X Jornadas *Usuarios de R* · Murcia'18

rjazz, an R package to control **Jazz**, the lightweight data processing framework

Open Source software released by

BBVA Data & Analytics

1. The motive

- AI winter?
- The GofAI myth
- On rectangles
- Optimizing
- All together!

AI winter (maybe) coming

Does AI live in the perfect world businesses and mainstream media are telling us?

AI is living the most exciting moment in decades. All the necessary tools to explore, train and deploy deep learning models are free software. They combine highly optimized low level CPU and GPU software with highly expressive and productive scripting languages on top of it. This allows AI researchers to try new ideas and publish results every day.

Meanwhile, in the same planet ... Most important AI researchers, *we just name three to avoid digressing*, agree that we still need drastic changes before we can solve problems such as common sense reasoning and many more. Humans can solve these tasks even as kids.

> ***Geoffrey Hinton***: *"'Science progresses one funeral at a time.' The future depends on some graduate student who is deeply suspicious of everything I have said."*

> ***Yoshua Bengio***: *"Engineers and companies will continue to tune getting slight progress. That's not going to be nearly enough. We need some fairly drastic changes in the way that we are considering learning."*

> ***François Chollet***: *"For all the progress made, it seems like almost all important questions in AI remain unanswered. Many have not even been properly asked yet."*

Our only intention making this point

We obviously don't know how to solve common sense reasoning, our point is: **nobody does**, which is relevant as we continue explaining.

1.1 AI winter?

The "Good Old Fashioned AI" argument is partial at best

Here is how it goes

We don't cite anyone as its author. The argument just got repeated to the point it is **common belief**. It goes: *"Long time ago, we as humans coded the solutions to the problems, that's **GOFAI** (Good Old Fashioned AI), we figured that was a dead end and switched to end-to-end optimization. Now, data makes the difference, and the machine creates the model using **deep learning**."* Many even state that, today, major AI players can only be the data owners.

The least important thing is how this argument is unfair to all the researchers and achievements AI produced for over sixty years ~1950-2013 and that some of these achievements were truly game changing. We just all got used to taking them for granted. That's what AI has always been about: the **next problem**.

The important thing is that even in the fields in which **DL** (Deep Learning) produced major advances, as in **computer go** this vision is partial to the point of not being true. **AlphaGo**, the program that won against Lee Sedol, the best player in the world, is a mix of Monte-Carlo Tree Search and DL. Without the DL it is still stronger than almost the whole human population and as strong as the best professionals in 9x9. With only the DL and no search, it just sucks. It becomes a player that plays strong and elegant moves, but does not understand **why** or **when** they should be played. It is like a person trying to fake being a physicist by repeating random phrases taken from academic papers in physics.

Our point being: **deep learning** is great, but hard problems require more than just DL. You may be thinking: *"I can do these other things on top of current platforms easily."* Well, easily, maybe, but **not efficiently** as we are about to explain.

(Not really) a digression on rectangles

We immediately recognize rectangular shapes as man made.

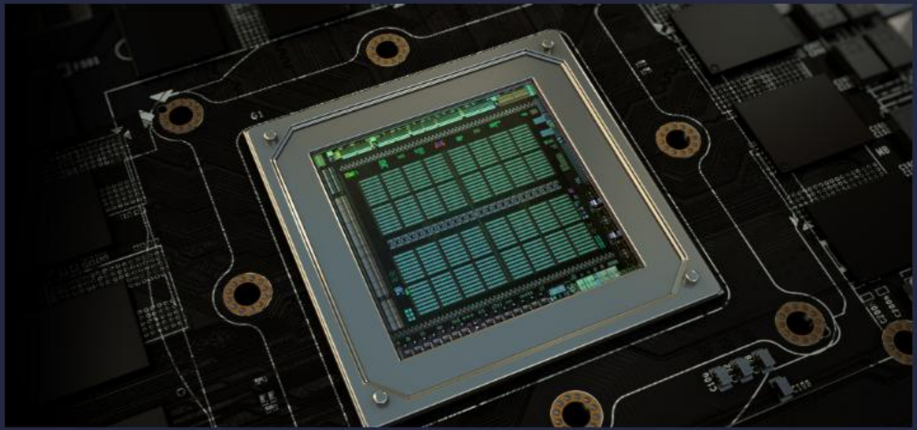
we abstract ideas as rectangles

- We build our data sets as rectangles. (like sheets and tables)
- Images and videos are rectangles. (in higher dimension)
- Neural networks are rectangles. (of units)
- Even words in **natural language processing** are rectangles!

