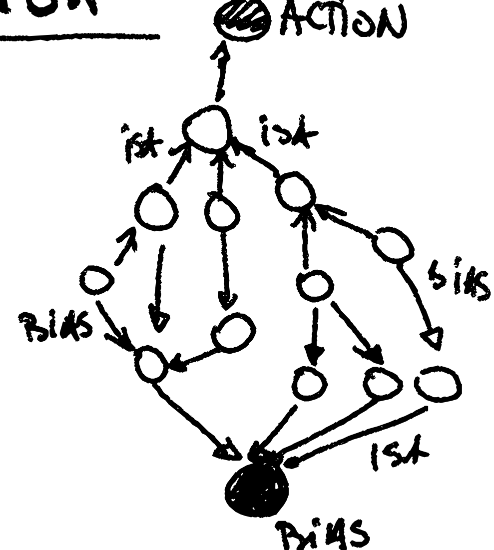
Automaton

- controls build flow
- communicates w. kb
- sends code / gets data from prep. structure

Architecture & Motivation







Automaton

- controls build flow
- communicates w. Kb
- sends code / gets data from prep-structure

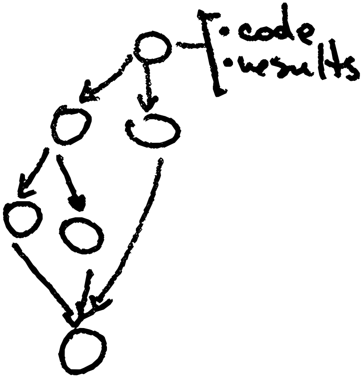
Kb

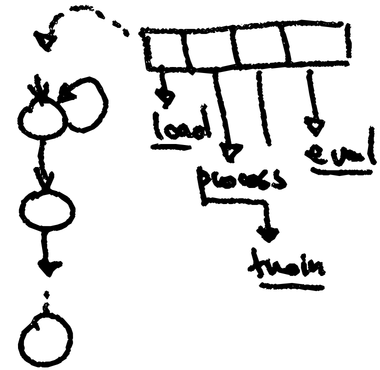
- stores code, actions hierarchy, * bias hierarchy, bias structure

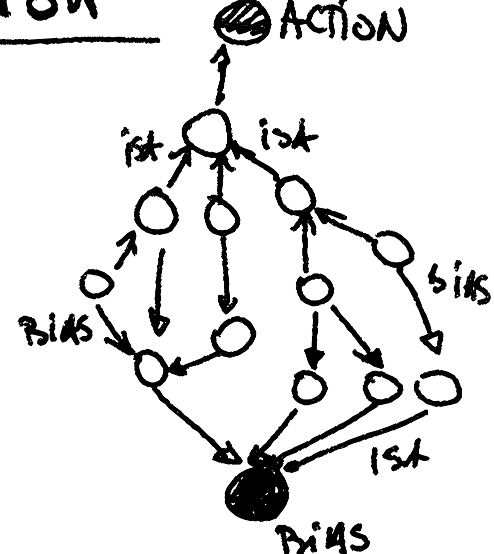
*bias = condition over an action/block/op

ARTIFICIAL INTELLIGENCE RESEARCH GROUP

Architecture & Motivation







program struct.

- contains prog. space
- can build code
- can execute code
- holds data/results
- depends on programming language/library

Automaton

- controls build flow
- communicates w. kb
- sends code/gets data from prog. structure

