The Profession of Solving

(the Wrong Problem)

a talk on failures: natural, artificial and stupid

Let's start with a true story.

It's about my first failure in data.

It actually happened.

I learned a lot from it.

But it is super embarrassing.

It's the story of my first A+ I ever got in college.

I was young ... and naive.

It was a first year course in statistics when my professor asked me to find data "out in the real world" and **ApplyStatistics[tm]**.

I was young ... and naive.

It was a first year course in statistics when my professor asked me to find data "out in the real world" and **ApplyStatistics[tm]**.

I had a gig doing the bar at a local theatre that just celebrated its 12 year anniversary. They were considering to expand but they weren't sure.

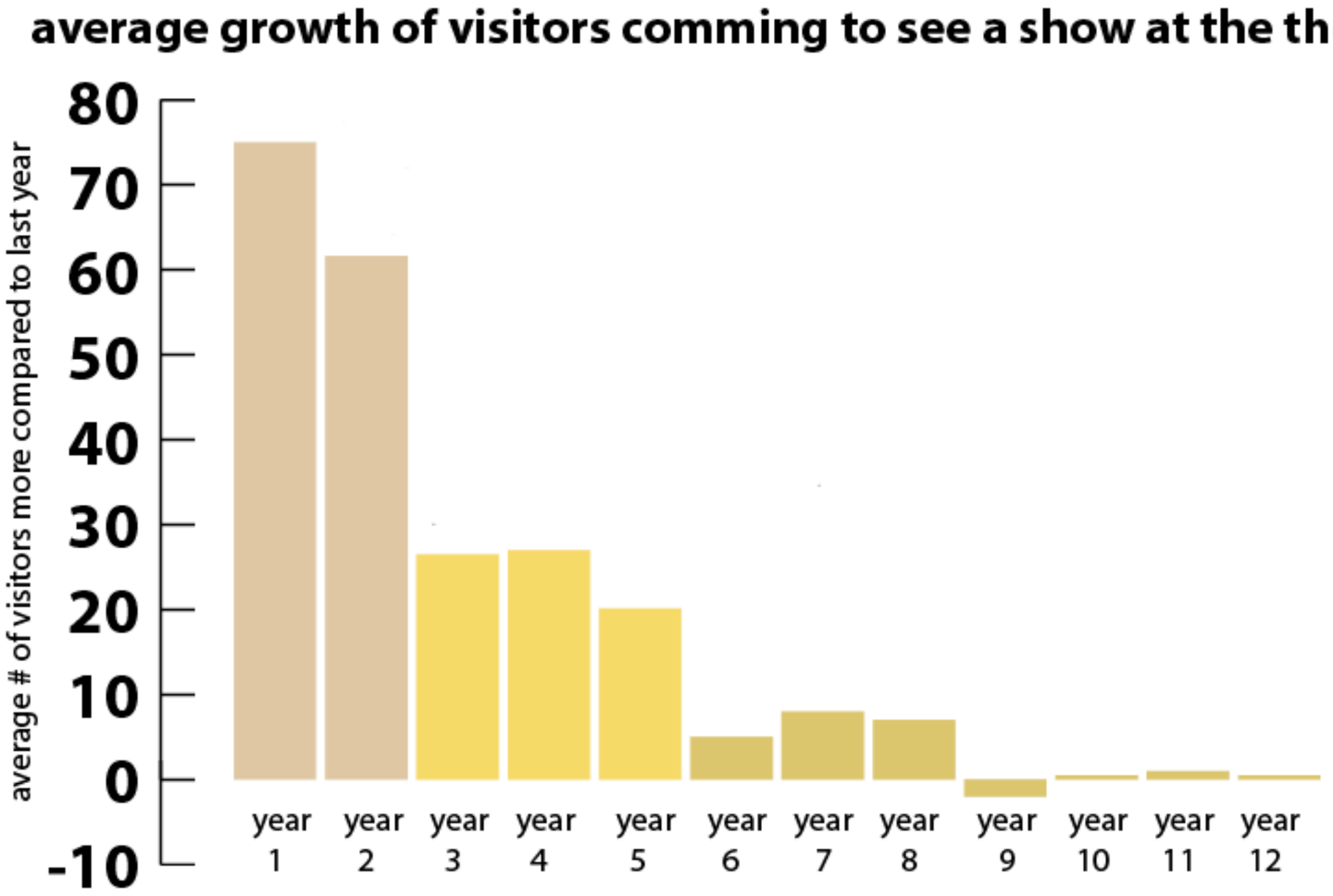
I was young ... and naive.

It was a first year course in statistics when my professor asked me to find data "out in the real world" and **ApplyStatistics[tm]**.

I had a gig doing the bar at a local theatre that just celebrated its 12 year anniversary. They were considering to expand but they weren't sure.

So I got data and got asked to **ApplyStatistics[tm]**.

This is a plot made by hand (couldn't code back then).



Results

I put a linear model through these data points and it turned out to be **significant**. I had statistical evidence that the theatre should not expand because the growth rate had stagnated.

Results

I put a linear model through these data points and it turned out to be **significant**. I had statistical evidence that the theatre should not expand because the growth rate had stagnated.

My teacher loved the report and gave me an 10/10.

Results

I put a linear model through these data points and it turned out to be **significant**. I had statistical evidence that the theatre should not expand because the growth rate had stagnated.

My teacher loved the report and gave me an 10/10.

The future was lookin' good ...

Then, the future happened.

Two days later I was working the bar at the theater and the house was packed. The the week after that, same story and the week after that was no different.

Then, the future happened.

Two days later I was working the bar at the theater and the house was packed. The the week after that, same story and the week after that was no different.

It hit me ...

You cannot sell more tickets if the every chair in the theatre is taken. If this is why the growth stagnates, expanding is a great idea.

Natural Stupidity > Artificial Intelligence

I was so focussed on the method that I completely neglected the fact that I was chasing a vanity task.

"I solved the wrong problem."

— me and a lot of people, a lot of times

Solving problems is *hard* but lately I've noticed that algorithms distract us of what we're supposed to do. There's a lot of hype suggesting **the Algorithm[tm]** is the solution. In this talk I hope to convince you that it is more all the stuff around the algorithm that allows us to work on the problem at hand.

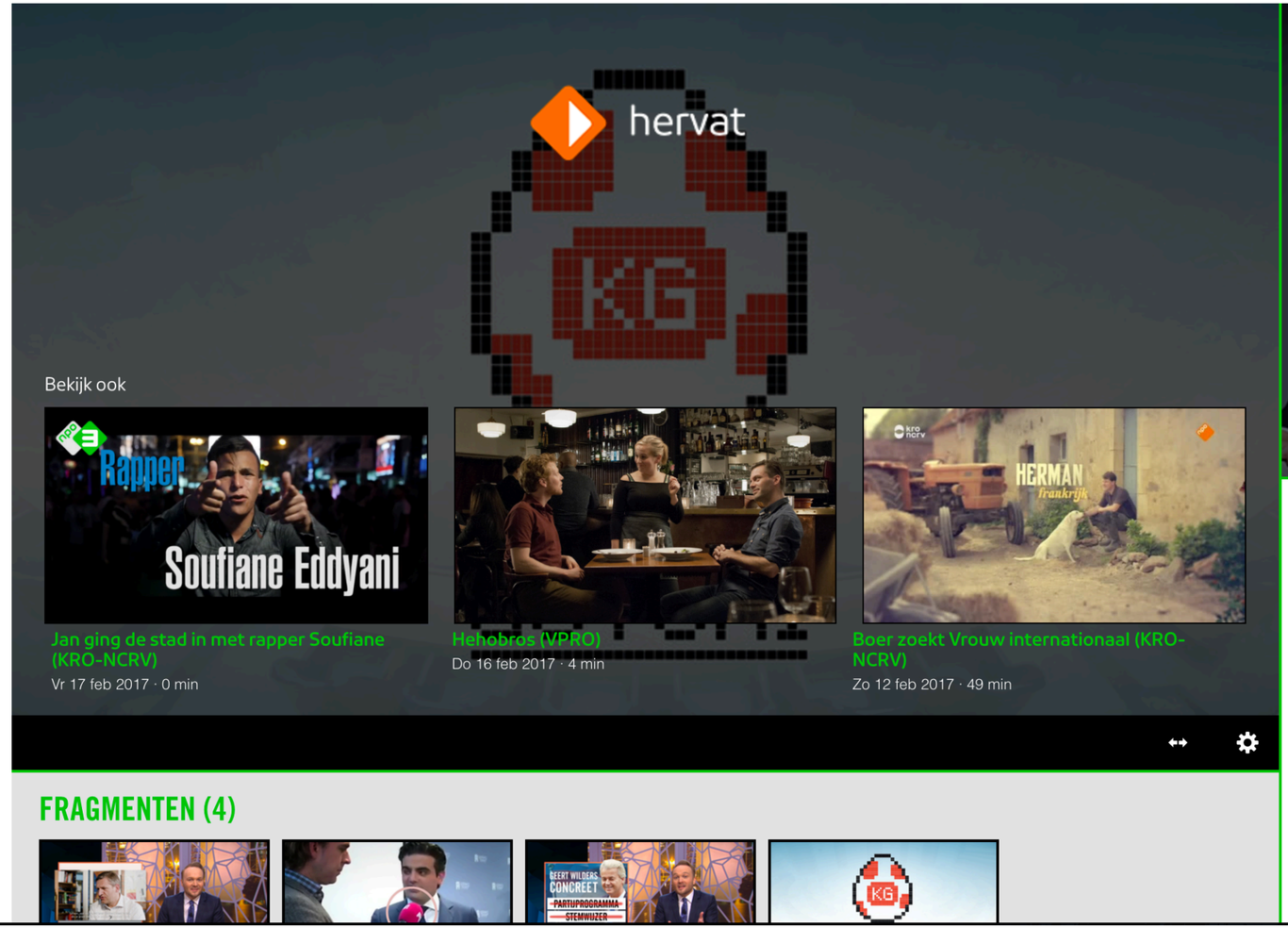
— story of a recommender

— story of rephrasing and hunger

— story of a kaggle hack

— story of a timeseries

ZONDAG MET LUBACH FRAGMENT



VANDAAG OP 3

19:25	QUICKEST QUIZ
19:50	NEDERLAND VERBOUWT PREVIEW
20:30	PROEFKONIJNEN PREVIEW
21:30	LAUREN! EN DE LIEFDE PREVIEW
22:10	BRAINWASHING STACEY PREVIEW
22:56	POLITIEKE PARTIJEN

Facts and Paraphrasing

— we had our first 2 months of logs
— they had a rule based system
— we had to improve their system
    — + 25% clicks

Facts and Paraphrasing

— we had our first 2 months of logs
— they had a rule based system
— we had to improve their system
    — + 25% clicks
    — + 25% people fully viewing recommendation

Facts and Paraphrasing

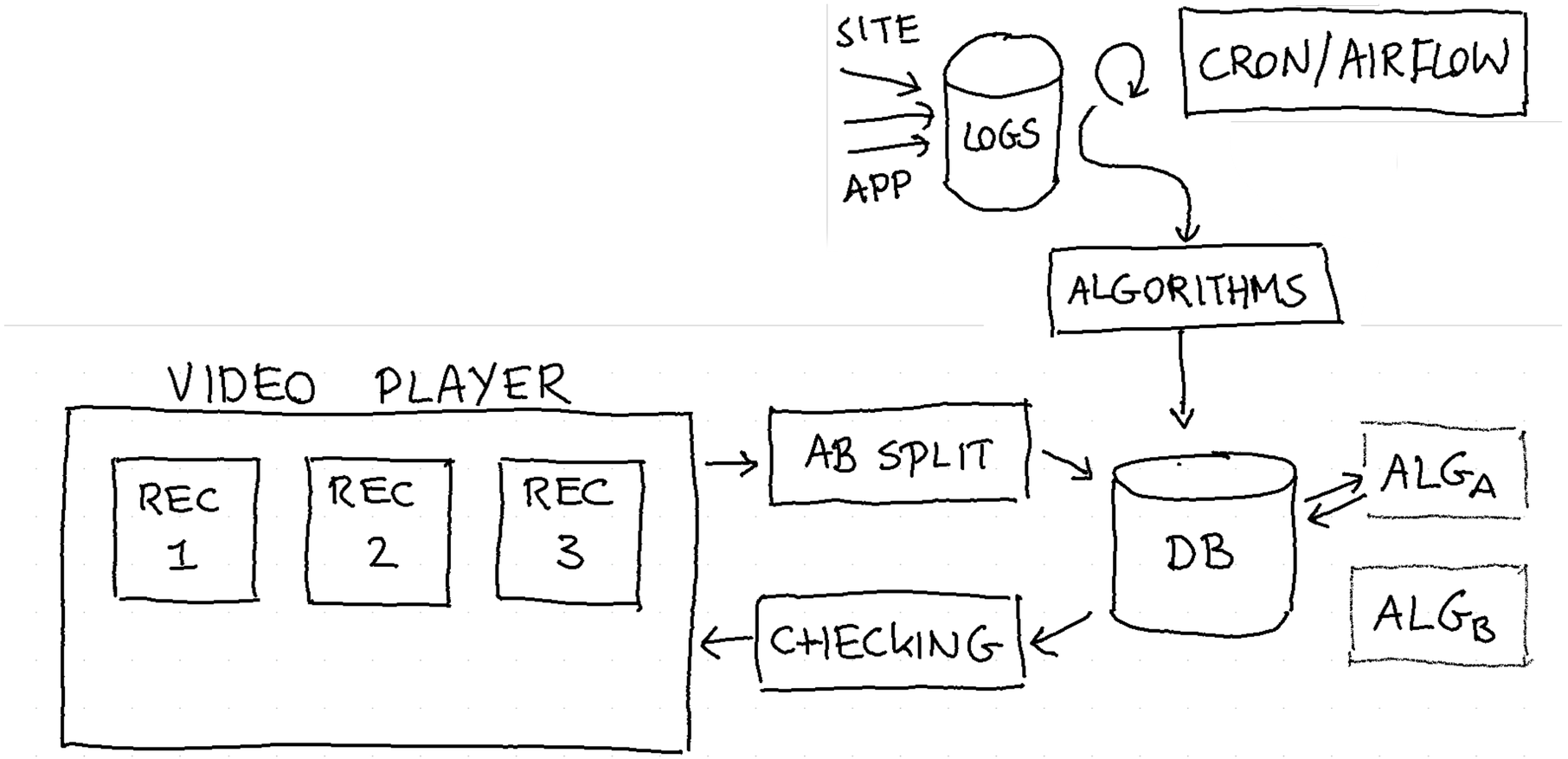
— we had our first 2 months of logs
— they had a rule based system
— we had to improve their system
    — + 25% clicks
    — + 25% people fully viewing recommendation
— people were suggesting **DeepLearning[tm]**