> *Believe it or not, the latter is true. In the online version of the Cambridge Dictionary, the many definitions of the word **do** take 442 lines, the word **think** takes 143. With just 4 you can define **pneumonoultramicroscopicsilicovolcanoconiosis**. In deep learning, they all take the same: **one row from one rectangle**.*

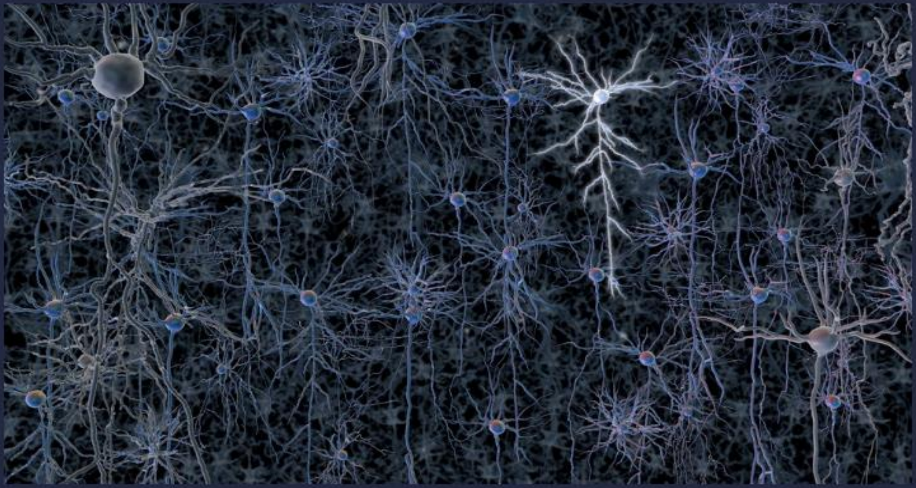
And that, of course, is very good, because **humans are very efficient pushing rectangles into other rectangles**.

We build rectangular buildings out of rectangular elements, store the energy from the rectangular panels in rectangular arrays of batteries that, again, are rectangular arrangements of rectangular cells. In AI, we push our data rectangles into rectangular GPUs where we compute rectangular arrays of artificial neural networks.



This beauty is a GPU. If you enlarge it, you can count 2048 cores.

> *You may be surprised to read that all the 2048 cores of the GPU not just execute the same program, but they do it exactly at the same time. When one of the cores, as a result of a conditional branch, takes a different logical path, all the other cores wait until it finishes. Cores can only execute the same instruction as the others or do nothing.*



And surprisingly, many writers seriously claim that artificial neural networks are biologically inspired. But **that** would be a digression.

1.3 On rectangles

So, how is the "perfect world" model for AI broken?

It is broken in different ways.

- Current platforms are only near optimal **in one direction**. If we add something like tree search, it can take 1000 times more in a high level language than it takes in a lower level one. Suddenly, the profiler shows the pattern exactly upside down.
- Even when we compute "just neural networks", its efficiency depends on us **pushing rectangles into other rectangles**. We stack fully connected layers without considering alternatives.
- Current platforms do not have efficient mechanisms to try ideas on **"just a few cases"**. You can, of course, build smaller data sets to explore ideas. But some hypotheses such as *"An elephant is heavier than an ant."* inherently require less cases to be verified than others like *"Martian gravity influences athletic records."*. This is not minor, learning is precisely about verifying hypotheses.