Kb

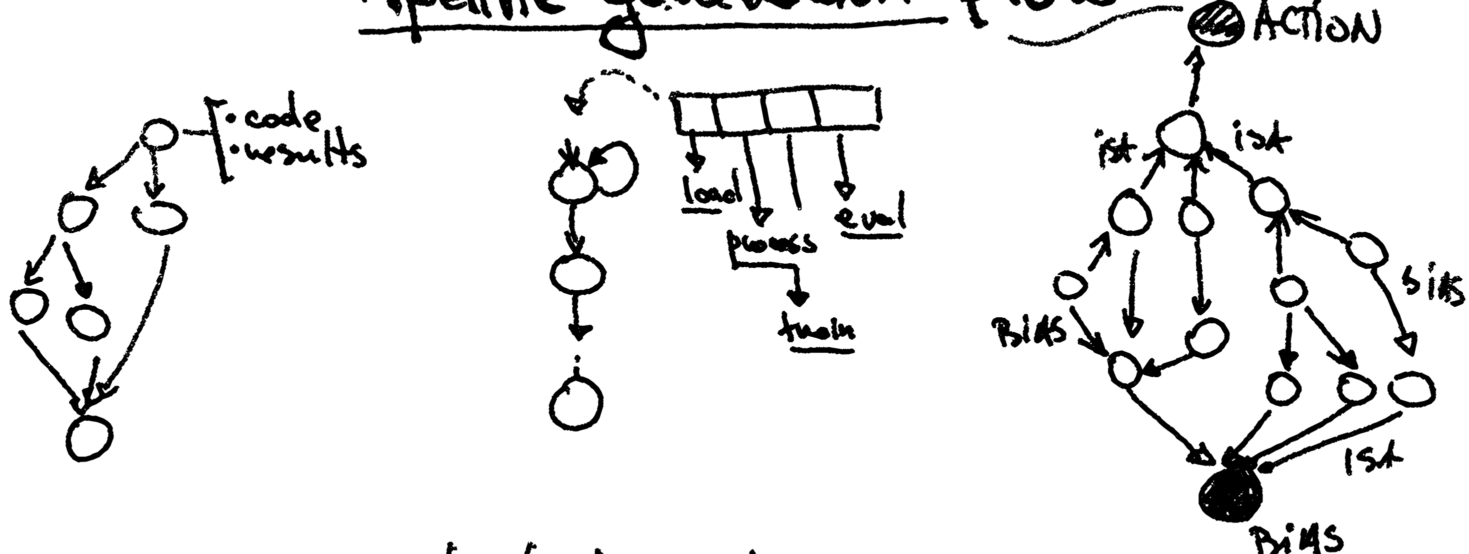
- stores code, actions hierarchy, * bias hierarchy, bias structure

*bias = condition over an action/block/op

ARTIFICIAL INTELLIGENCE RESEARCH GROUP

Pipeline generation flow

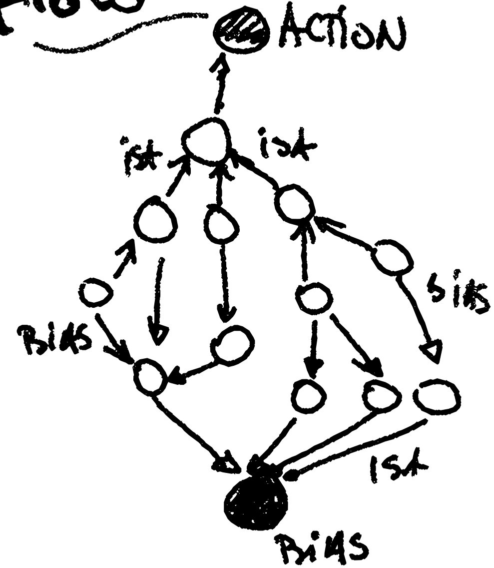
Pipeline generation flow



* Let's start from the beginning...

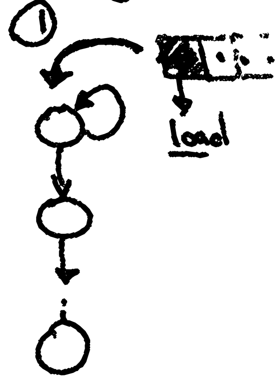
Pipeline generation flow

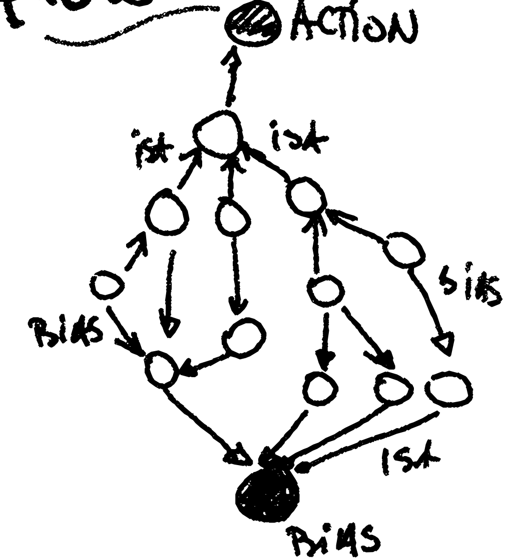
//no program.



Pipeline generation flow

//no program.