Facts and Paraphrasing

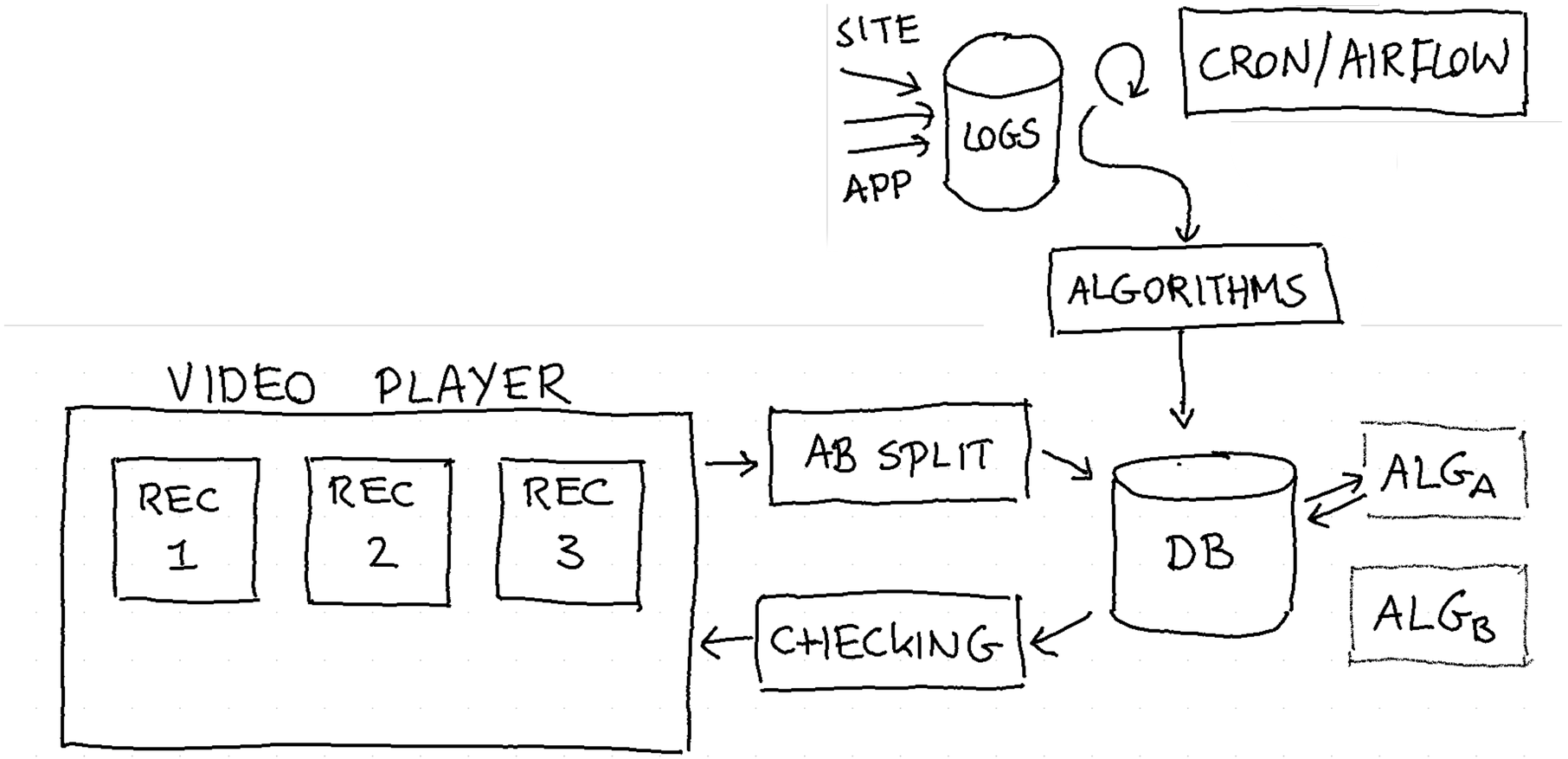
— we had our first 2 months of logs
— they had a rule based system
— we had to improve their system
    — + 25% clicks
    — + 25% people fully viewing recommendation
— people were suggesting **DeepLearning[tm]**

What is the first thing we did?

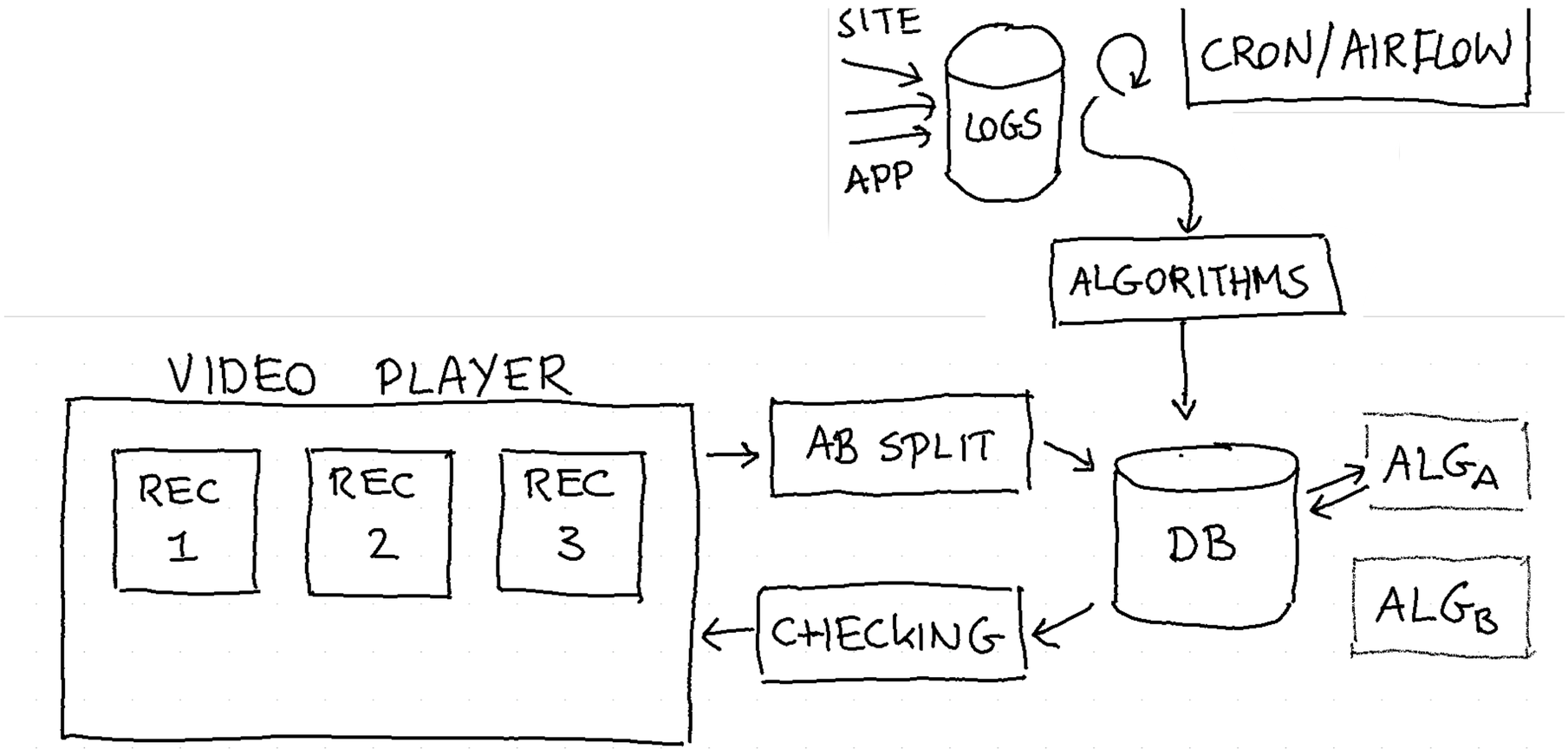
Step 1: DrawPictures not DeepLearning



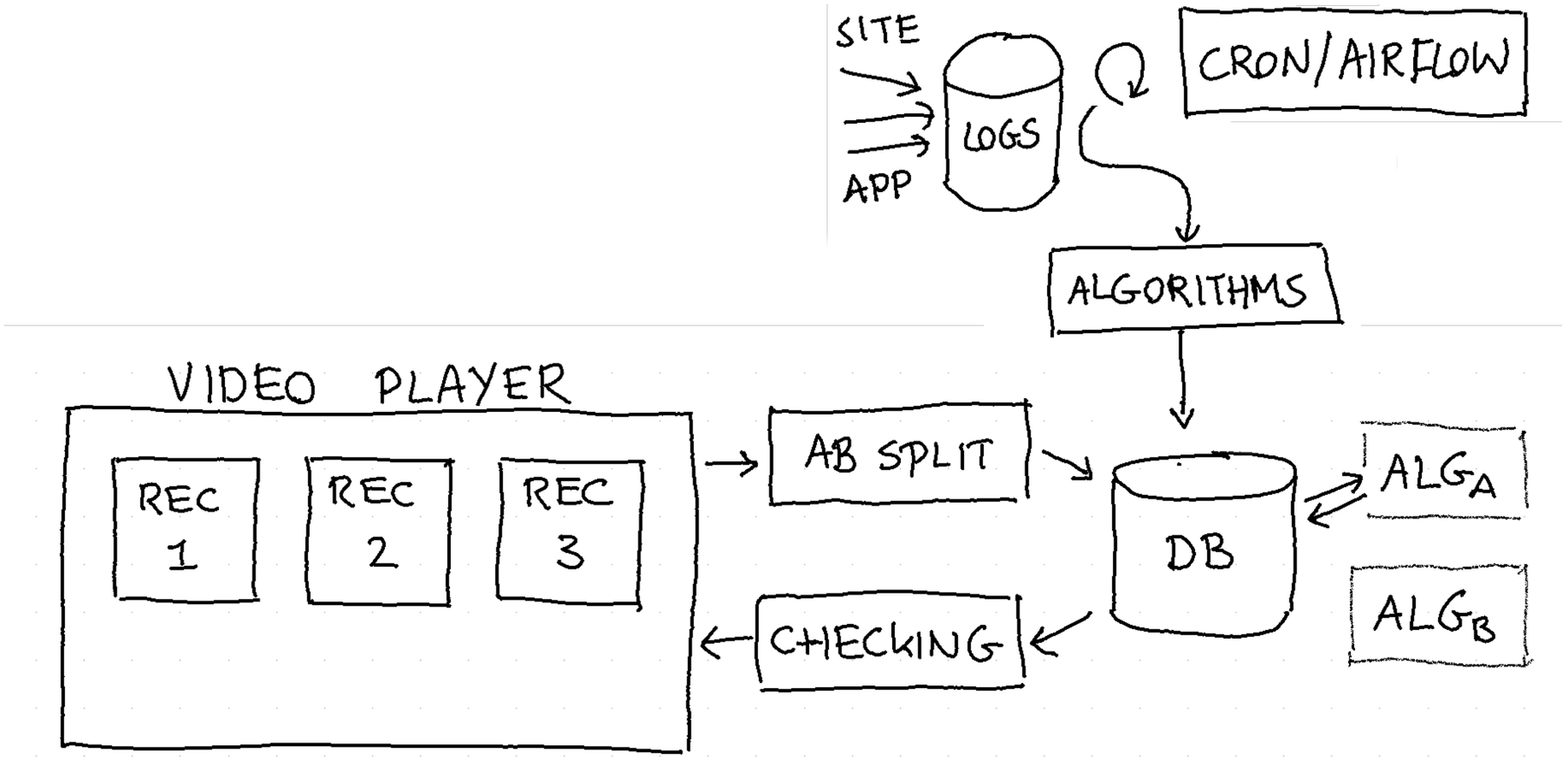
Step 2: A/A test this setup!



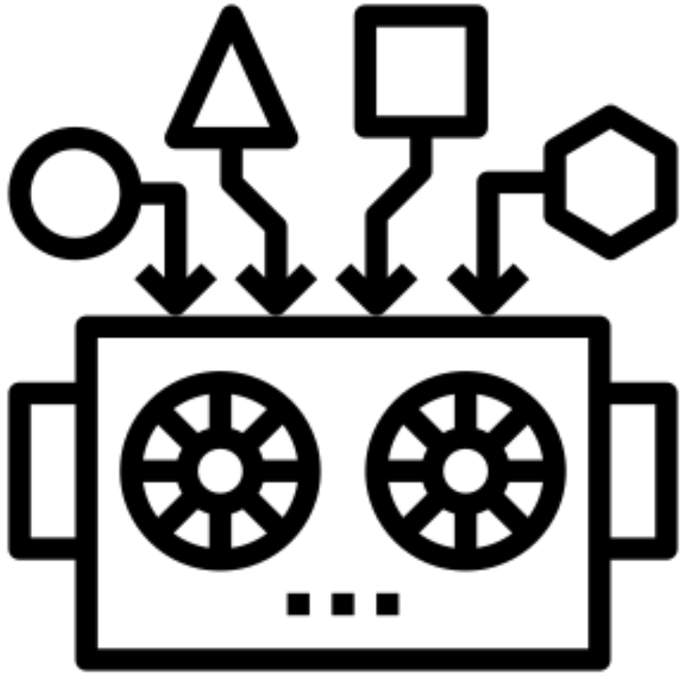
Step 3: $\text{alg}(B) = \text{entropy}$



Step 4: More Measuring: the order!



Step 5



Anybody guess? Is it finally **DeepLearning[tm]**?

Step 5: not DeepLearning, but DumbThinking

We started recommending the next episode.

"... oh."
— the DeepLearning[tm] Advocates

"... oh."
— the DeepLearning[tm] Advocates

People dreamt about the solution. Not the problem.

Let us remember

Was the real issue in 2016
the fact that we weren't using **ApacheSpark[tm]**?