Converging to a solution vs. "just doing it"

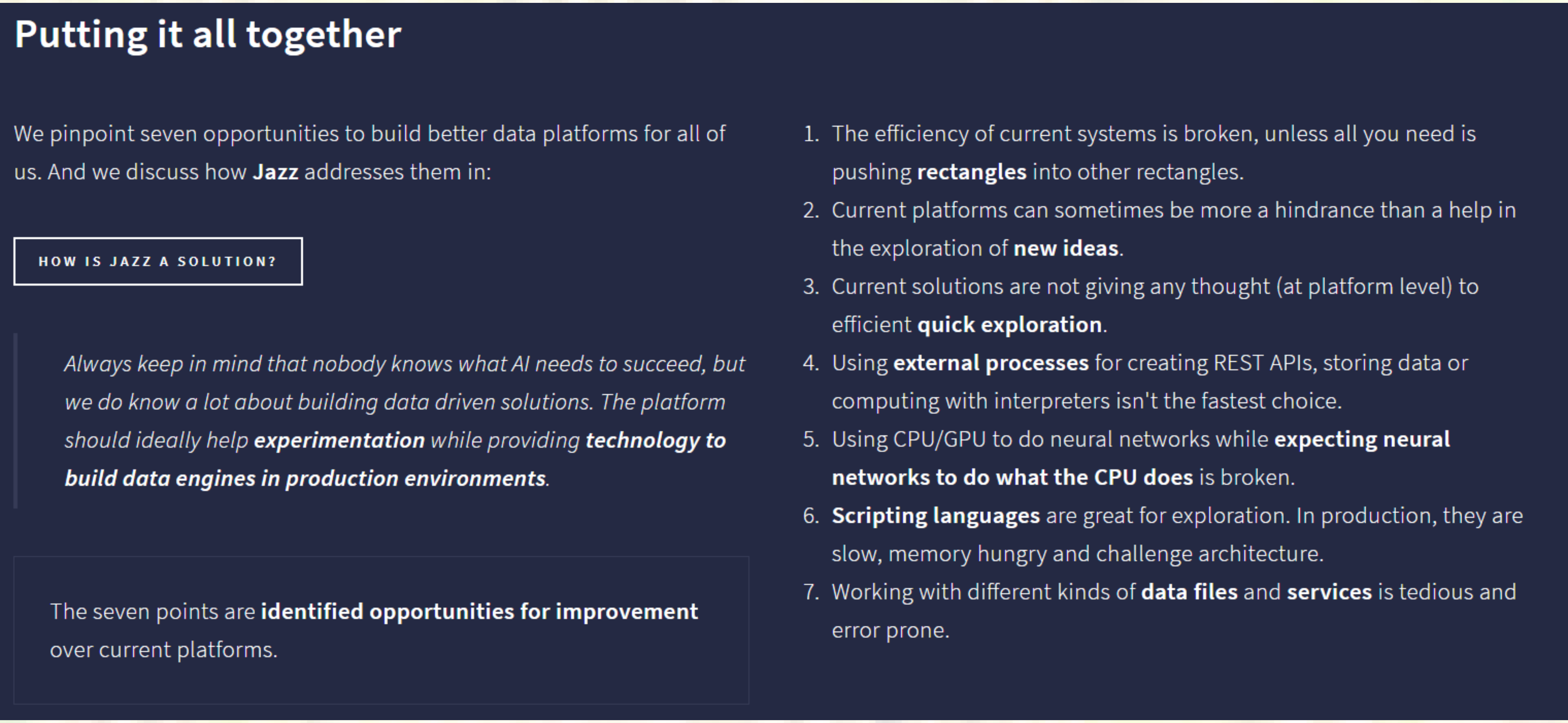
If you hear the phrase **"Turing complete"** applied to any system, it means the system can compute anything. There is a remarkable theoretical framework called the **"Church-Turing Thesis"** that states (simplifying) that all Turing complete systems are equivalent. This is a solid foundation for AI: if the brain can do it, a computer can, since both are Turing complete.

> *There is a lot of excitement about neural networks being Turing complete (except in trivial cases) since it should mean: **we don't need anything else** to do AI.*

Actually, we do, at least if we want it **before the sun burns out**.

But, when we need a multiplication, we can choose: A **CPU** doing hundreds of millions per core per second with 0% error in 20 digit precision or a **neural network** doing 100 per second after three days of training in 3 digit precision with a 2% error rate. (The former is just average performance of devices costing $10, the latter a generous estimation without even mentioning its cost.)