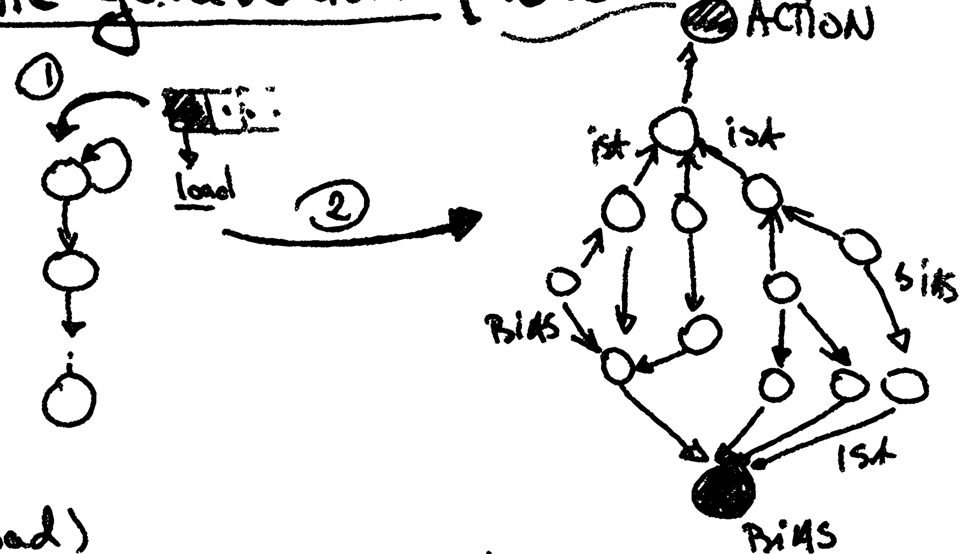




① Read first op (load)

Pipeline generation flow

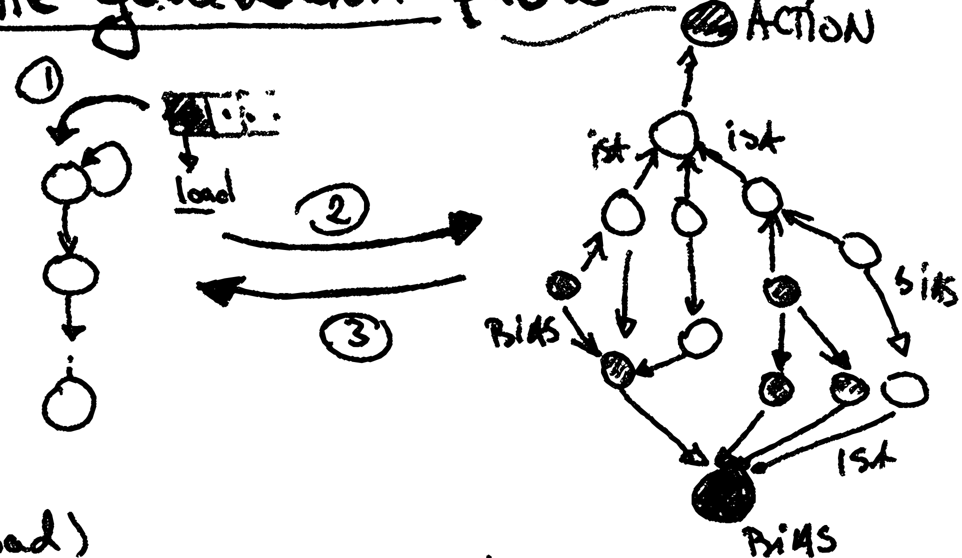
//no programm.



1. Read first op (load)
2. Query KB what ops + biases are available

Pipeline generation flow

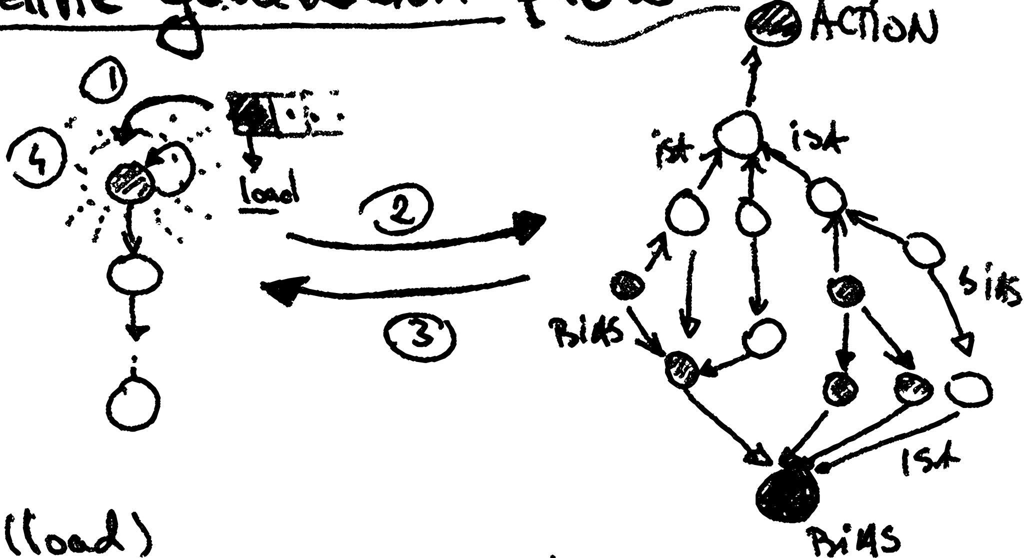
// no programm.