Was the real issue in 2018
the fact that we're not using **DeepLearning[tm]**?

Let us remember

Was the real issue in 2016
the fact that we weren't using **ApacheSpark[tm]**?

Was the real issue in 2018
the fact that we're not using **DeepLearning[tm]**?

Was the real issue in 2019 going to be
the fact that we're not using **AutoML**?

Let us remember

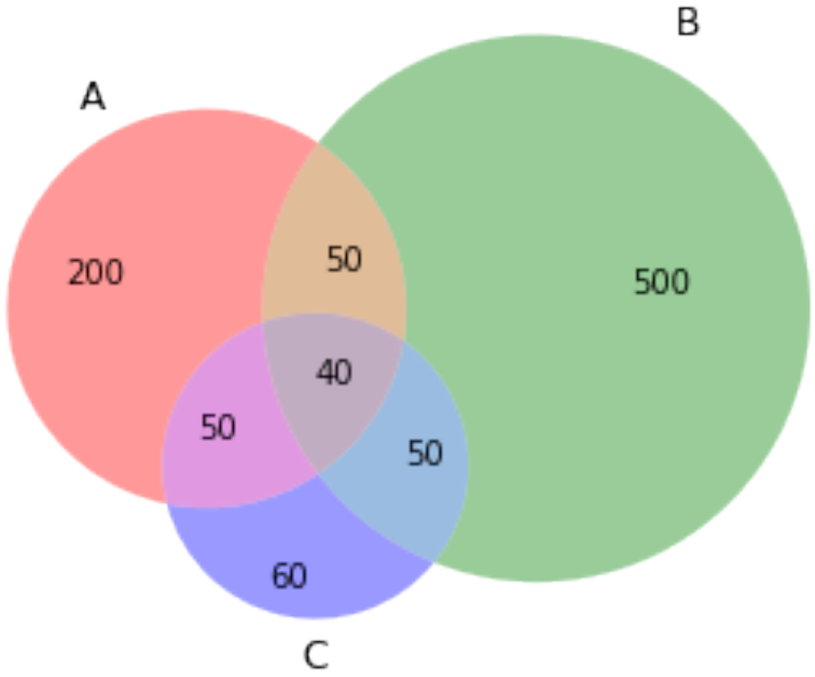
Or is the problem that we don't understand it?

— Do you talk with the end-users?
— Do you regularily drink coffee with the working folks down the factory floor?
— Can you explain on a qualitative level that your solution needs to do?

This led me to my full solution ...

Vincent's Lemma: Serendipity

I started thinking about the problem ...



Vincent's Lemma: Serendipity

$$R_{j \to i} = \frac{p(s_i | s_j)}{p(s_i | s_j^c)} \approx \frac{p(s_i | s_j)}{p(s_i)}$$

Once I calculate all "serendipity scores" $\forall i$ then I merely need to sort the scores and recommend the top k series.

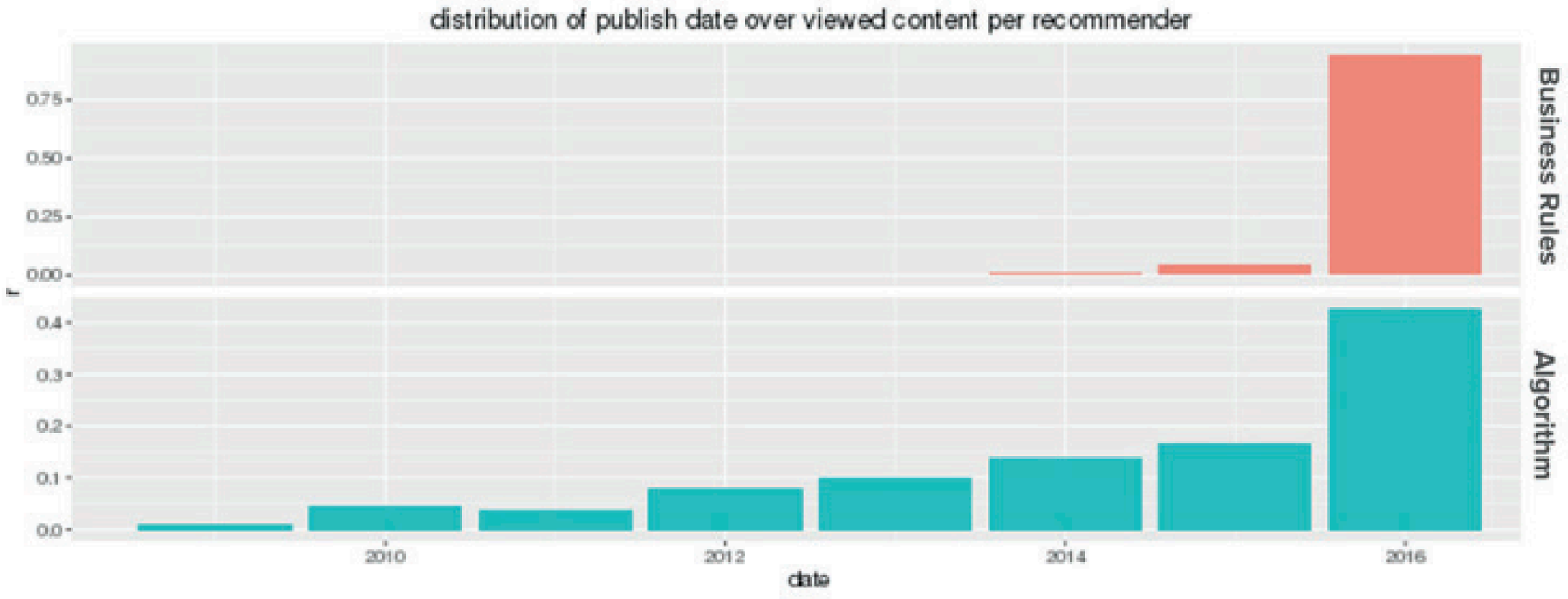
Note that $R_{j \to i} \neq R_{i \to j}$ which is what you want.

Final Algorithm

This algorithm is still live today. When we launched it (together with first episode) it caused a +X% on the entire platform and more than met our target.

We also were showing items that people were watching. No clickbait.

This wasn't why they were super happy though.


distribution of publish date over viewed content per recommender
0.75
0.50
0.25
0.00
0.4
0.3
0.2
0.1
0.0
r
Business Rules
Algorithm
2010
2012
2014
2016
date

Gotta Love Heuristics: Lessons

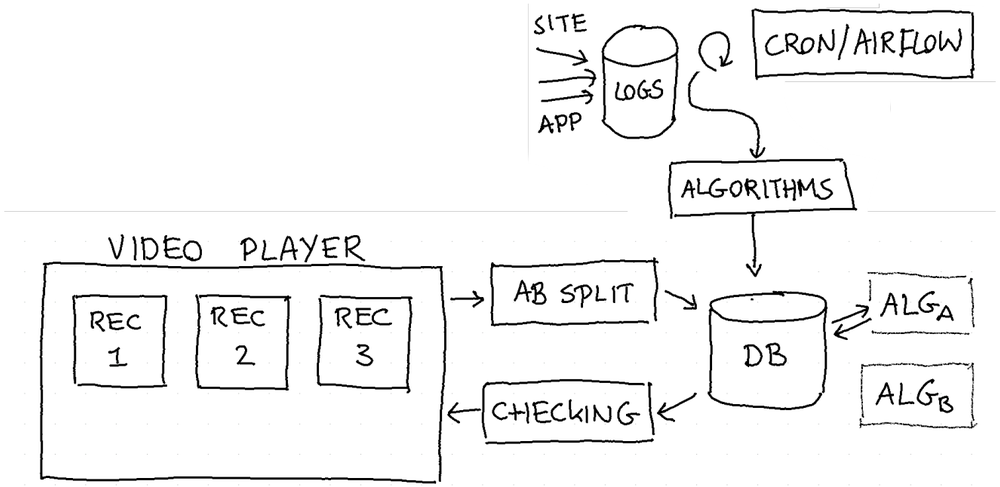
— two for loops and an if statement can beat a neural network easily if you've got a great heuristic around

— taking a step back and wondering "what is the actual problem" helped us get that

— there is no shame in spending time to understand the problem, this is also progress towards a solution

— blindly following a book/blog on recommenders would not have given us this approach

Me Gusta System



This system can change if need be, we can replace parts. A system that does not have this property replaces the old problem with new ones in due time.

Other Times

There is another approach to solving a problem that might offer solutions: try to force a rephrase!

I'll introduce a silly mathematical problem to show the power of rephrasing and next I'll mention how this also led to an improvement of a recommendation.

261 days: a billboard lottery

The Billboard Exercise

You need to decide the optimal number of letters to send to a contest. You'll loose if they open two envelopes but you win if they open one.

We have a few free variables:

— c: the number of contestants who will be picked
— s: the number of letters we send in
— a: the number of other letters sent by other people

The Billboard Exercise: 1 person

$$p_{c=1}\left(\text{picked exactly once}\right) = \frac{s}{a+s}$$

The Billboard Exercise: 2 persons

$$p_{c=2}(\text{picked once}) = \frac{s}{a+s} \times \frac{a}{a+s-1} + \frac{a}{a+s} \times \frac{s}{a+s-1}$$

$$= \frac{s \times a}{(a+s) \times (a+s-1)} + \frac{s \times a}{(a+s) \times (a+s-1)}$$

$$= 2 \times \frac{s \times a}{(a+s) \times (a+s-1)}$$

The Billboard Exercise: 3 persons

$$
\begin{aligned}
p_{c=3}(\text{picked once}) = &\frac{s}{a+s} \times \frac{a}{a+s-1} \times \frac{a-1}{a+s-2} \\
&+ \frac{a}{a+s} \times \frac{s}{a+s-1} \times \frac{a-1}{a+s-2} \\
&+ \frac{a}{a+s} \times \frac{a-1}{a+s-1} \times \frac{s}{a+s-2}
\end{aligned}
$$

The Billboard Exercise: 3 persons

But now it gets a bit tricky. How do we optimise for s?

$$p_{c=3}(\text{picked once}) = 3 \times \frac{s \times a \times (a-1)}{(a+s) \times (a+s-1) \times (a+s-2)}$$

Differentiating (forget solving) this is hard, even `sympy`:

$$-\frac{as\,(a-1)}{(a+s)\,(a+s-2)\,(a+s-1)^2} - \frac{as\,(a-1)}{(a+s)\,(a+s-2)^2\,(a+s-1)} - \frac{as\,(a-1)}{(a+s)^2\,(a+s-2)\,(a+s-1)} + \frac{a\,(a-1)}{(a+s)\,(a+s-2)\,(a+s-1)}$$

The Billboard Exercise: rephrasing

$$p_{c=3}\left(\text{picked once}\right) = 3 \times \frac{s \times a \times (a-1)}{(a+s) \times (a+s-1) \times (a+s-2)}$$

Let's now replace the variable $a + s = t$ such that t represents the total amount of letters sent in.

$$p_{c=3}\left(\text{picked once}\right) = 3 \times \frac{s \times (t-s) \times (t-s-1)}{t \times (t-1) \times (t-2)}$$

The Billboard Exercise: rephrasing

$$p_{c=3}(\text{picked once}) = 3 \times \frac{s \times (t-s) \times (t-s-1)}{t \times (t-1) \times (t-2)}$$

Solving this system is a lot easier since the lower part of the fraction merely contains a constant with regards to s. The rest of the maths is left to `sympy`:

$$\text{opt}(s) = \frac{2t}{3} + \frac{\sqrt{t^2 - t + 1}}{3} - \frac{1}{3}$$

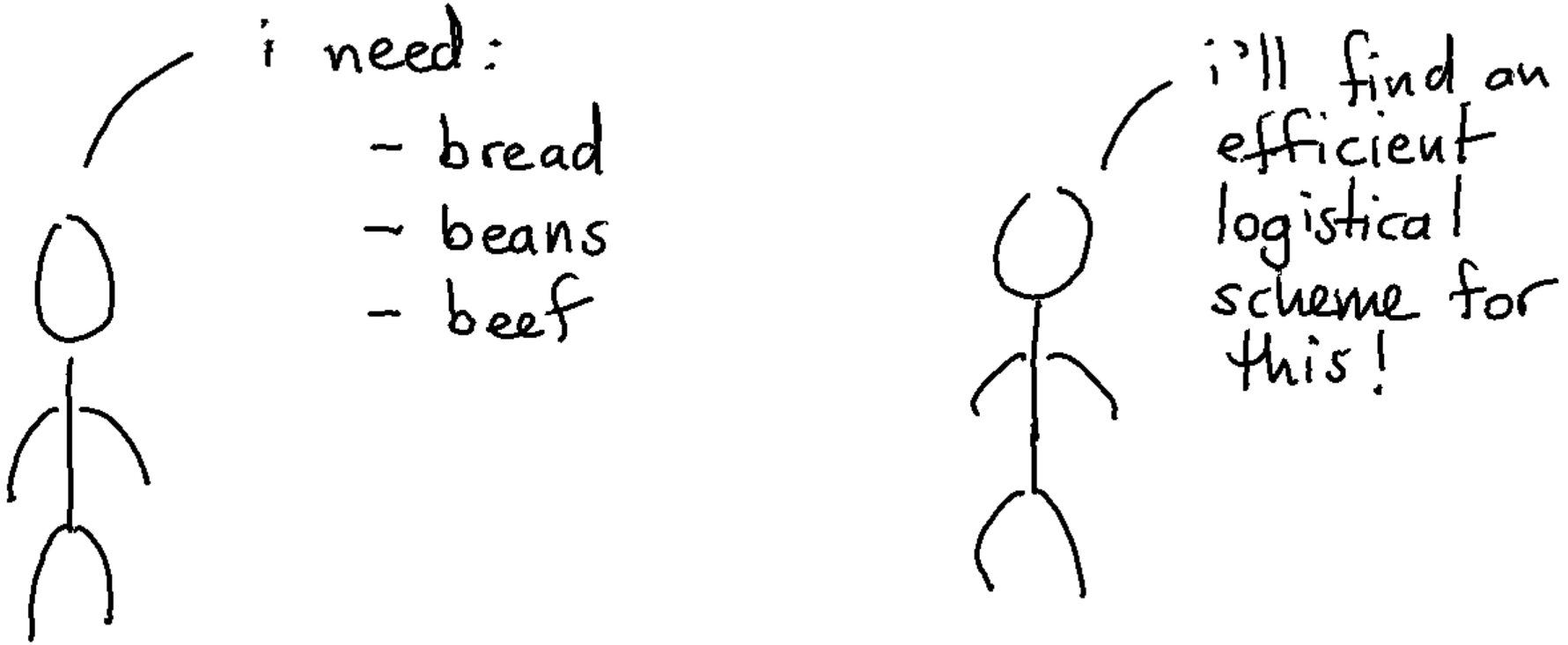
Observation

The exercise was hard until we rephrased it.