1.4 Optimizing



Putting it all together

We pinpoint seven opportunities to build better data platforms for all of us. And we discuss how **Jazz** addresses them in:

HOW IS JAZZ A SOLUTION?

> *Always keep in mind that nobody knows what AI needs to succeed, but we do know a lot about building data driven solutions. The platform should ideally help* ***experimentation*** *while providing* ***technology to build data engines in production environments****.*

The seven points are **identified opportunities for improvement** over current platforms.

1. The efficiency of current systems is broken, unless all you need is pushing **rectangles** into other rectangles.
2. Current platforms can sometimes be more a hindrance than a help in the exploration of **new ideas**.
3. Current solutions are not giving any thought (at platform level) to efficient **quick exploration**.
4. Using **external processes** for creating REST APIs, storing data or computing with interpreters isn't the fastest choice.
5. Using CPU/GPU to do neural networks while **expecting neural networks to do what the CPU does** is broken.
6. **Scripting languages** are great for exploration. In production, they are slow, memory hungry and challenge architecture.
7. Working with different kinds of **data files** and **services** is tedious and error prone.

1.5 All together!

2. AI technology the Jazz way

- Everything is a block!
- What makes it special?
- Python, R, http

In Jazz, everything is a block!

Blocks are (at the simplest)

- **Tensors**(*) with attributes
- **Files** with an instruction manual (**)
- **Web resources** including (urls, mime types, etc.) (**)

(*) A tensor is a vector of vectors of vectors … of **one type**.
(**) All are the same: **tensors with attributes**.

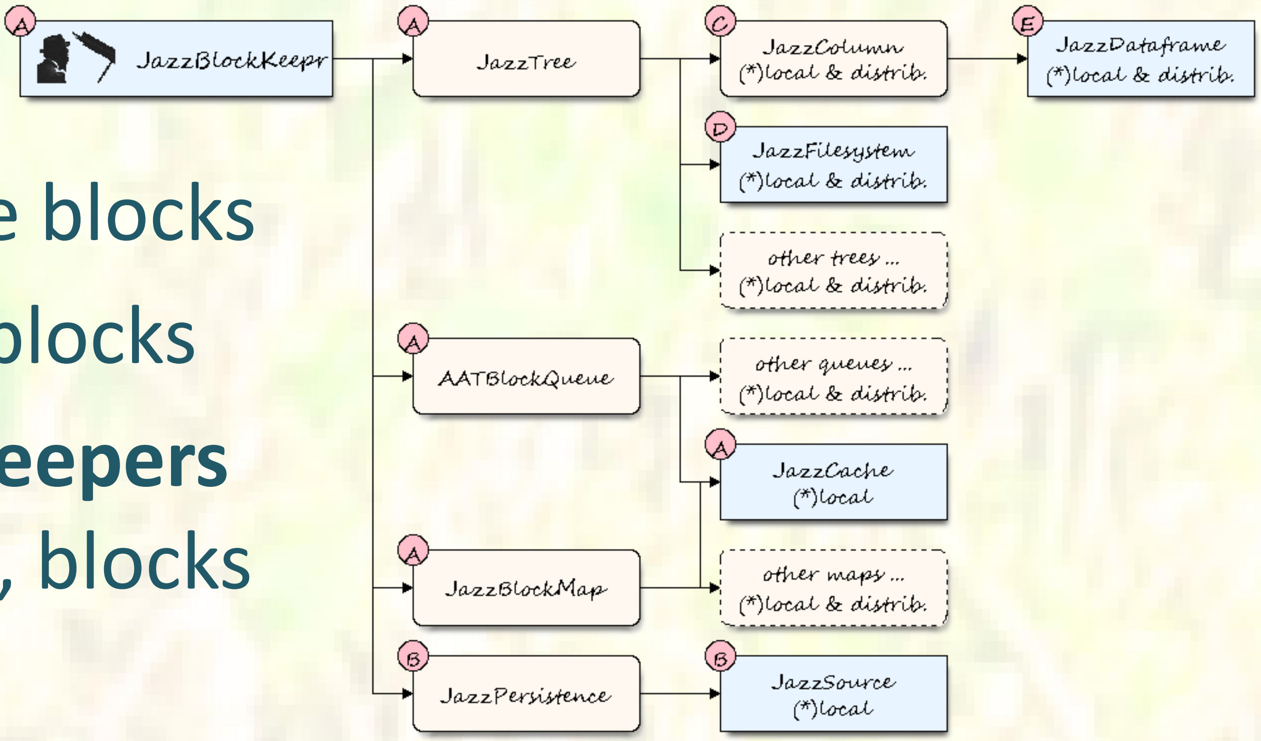
In Jazz, everything is a block!