1. Read first op (load)
2. Query KB what ops + biases are available
3. Get list of (ops, biases)

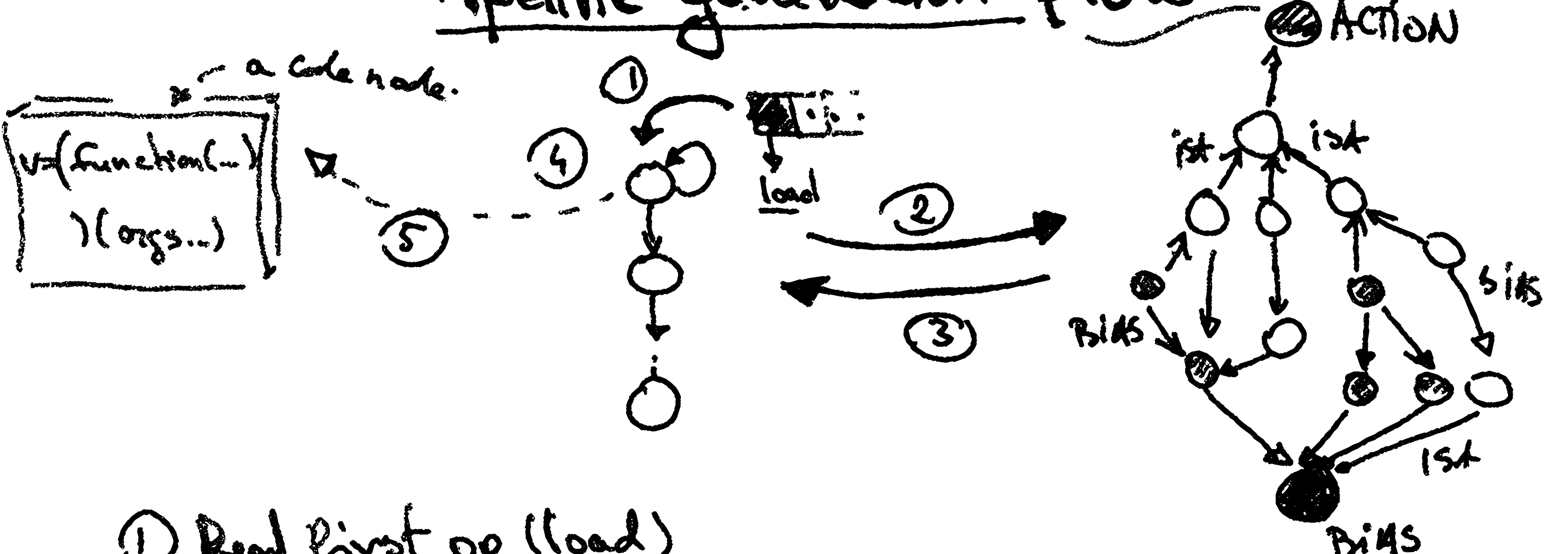
Pipeline generation flow

//no programm.



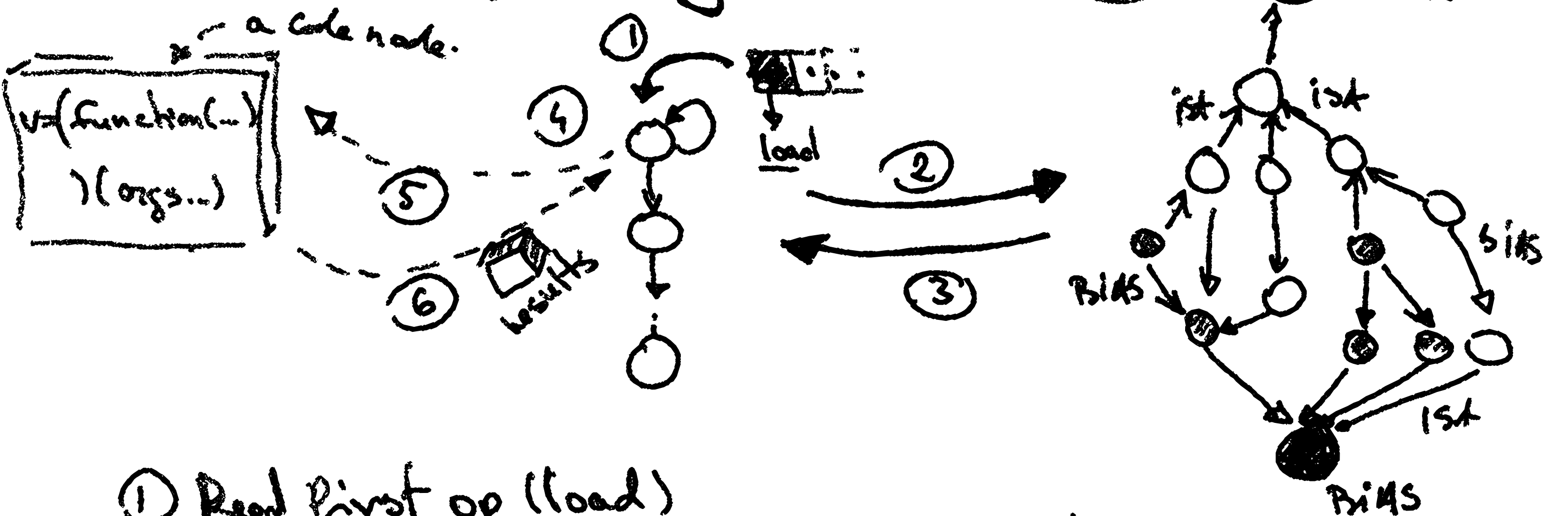
1. Read first op (load)
2. Query KB what ops + biases are available
3. Got list of (ops, biases)
4. Execute bias functions, retain feasible ops