This mental step might have been small but it had a surprisingly large side effect.

I've noticed in machine learning that we've sometimes also got this option but it is hard to recognise. Rephrasing is an awesome tool to solve the real problem though.

World Food Rephrasing: Second Hand Story



World Food Rephrasing: Second Hand Story

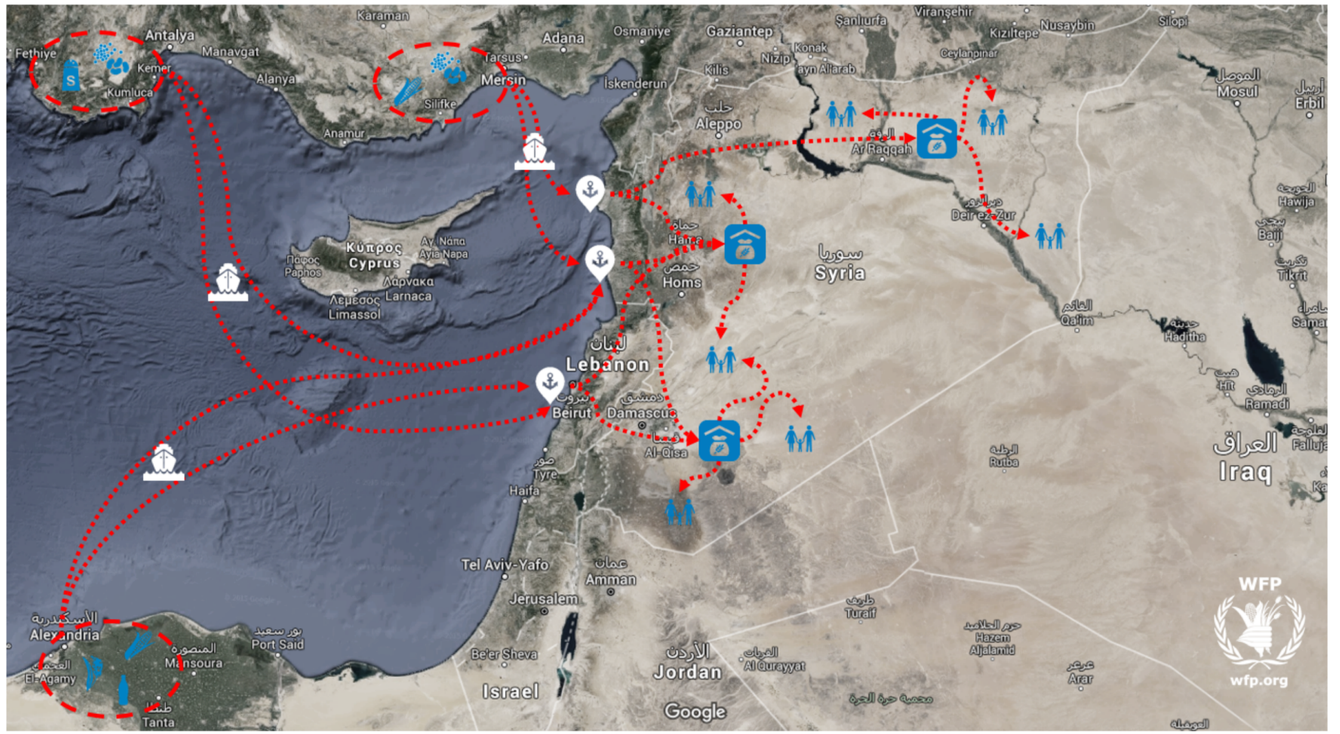


Figure 1: Illustrative example of a WFP supply chain. The figure shows how commodities move from international suppliers to discharge ports (by sea), from discharge ports to extended delivery points / warehouses, and from these warehouses to final delivery points.

Stigler Diet

TSP-like problems are usually quite tricky.

— they are NP-hard
— the algorithms don't scale well
— it's not unheard of to hear stories of servers running for days/weeks to solve the problem

Typical Algorithms

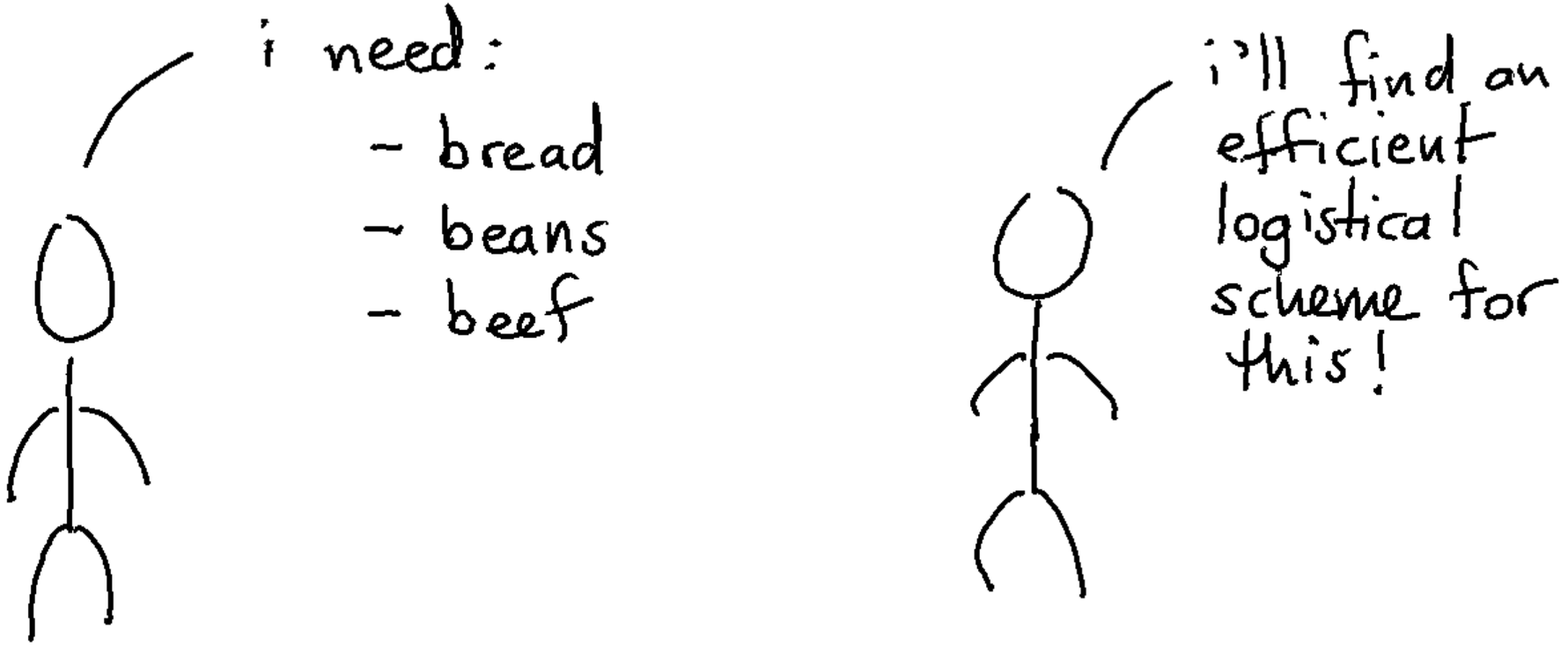
— christofedes
— cheapest/furthest insertion
— 2opt/3opt
— simulated annealing
— greedy

Typical Algorithms

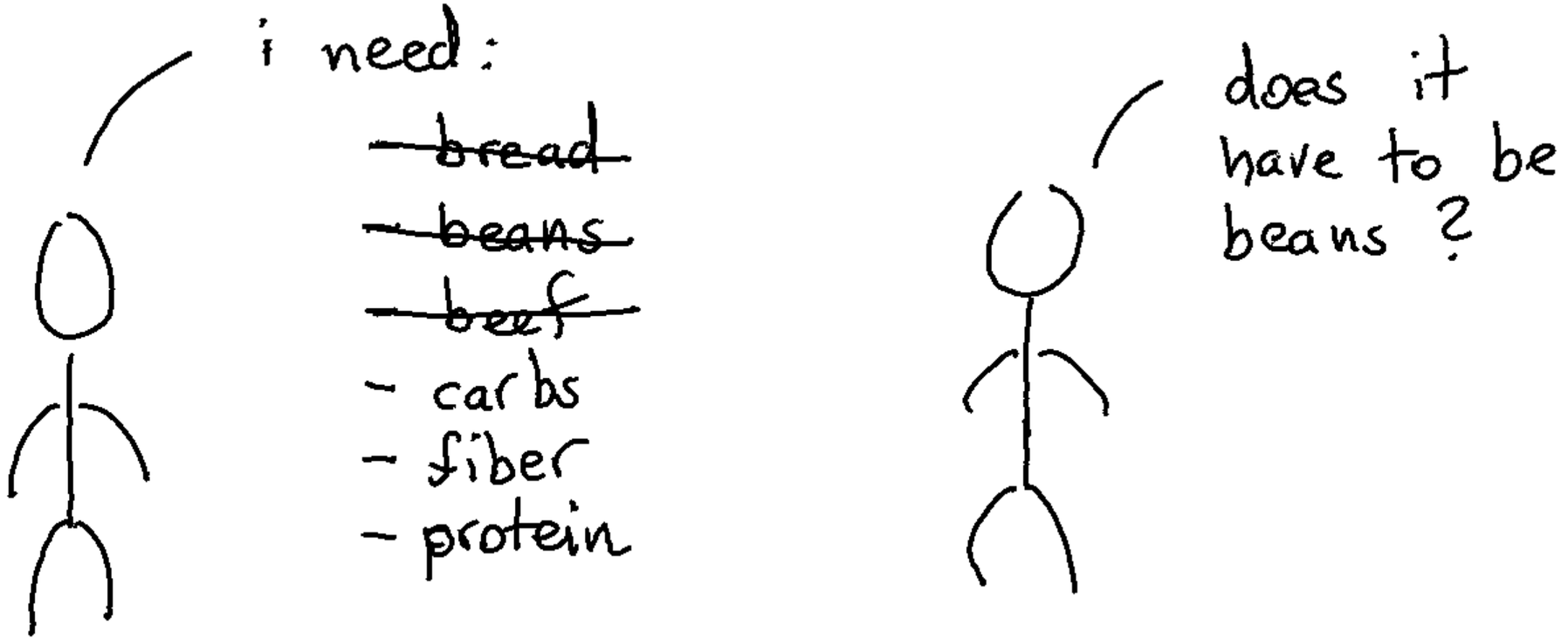
— christofedes
— cheapest/furthest insertion
— 2opt/3opt
— simulated annealing
— greedy

The algorithm does not matter much ... because the problem is the WrongProblem[tm].

World Food Rephrasing: Second Hand Story



World Food Rephrasing: Second Hand Story



Stigler Diet

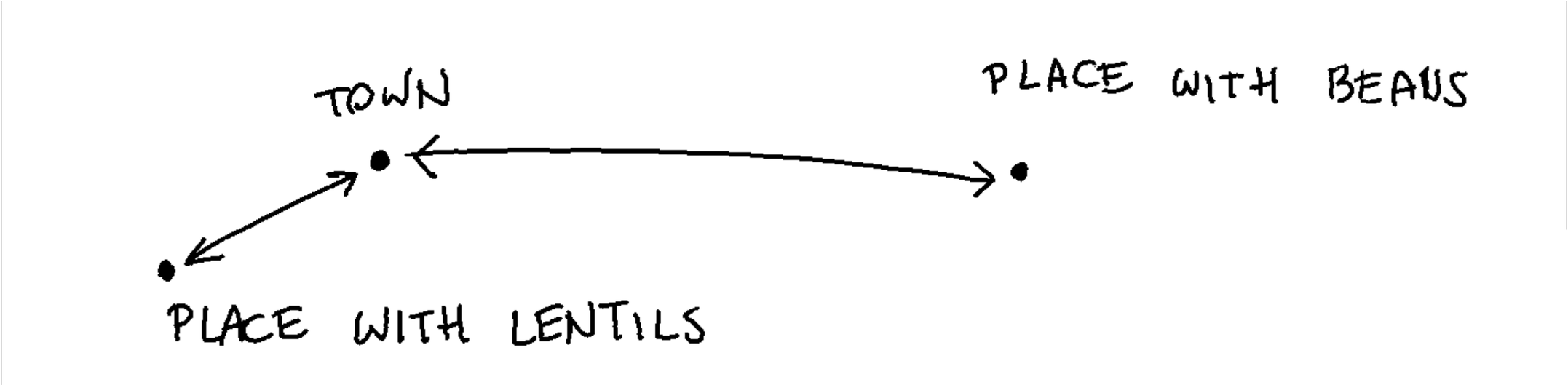
Nutrient	Daily Recommended Intake
Calories	3,000 Calories
Protein	70 grams
Calcium	.8 grams
Iron	12 milligrams
Vitamin A	5,000 IU
Thiamine (Vitamin B1)	1.8 milligrams
Riboflavin (Vitamin B2)	2.7 milligrams
Niacin	18 milligrams
Ascorbic Acid (Vitamin C)	75 milligrams