- **Blocks** are owned by **keepers**(*).
- **Bebop**(**) **functions** do operations on **blocks**.
- Bebop functions are **methods of classes**.

(*) Efficient C++ structures: queues, pipes, trees, priority queues, maps, memory mapped files, etc.

(**) **Bebop** is a thread safe, automatically paralleled language with support for **automatic programming**.

In Jazz, everything is a block!


A JazzBlockKeepr
A JazzTree
C JazzColumn (*)local & distrib.
E JazzDataframe (*)local & distrib.
D JazzFilesystem (*)local & distrib.
other trees ... (*)local & distrib.
A AATBlockQueue
other queues ... (*)local & distrib.
A JazzCache (*)local
A JazzBlockMap
other maps ... (*)local & distrib.
B JazzPersistence
B JazzSource (*)local

- **Functions** are blocks
- **Keepers** are blocks
- **Classes are keepers** and, therefore, blocks

Okay, you are describing a data processing framework, what makes it special?

What makes it special?

1. Blocks are **volatile**, **persisted**, **distributed** or **ubiquitous** as defined. Abstraction from **extremely small** to **extremely large** is automatic. The same function works on a small vector or a distributed data-frame.

What makes it special?

2. Extremely efficient persistence (a memory mapped file) in the same process allows **sampling without copying**. (No need of a DB.)

3. Extremely efficient **REST API** in the same process **with a thread pool**. (No need of an http server.)

What makes it special?

4. A full language that is close to perfect performance (assuming enough tensor size), thread safe and auto distributing instead of just **some optimized data mangling primitives**. (No need of an interpreter.)

What makes it special?

5. Distributed data-frames where cells are tensors (E.g., video or sound tracks, not links to files)

6. An **abstraction over the file system** that allows reading/writing binary and compressed files on demand.

What makes it special?

7. **Tree Search** as a first class citizen

8. **Reinforcement learning** “out of the box”

9. Inference on **small samples** done efficiently

10. Support for **automatic programming** at bytecode level

What makes it special?

11. Deep Learning as a first class citizen

Of course!

It's just **not our only** (or even our first) **priority**.

Jazz can be as efficient doing **just DL** as a specialized platform.

I am a Python guy, an R lady, a front end developer ...

- install.packages('rjazz')
- pip install pyjazz
- docker run kaalam/jazz_neat.latest

3. Jazz today

- Strengths and weaknesses
- The most important thing in OSS

Now, we are on our own, with our strengths …

- We don't just **shoot at the moon**, we shoot at a moon in a distant galaxy.
- We have **decades of experience** doing AI and ideas about what could work.
- **We deliver**.
- We stand **on the shoulders of giants**. **OSS**
- We **enjoy** doing it.

… and weaknesses

- We still have to finish the MVP.
- We have to **create a community**.
- We need success stories.
- Short version: We need help.

And remember, the product is
the second most important thing
in any Open Source Software
Project

3.2 The most important thing in OSS

The most important thing in any Open Source Software Project is, of course …

3.2 The most important thing in OSS

The community

3.2 The most important thing in OSS

We need volunteers!

3.2 The most important thing in OSS

X Jornadas Usuarios de R · Murcia'18

Thank you!

kaalam.ai

Release only: **https://github.com/bbvadata/Jazz**
Development: **https://github.com/kaalam/Jazz**
Programming doc: **https://kaalam.github.io/develop**
Jazz reference: **https://kaalam.github.io/jazz_reference**

@kaalam_ai