Pipeline generation flow



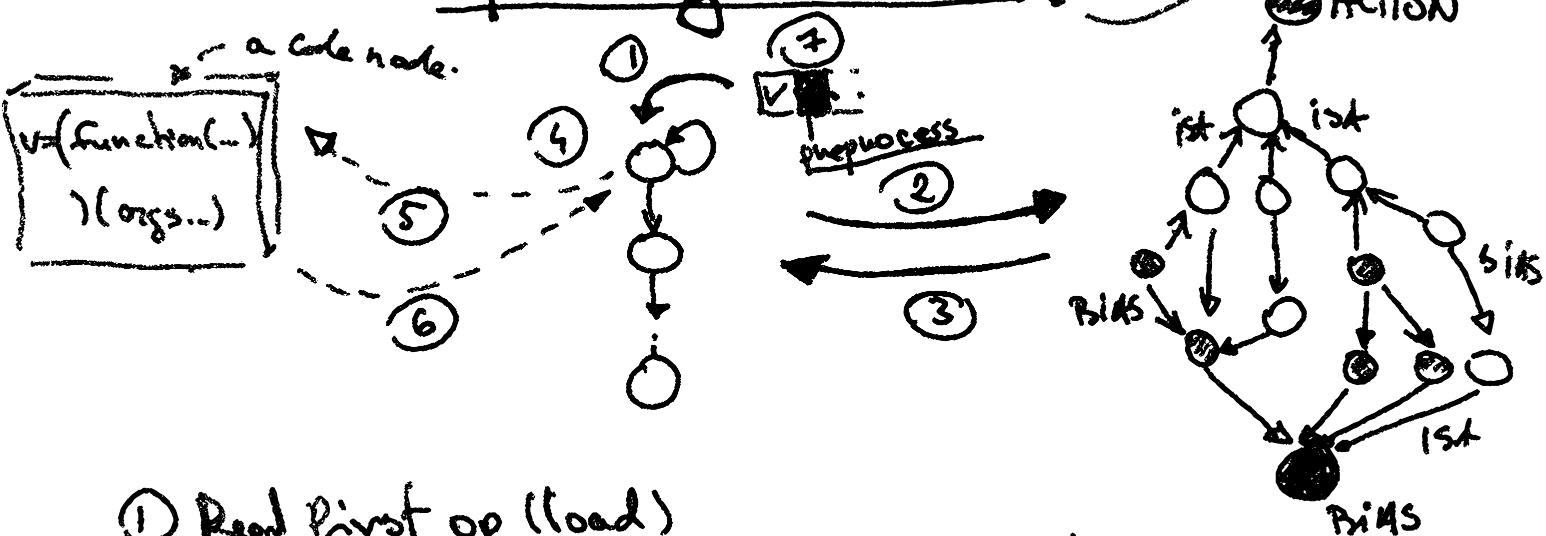
1. Read first op (load)
2. Query KB what ops + biases are available
3. Got list of (ops, biases)
4. Execute bias functions, retain feasible ops
5. Construct code

Pipeline generation flow



1. Read first op (load)
2. Query KB what ops + biases are available
3. Get list of (ops, biases)
4. Execute bias functions, retain feasible ops
5. Construct code 6. Execute pipeline