Stigler Diet

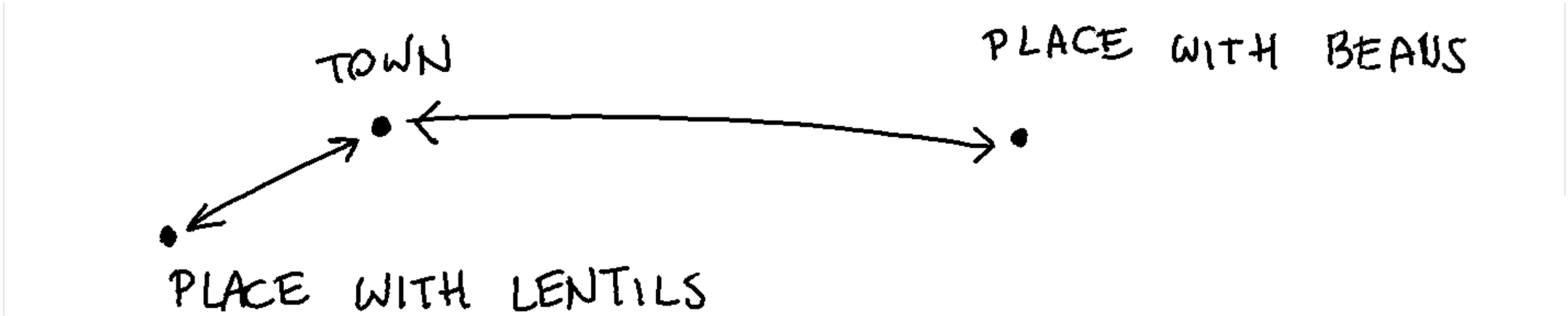
Commodity	Unit	1939 price (cents)	Calories	Protein (g)	Calcium (g)	Iron (mg)	Vitamin A (IU)	Thiamine (mg)	Riboflavin (mg)	Niacin (mg)	Ascorbic Acid (mg)
Wheat Flour (Enriched)	10 lb.	36	44.7	1411	2	365	0	55.4	33.3	441	0
Macaroni	1 lb.	14.1	11.6	418	0.7	54	0	3.2	1.9	68	0
Wheat Cereal (Enriched)	28 oz.	24.2	11.8	377	14.4	175	0	14.4	8.8	114	0
Corn Flakes	8 oz.	7.1	11.4	252	0.1	56	0	13.5	2.3	68	0
Corn Meal	1 lb.	4.6	36.0	897	1.7	99	30.9	17.4	7.9	106	0
Hominy Grits	24 oz.	8.5	28.6	680	0.8	80	0	10.6	1.6	110	0
Rice	1 lb.	7.5	21.2	460	0.6	41	0	2	4.8	60	0
Rolled Oats	1 lb.	7.1	25.3	907	5.1	341	0	37.1	8.9	64	0
White Bread (Enriched)	1 lb.	7.9	15.0	488	2.5	115	0	13.8	8.5	126	0
Whole Wheat Bread	1 lb.	9.1	12.2	484	2.7	125	0	13.9	6.4	160	0

Stigler TSP

When you realise that you can replace beans (maybe far away) with lentils (maybe be much closer) then you certainly want to be able to consider it.

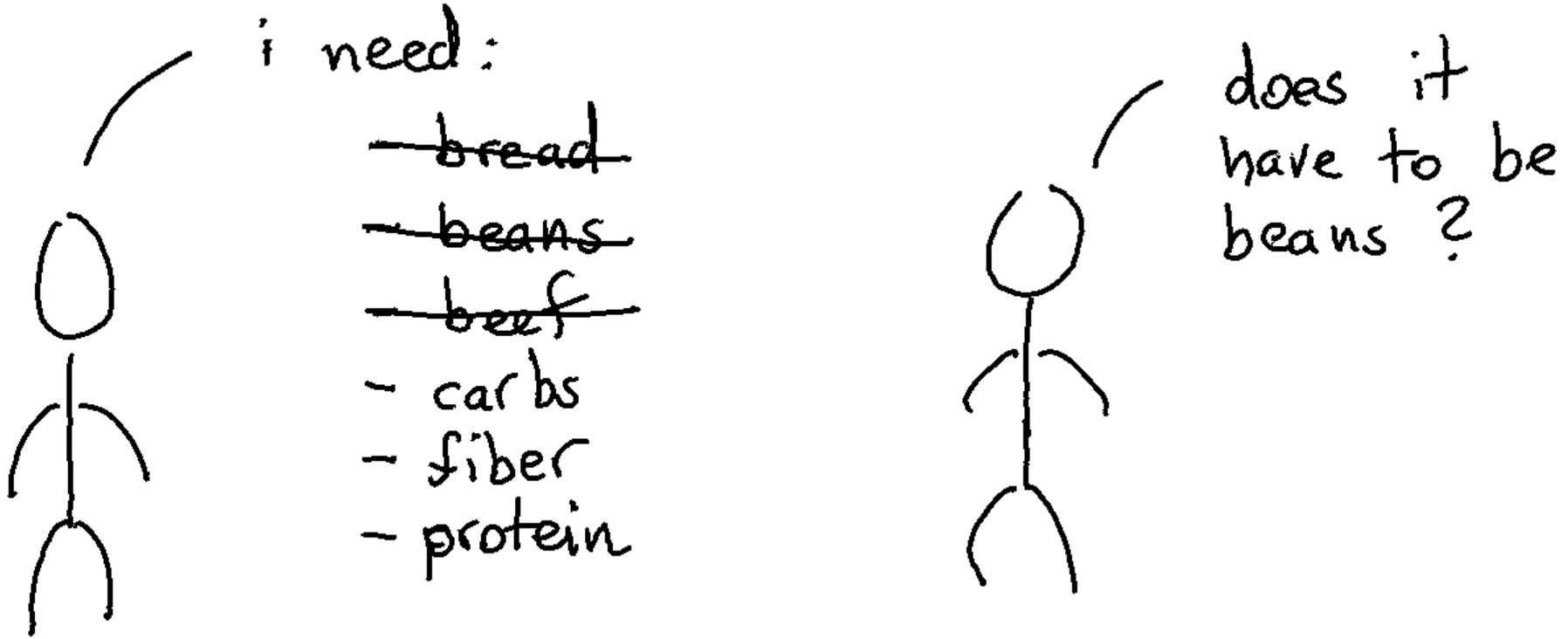


Stigler TSP

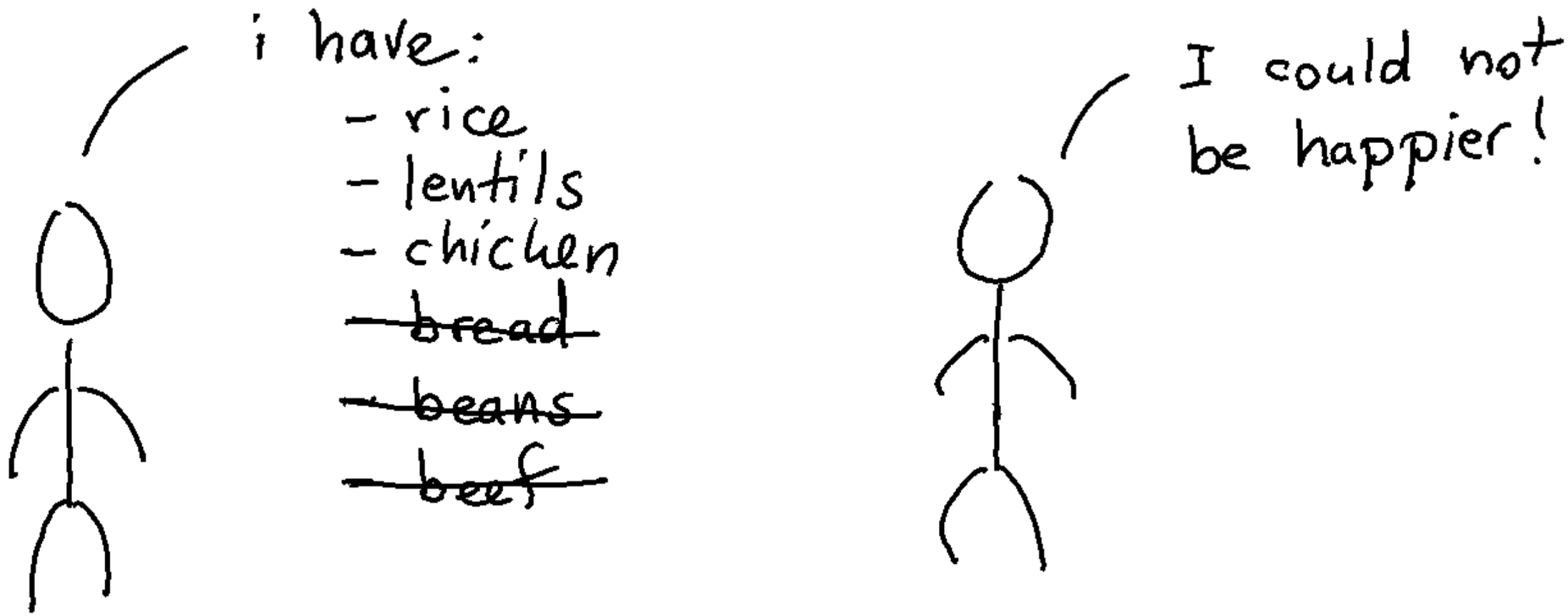


The algorithm won't figure this out on it's own. We need to design an algorithm to take this into account. If the algorithm isn't aware of this search space it won't look for it.

World Food Rephrasing: Second Hand Story



World Food Rephrasing: Second Hand Story

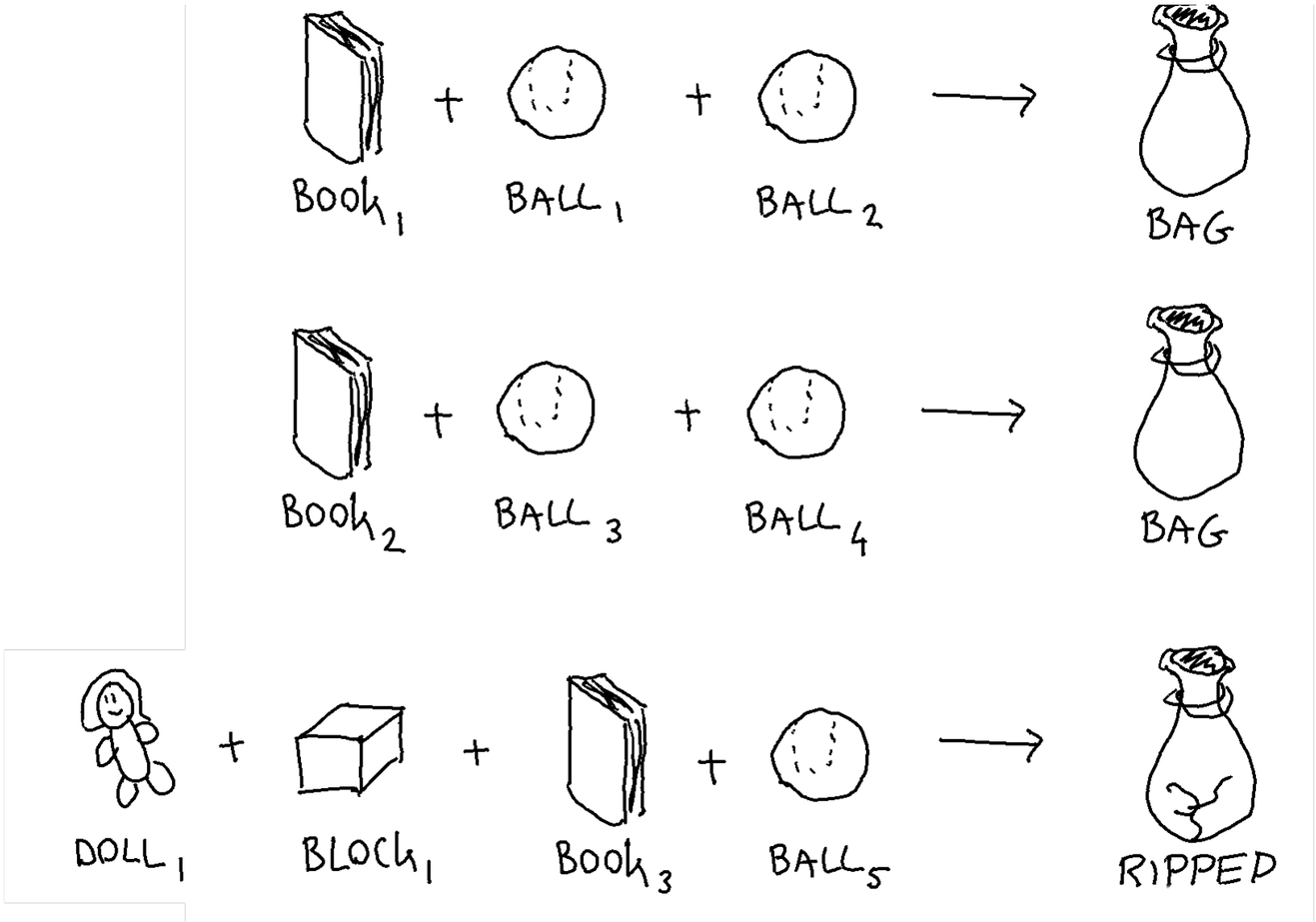


There's a risk if you're in a team with only algorithm people because it is hard to test if the algorithm you have in your mind fits the real world.

It wasn't the algorithm that saved the day, rather understanding of the world. A better algorithm would yield a worse outcome if it is used on the wrong problem.

Turns out, that this "take a step back"/"rethinking" trick also works for purely mechanical problems on kaggle that don't pertain to a "real world problem".