Pipeline generation flow



1. Read first op (load)
2. Query KB what ops + biases are available
3. Got list of (ops, biases)
4. Execute bias functions, retain feasible ops
5. Construct code
6. Execute pipeline
7. next op

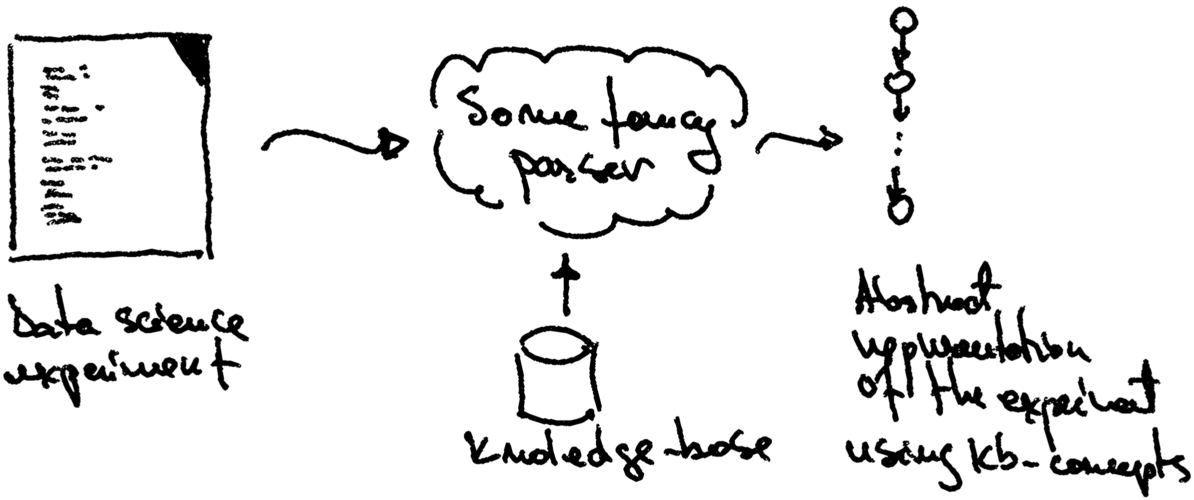
~ Open research challenges ~

~ Open research challenges ~

> Mining & sharing experiments

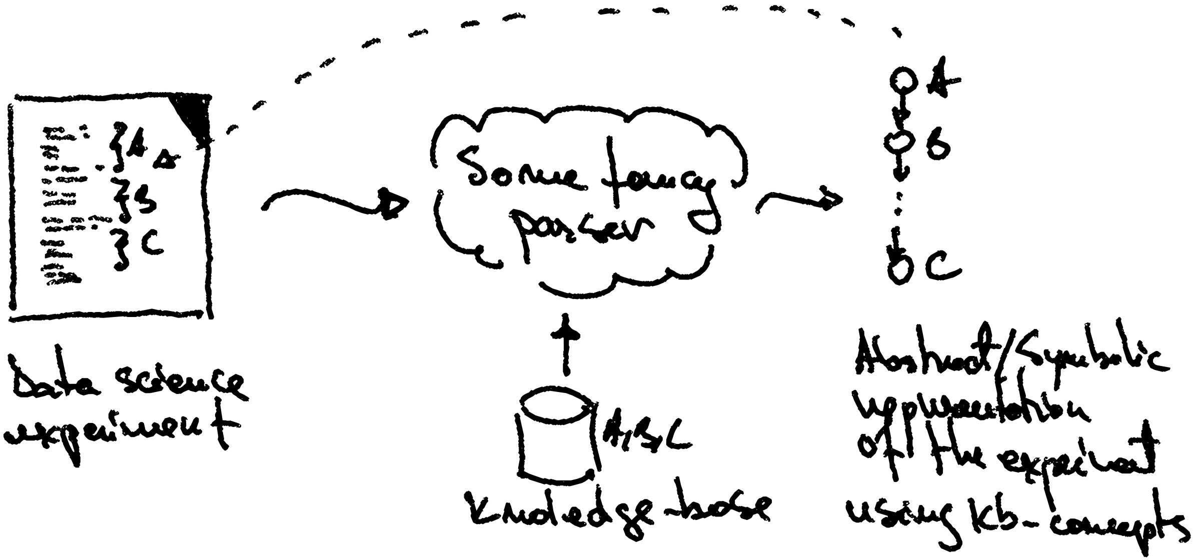
~ Open research challenges ~

> Mining & sharing experiments



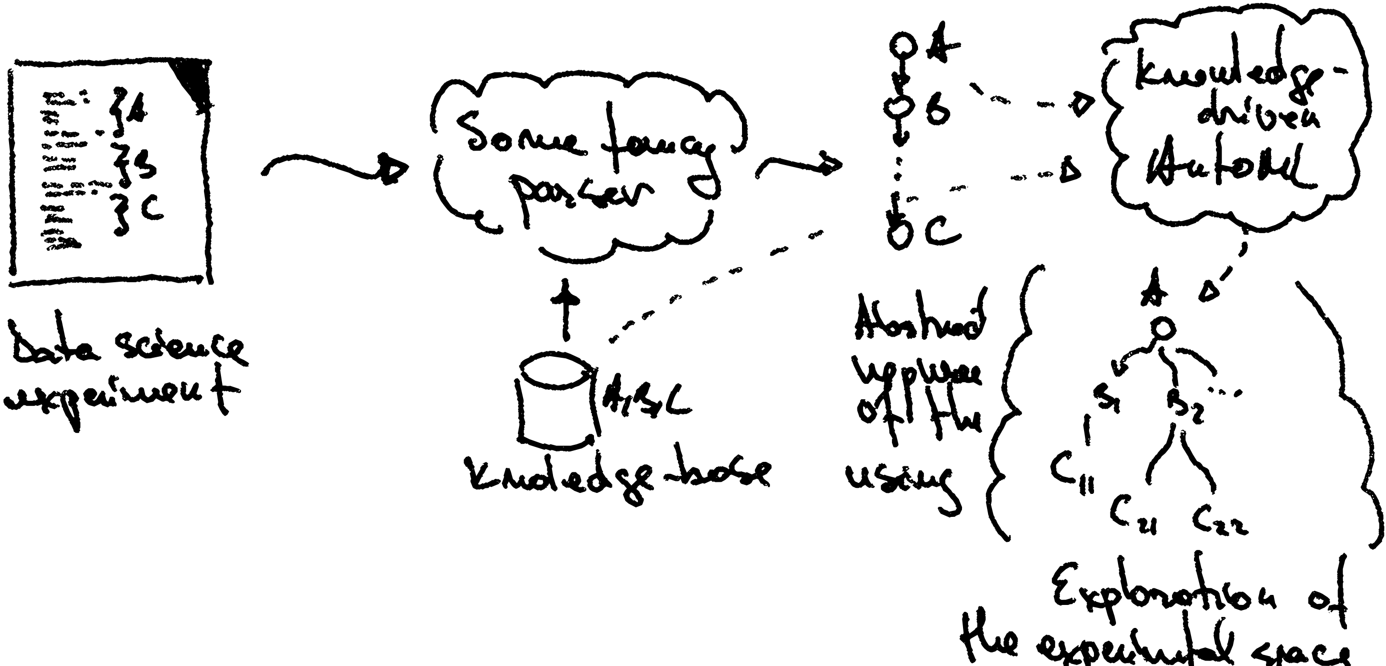
~ Open research challenges ~

> Mining & sharing experiments



~ Open research challenges ~

> Mining & sharing experiments



~ Open research challenges ~

~ Open research challenges ~

> wrapping ML libs

~ Open research challenges ~

> wrapping ML libs

> currently, each AutoML library depends on a <u>single</u> ML-library

$$
\begin{tikzcd}
\text{AutoML}_1 \arrow[d, "\text{DEPENDS}"] & & \text{AutoML}_2 \arrow[d, "\text{DEPENDS}"] & \dots & \text{AutoML}_n \arrow[d, "\text{DEPENDS}"] \\
\text{ML}_1 & \text{ML}_2 & \text{ML}_3 & \dots & \text{ML}_n
\end{tikzcd}
$$

> changes in ML (frequent) influence AutoML

~ Open research challenges ~

> wrapping ML libs

> what if a single AutoML library supports multiple ML libraries

AutoML$_1$

ML$_1$ ML$_2$ .. ML$_n$

~ Open research challenges ~

> wrapping ML libs

> what if a single AutoML library supports multiple ML libraries?

$$\begin{array}{ccc} & AutoML_1 & \\ \nearrow & \uparrow & \nwarrow \\ ML_1 & ML_2 \,.. & ML_n \end{array}$$

Dependency inversion!

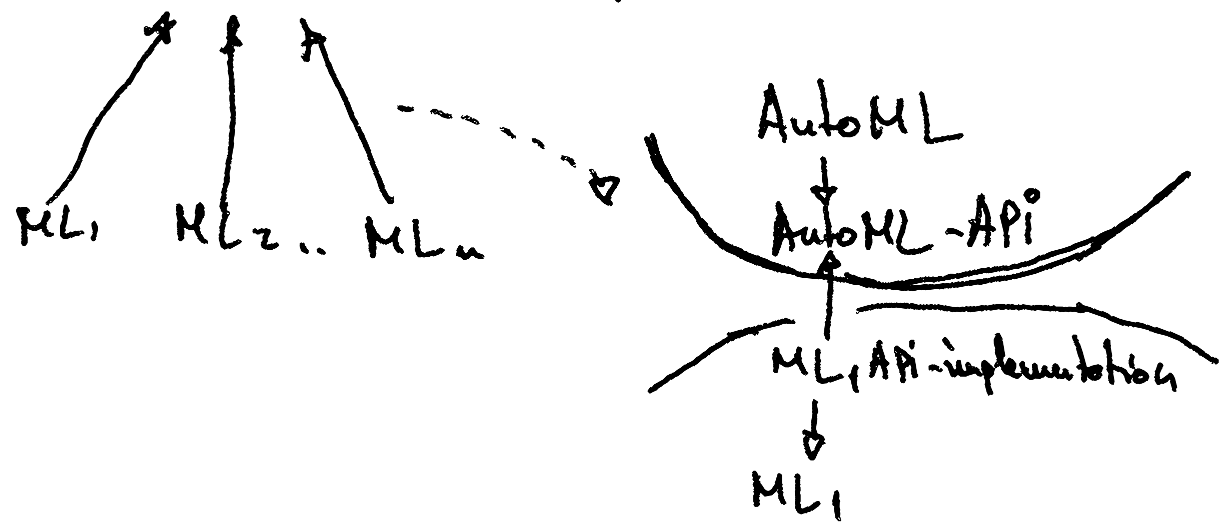
"Bob Martin – Clean Architecture"

~ Open research challenges ~

> wrapping ML libs

> what if a single AutoML library supports multiple ML libraries?

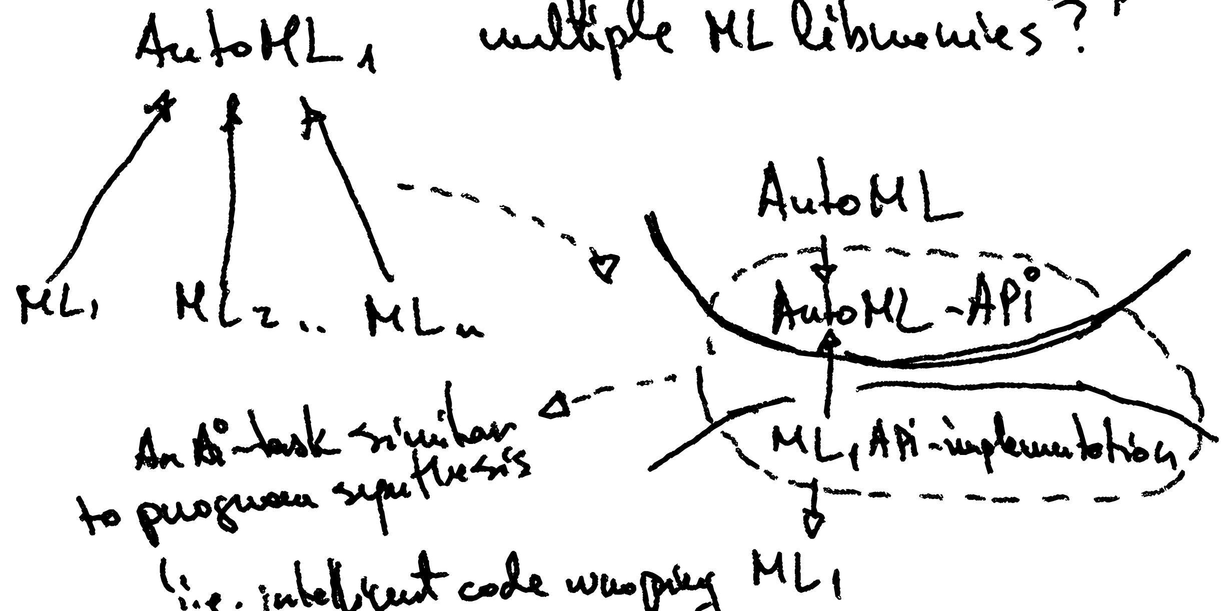
AutoML1



~ Open research challenges ~

> wrapping ML libs

> what if a single AutoML library supports multiple ML libraries?



Open research challenges

~ Open research challenges ~

> Optimization & Evaluation

~ Open research challenges ~

> Optimization & Evaluation

↓

Currently that's done using some form of metric involving accuracy & time/resources

~ Open research challenges ~

> Optimization & Evaluation

Currently that's done using some form of metric involving accuracy & time/resources

Interpretability?

Explainability?

Ease of use?

Scalability?

→ HOW TO BEST INTEGRATE THE QUALITATIVE?

Thank you.

Q & A Time ...