Kaggle's Santa Problem

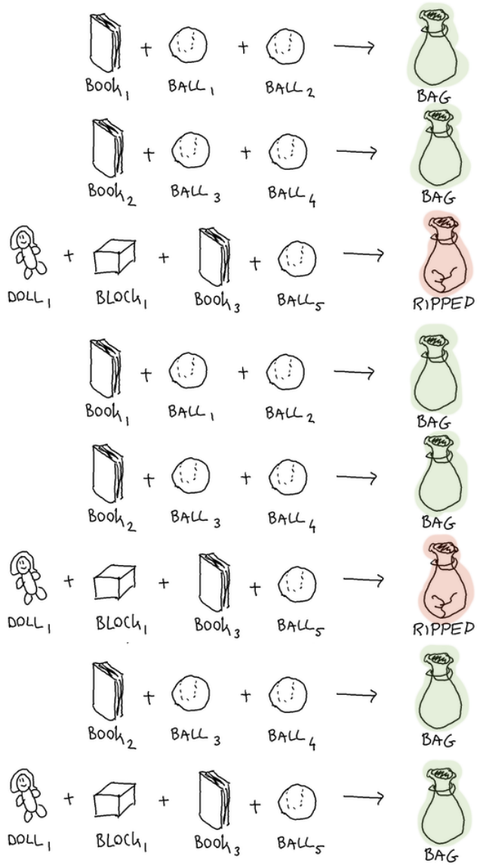


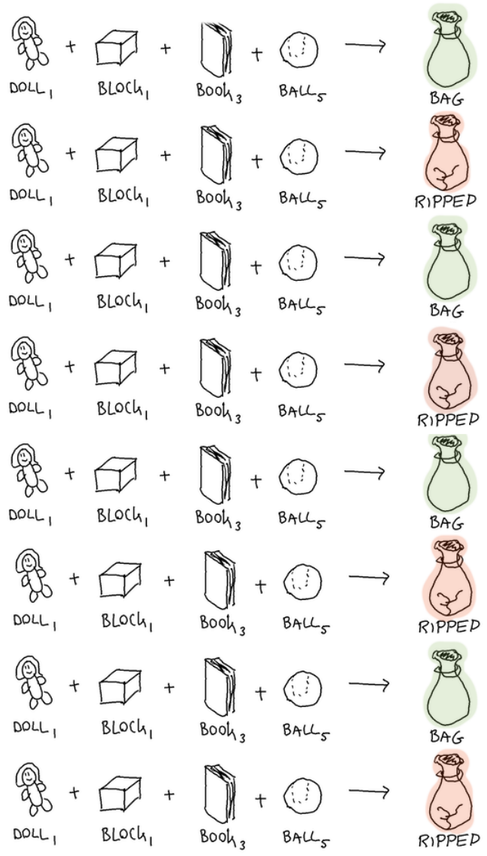
Kaggle Gave Code

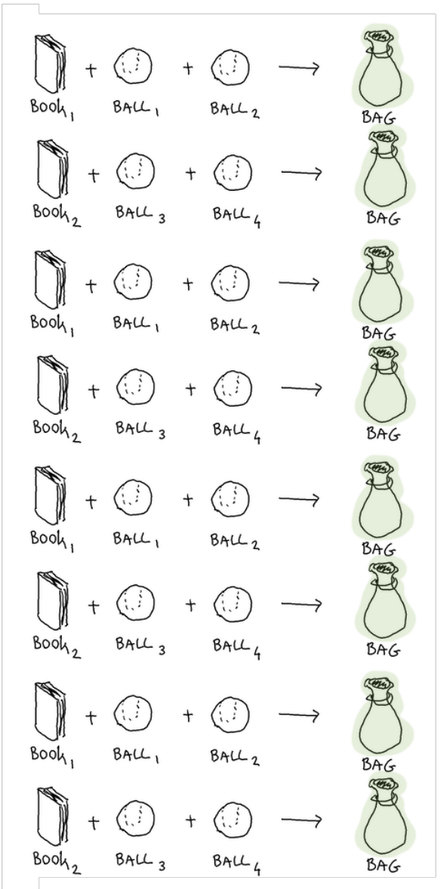
Weights of gifts were sampled via:

```
horse = max(0, np.random.normal(5,2,1)[0])
ball = max(0, 1 + np.random.normal(1,0.3,1)[0])
bike = max(0, np.random.normal(20,10,1)[0])
train = max(0, np.random.normal(10,5,1)[0])
coal = 47 * np.random.beta(0.5,0.5,1)[0]
book = np.random.chisquare(2,1)[0]
doll = np.random.gamma(5,1,1)[0]
block = np.random.triangular(5,10,20,1)[0]
...
```

This is the Problem







Maths

Here's my formulation of the problem.

$$\max \sum_i \sum_j I_{ij} \mathbf{E}(w_i)$$

$$\text{subject to}$$

$$\sum_j I_{ij} \mathbf{E}(w_i) \leq \text{risk}(\text{max weight}) \; \forall j$$

Here $I_{ij} \in \{0, 1\}$ indicates if gift i goes into bag j.

Maths

This was the basis of my solution set.

1. Use OR-tools with simulated weights.
2. Generate solutions based on different preferences for risk.
3. Upload often and pray.

Others did guessing games, they tried to learn the weights by uploading (max 3 uploads a day).

Coolest Hack

After a few weeks though, somebody on the forums demonstrated that we were all solving the wrong problem.

Let's focus on the epic hint that was given.

Remember, Kaggle gave this.

Weights of gifts were sampled via:

```
horse = max(0, np.random.normal(5,2,1)[0])
ball = max(0, 1 + np.random.normal(1,0.3,1)[0])
bike = max(0, np.random.normal(20,10,1)[0])
train = max(0, np.random.normal(10,5,1)[0])
coal = 47 * np.random.beta(0.5,0.5,1)[0]
book = np.random.chisquare(2,1)[0]
doll = np.random.gamma(5,1,1)[0]
block = np.random.triangular(5,10,20,1)[0]
...
```

Remember, Kaggle gave this.

Presumably, this was part of that code ...

```
np.random.seed(s)
horse = max(0, np.random.normal(5,2,1)[0])
ball = max(0, 1 + np.random.normal(1,0.3,1)[0])
bike = max(0, np.random.normal(20,10,1)[0])
train = max(0, np.random.normal(10,5,1)[0])
coal = 47 * np.random.beta(0.5,0.5,1)[0]
book = np.random.chisquare(2,1)[0]
doll = np.random.gamma(5,1,1)[0]
block = np.random.triangular(5,10,20,1)[0]
...
```

Toby Cheese
11th place

An unexpected discovery

posted in Santa's Uncertain Bags 2 years ago

46

So, last Thursday, four days before the competition deadline, I finished coding my solution. I still had to update it with the results of the remaining four days worth of submissions, but the code was done. Yet there was this one thing left in this competition that I have been wanting to do ever since the discussion here (whether the accusations are justified or not I don't know, but that thread was what sparked my interest). And that thing was to see if it's maybe possible to brute-force the seeds used to generate the toy weights.

Imagine my disappointment when I actually succeeded. Because if I can do it, so can others. Sure, it took me a few tries and a bit of optimising my code, but once I had done that, I broke the seed for the first bag type in less than a minute, the next two not taking significantly longer.

The following things are what made my attempt possible:

- The actual weights were generated with numpy.random(), exactly as described on the competition page
- There is one unique seed per bag type, which reduces the amount of random data to be produced for each check
- The number of possible seeds for numpy.random.seed() is only 4294967295 (I initially expected it to be sys.maxint, i.e. 9223372036854775807, which would have called for some luck to find the seed in workable time)
- Tests can be parallelized without messing up seeding

Remember, Kaggle gave this.

Presumably, this was part of that code ...

```
for s in range(...):
  np.random.seed(s)
  horse = max(0, np.random.normal(5,2,1)[0])
  ball = max(0, 1 + np.random.normal(1,0.3,1)[0])
  ...
  score = submission_score(allocation)
  if score == what_kaggle_confirmed:
    print("i can haz win!")
```

William Cukierski Kaggle Team • 2 years ago • Options • Reply

13

Nice catch Toby! I am embarrassed to admit that, in my pre-holidays scramble to put this together, I thought enough about seeds to vary the seed, but not enough to remember that you could probe them offline. The code was effectively:

```
np.random.seed(782356)
horses = [Horse(x) for x in range(num_horses)]
balls = [Ball(x) for x in range(num_balls)]
bikes = [Bike(x) for x in range(num_bikes)]
trains = [Train(x) for x in range(num_trains)]
np.random.seed(2956325) # Intentionally pick new seed here to discourage seed snooping
coals = [Coal(x) for x in range(num_coal)]
books = [Book(x) for x in range(num_books)]
dolls = [Doll(x) for x in range(num_dolls)]
blocks = [Block(x) for x in range(num_blocks)]
gloves = [Gloves(x) for x in range(num_gloves)]
```

Yes, that comment is in the source code. *Sigh*.

We didn't publish new weights near the end because it seemed like people were having too much fun playing the "game within the game".

Hope this was fun nonetheless. We may do more of these in the coming year.

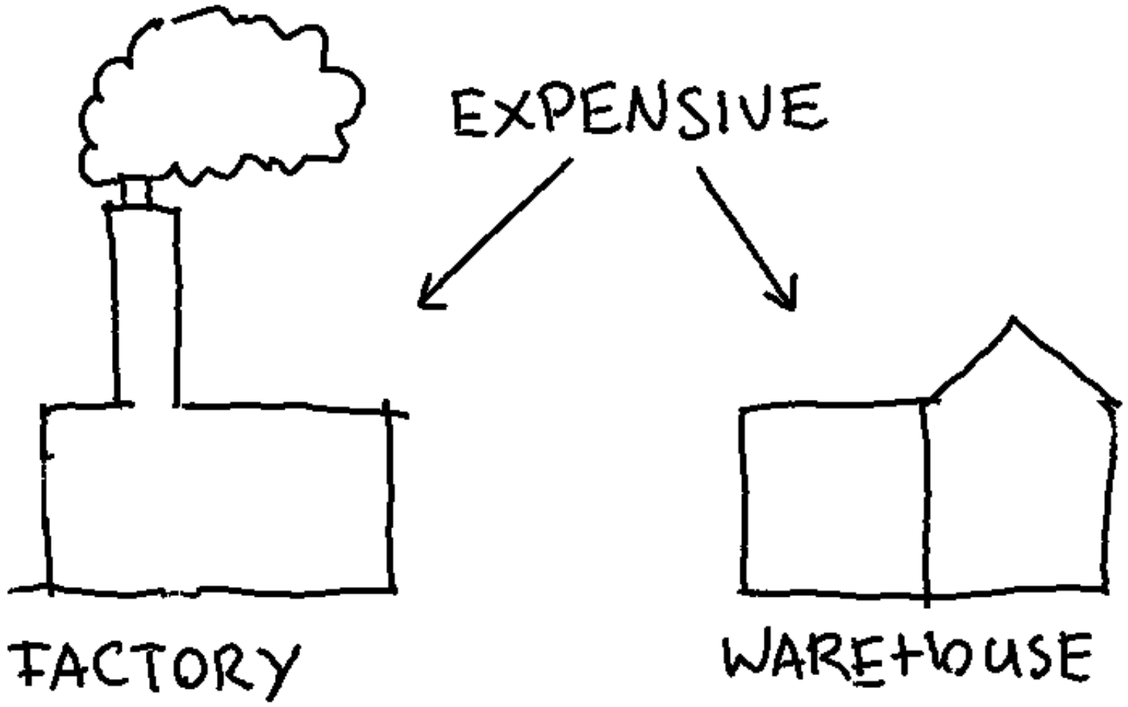
Lessons

— It is really hard to think outside the box when you're in it. This is why the optimal solution on kaggle was almost an afterthough.

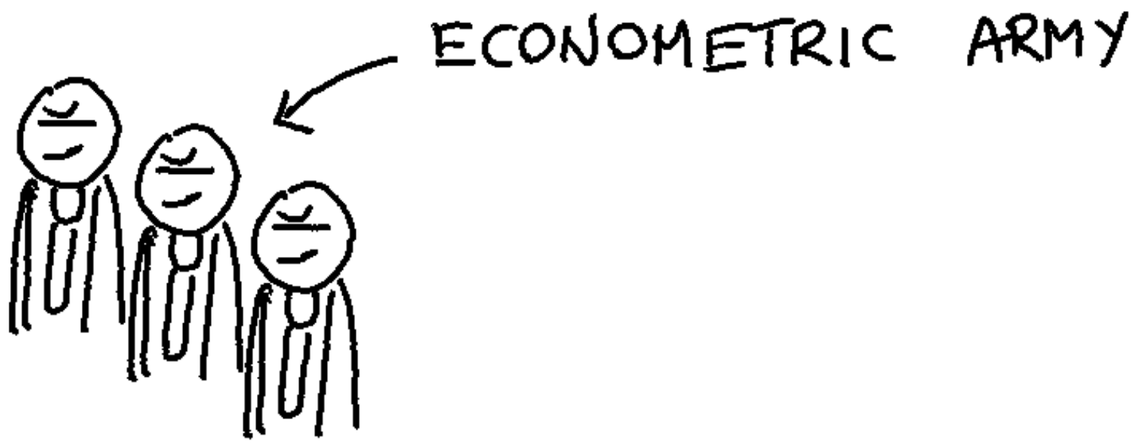
— It does show that good ideas pop around when you're not in "work-work" mode but rather in "work-play" mode. Encouraging this sounds like a no brainer, but very few people actively pursuit this.

The next story demonstrates this point once more.

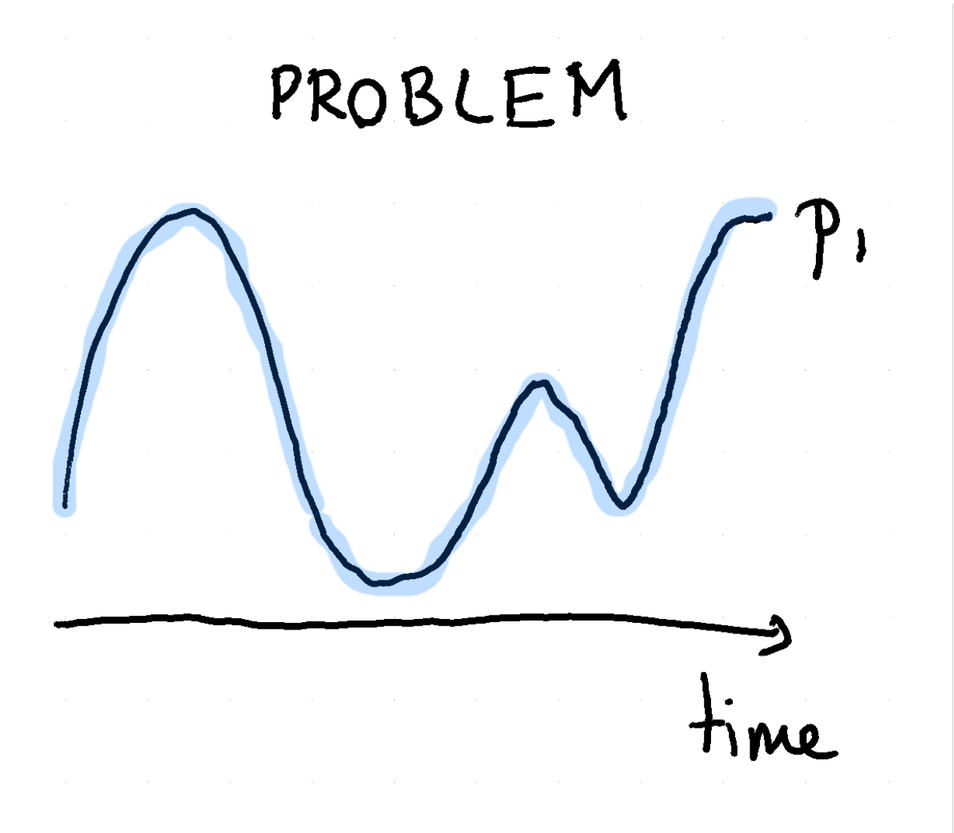
One Final Fable



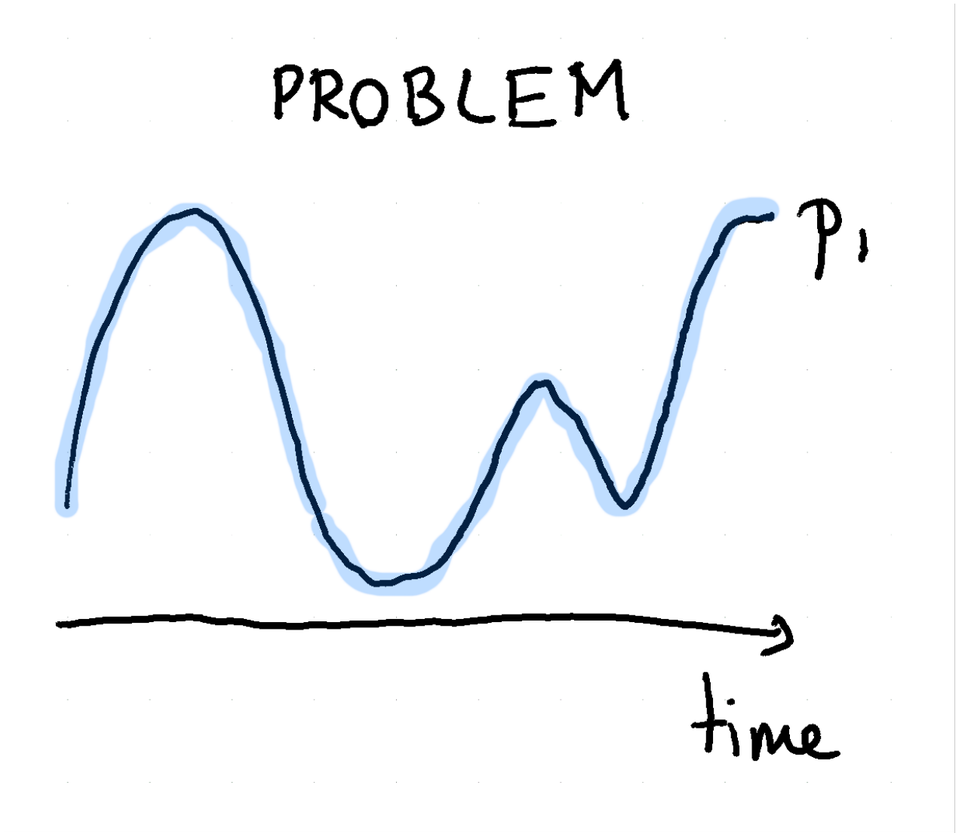
One Final Fable

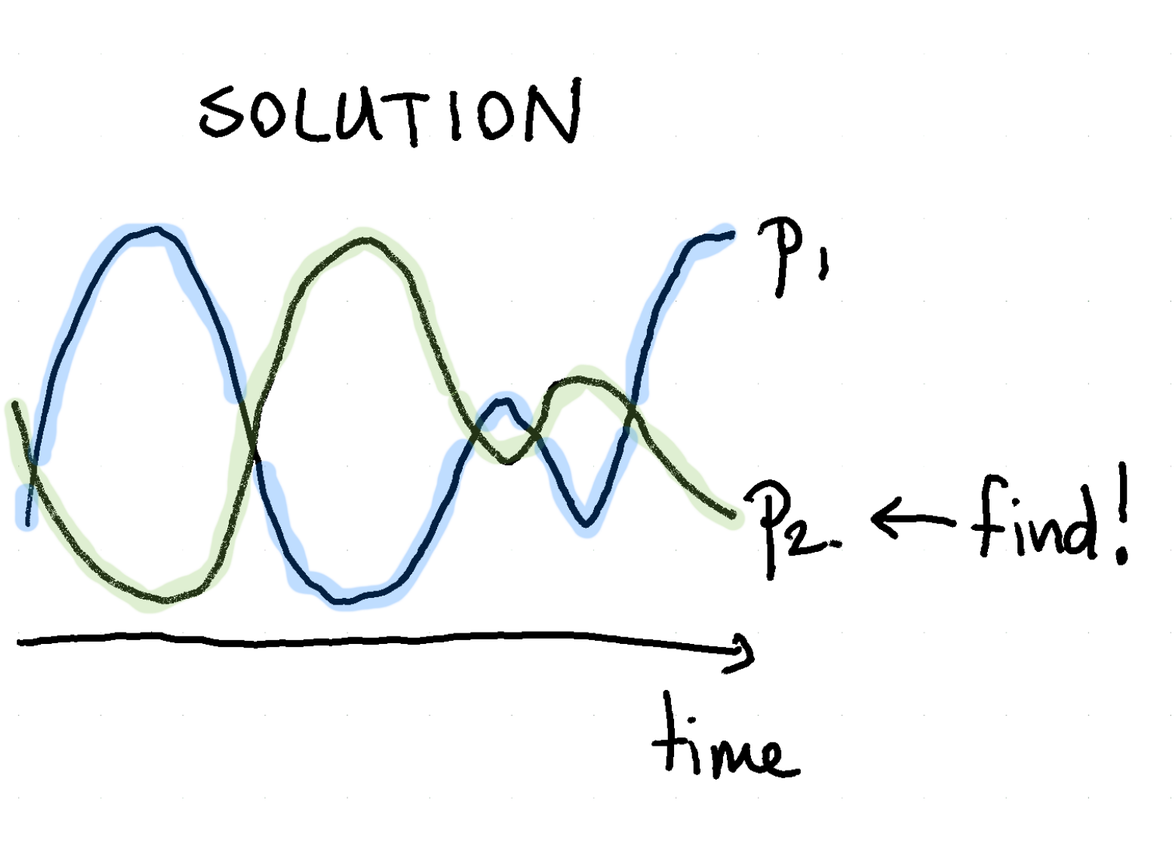


One Final Fable

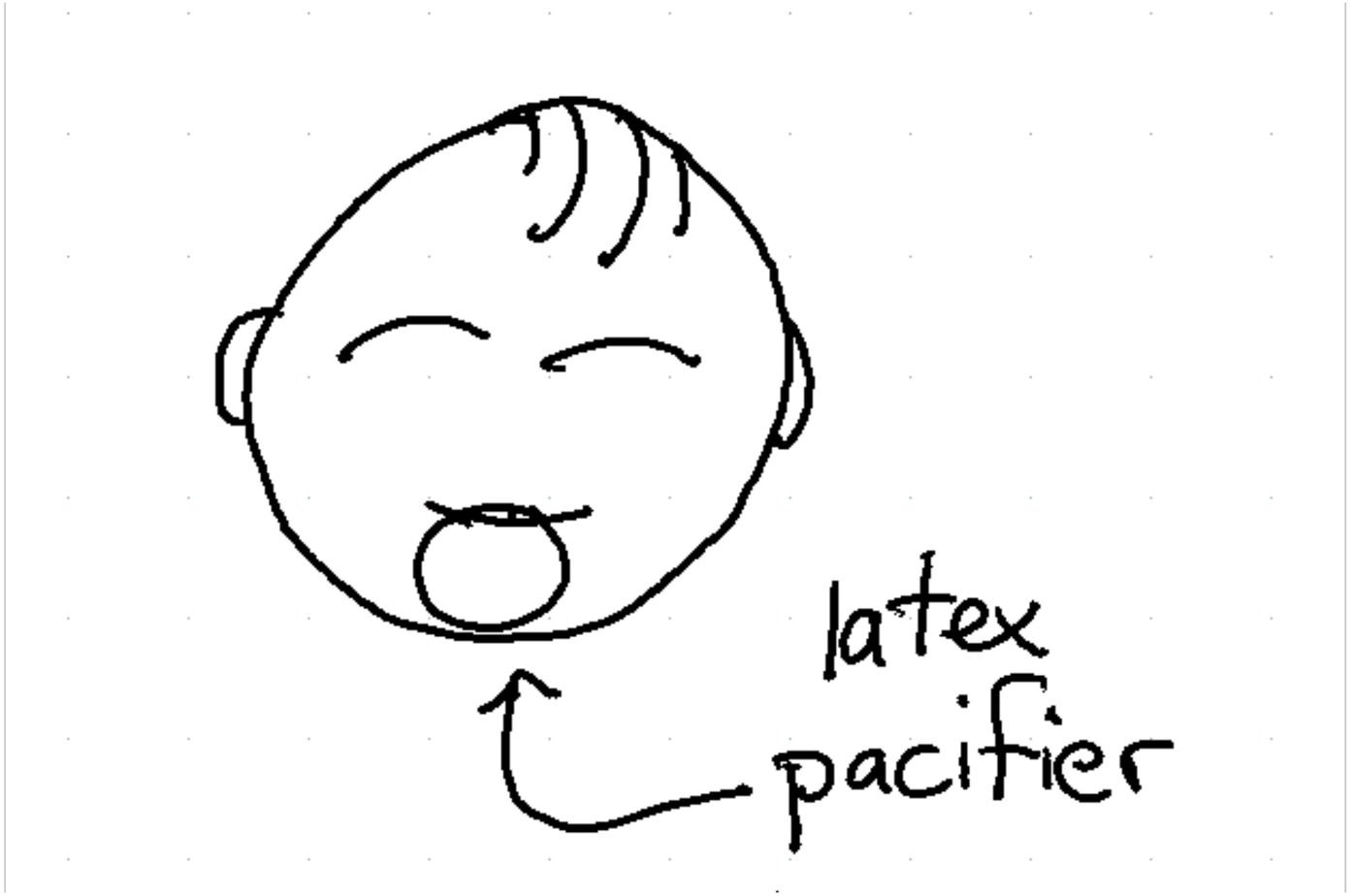


One Final Fable





One Final Fable: Latex Pacifier



Parting words:

It might be good to get better at:

— understanding the theatre
— recommending the next episode
— considering lentils over beans
— using street smarts at kaggle
— pivoting to pacifiers

Parting words:

— $v(\text{Understand}(\text{Problem})) > v(\text{Understand}(\text{Solution}))$

— Think more about the system instead of the algorithm. Don't optimise a single part, rather optimise the communication between two parts.

— Rephrasing might be you friend.

— Maybe stop being a data team and start behind an interdisciplinary team that owns a product.

Advice

— Don't live in a rabit hole.

Advice

— Don't live in a rabit hole.

— Try to prevent intellectual laziness. It's is OK to admit that you need to rethink a problem.

— Have a break once in a while. A lot of problems solve themselves if you go running or have a baantjer moment in a bar with friends.

Advice

— Don't live in a rabit hole.

— Try to prevent intellectual laziness. It's is OK to admit that you need to rethink a problem.

— Have a break once in a while. A lot of problems solve themselves if you go running or have a baantjer moment in a bar with friends.

— Go to your local theatre, there's some **AMAZING** things happening on stage.

MATERIAL

in the odd case Vincent is not

OVER TIME

DeepLearning[tm]

It's not just in the "application" where it is tempting to fix the wrong problem; it can often also occur in the machine learning part of it as well. I've got one example of this.

DeepLearning[tm] in vision [faces]

Life at GoDataDriven

GoDataDriven applies data innovation to ensure any business is in front of the wave. From online retail to financial services; public to telco; we transform organizations into data-driven enterprises. We're on the forefront of data technologies and data science, and we're looking for exceptional people to join us.

data engineer | **software engineer** | **lead data engineer cloud** | **data scientist** | **machine learning engineer** | **growth hacker**

WANTED: IT Professionals for Unique And Personalized Data Engineering Training Program

DeepLearning[tm] in vision [faces]

Stijn Tonk

Data Mining Scientist

bio

Robert Rodger

Data Scientist

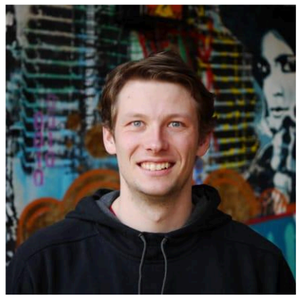
bio

Henk Griffioen

Data Scientist

bio

Bas Harenslak



Big Data Hacker

bio

Jelte Hoekstra

Data Scientist

bio

Rodrigo Agundez

Data Maverick

bio

Fokko Driesprong

Data Evangelist

bio

Nelli Gofman

Data Scientist

bio

$$\{X_n\} = 37 \rightarrow 1000+$$

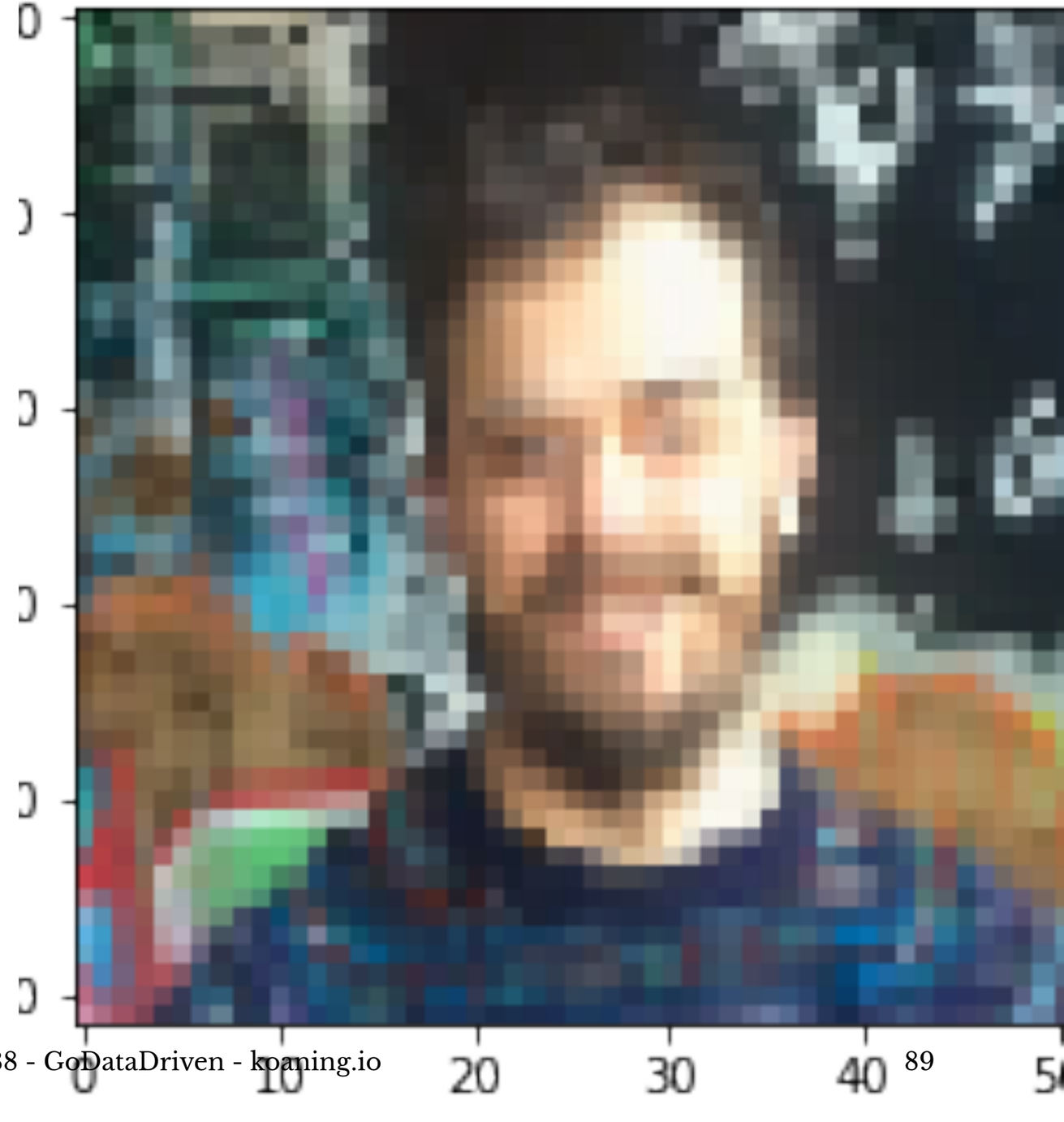
I can generate 1000s of images from my 37 collegues.

```
datagen = ImageDataGenerator(
    horizontal_flip=True,
    rotation_range=25,
    samplewise_std_normalization=True,
    zoom_range=0.2)
```

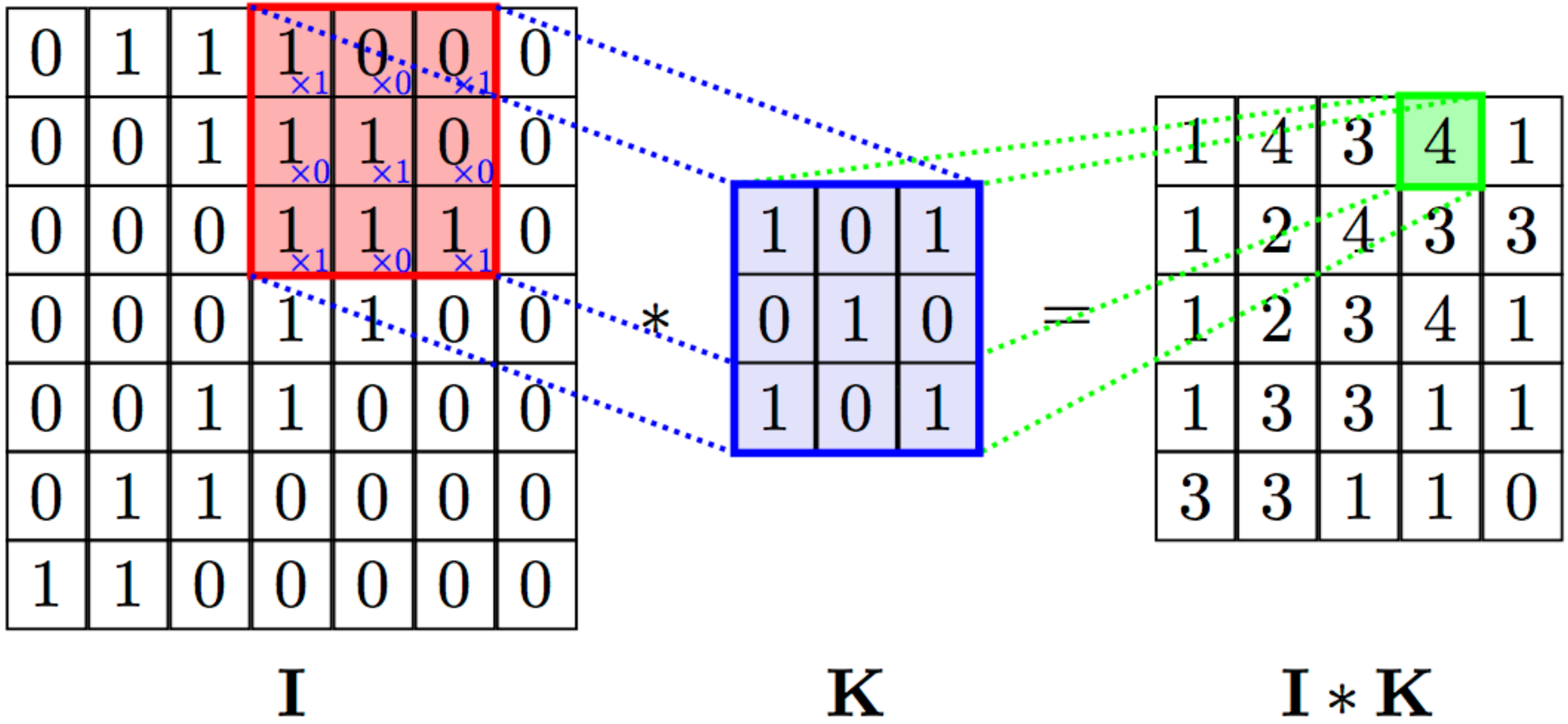
Here's an example from the generator.

With this: let's see if we can generate collegue faces. A good starting point might be to see if we can build a proper encoder.

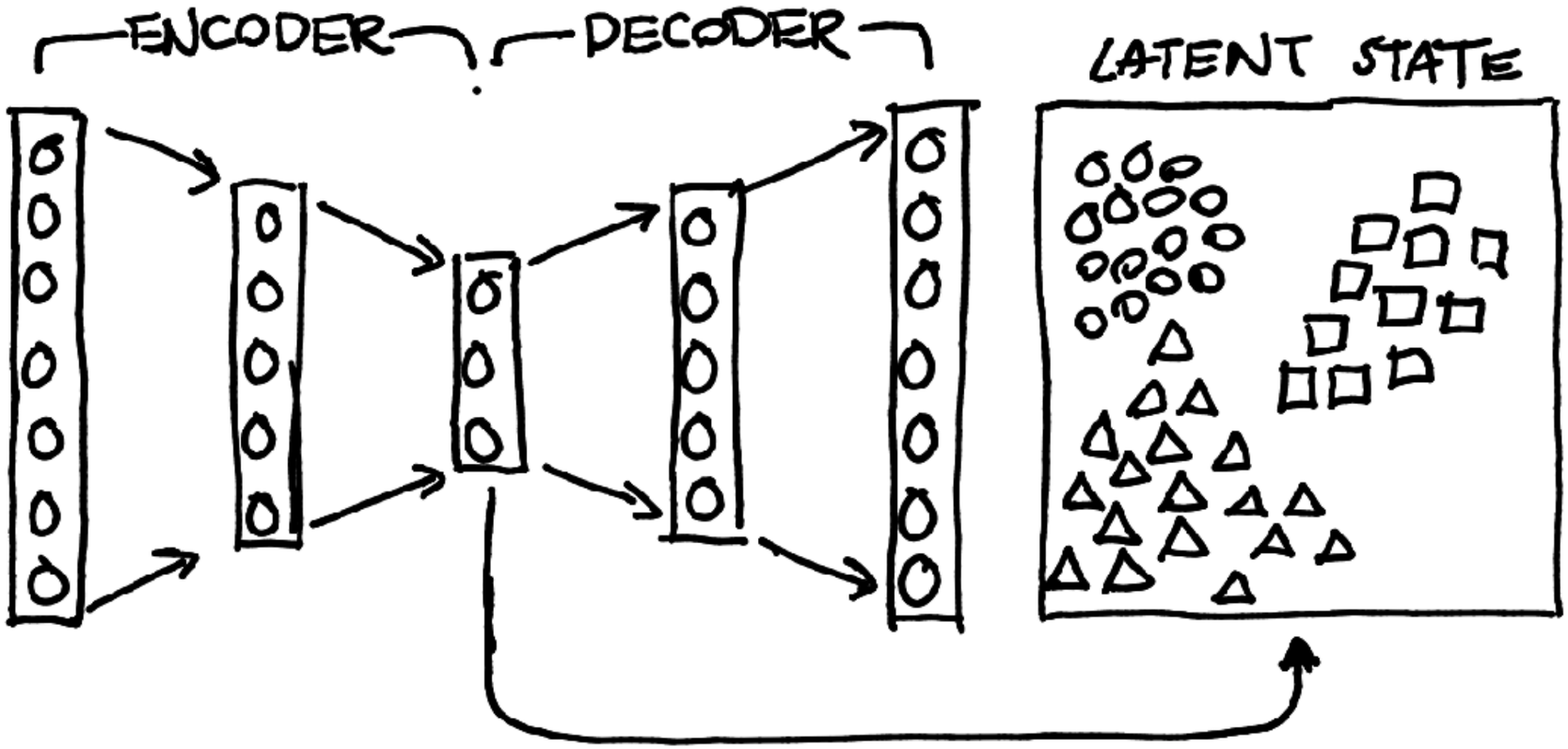
I've decided to build one in keras and after a small afternoon trying to make sure that all the layers fit I had some basic code.



I'll be using Convolutions a lot



And AutoEncoders

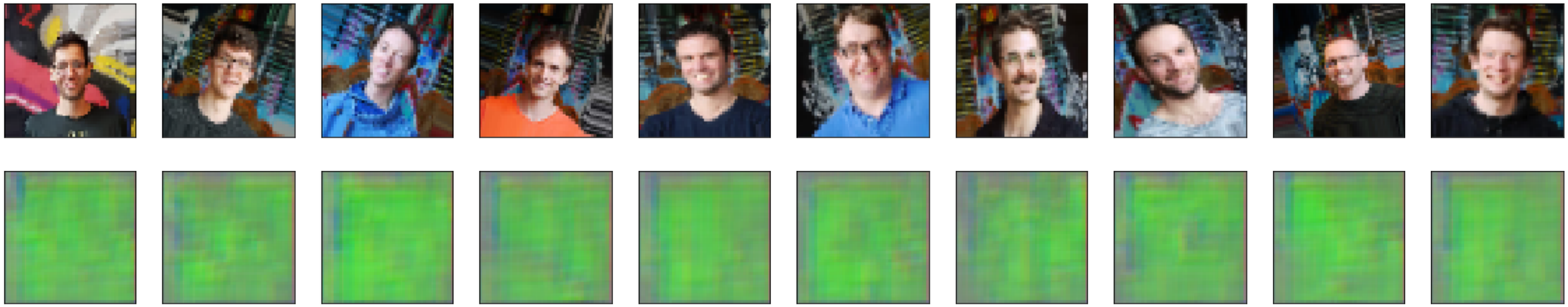


```
x = Conv2D(8, (3, 3), activation='relu', padding='same')(input_img)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same', name='encoded')(x)
encoded = Flatten()(x)
x = Reshape(target_shape = [_.value for _ in x.shape[1:]])(encoded)
x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='relu')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer=Adam(lr=0.00001), loss='binary_crossentropy')
```

It went ... horribly

So can anybody tell me why?



It went ... horribly

It was the activation; `relu` might push the embedding in the wrong direction. The embedding might be better if it is constrained.

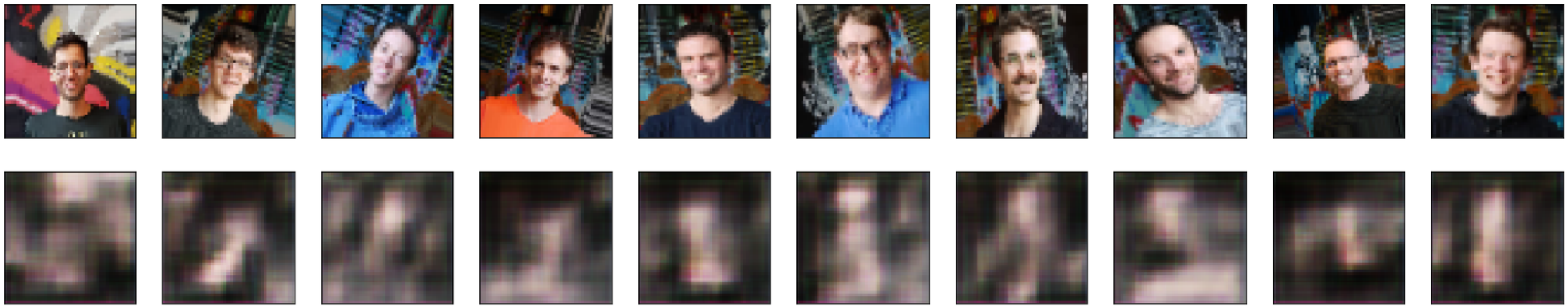
$$f(x) = \begin{cases} \text{relu} < \tanh & \text{if goal(task)} = \text{embedding} \\ \text{relu} > \tanh & \text{if goal(task)} = \text{pattern matching} \end{cases}$$

Made the improvements. Will this work?

```
x = Conv2D(8, (3, 3), activation='tanh', padding='same')(input_img)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same', name='encoded')(x)
encoded = Flatten()(x)
x = Reshape(target_shape = [_.value for _ in x.shape[1:]])(encoded)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='tanh')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)
```

It went ... meh

So can anybody tell me why?



It went ... meh

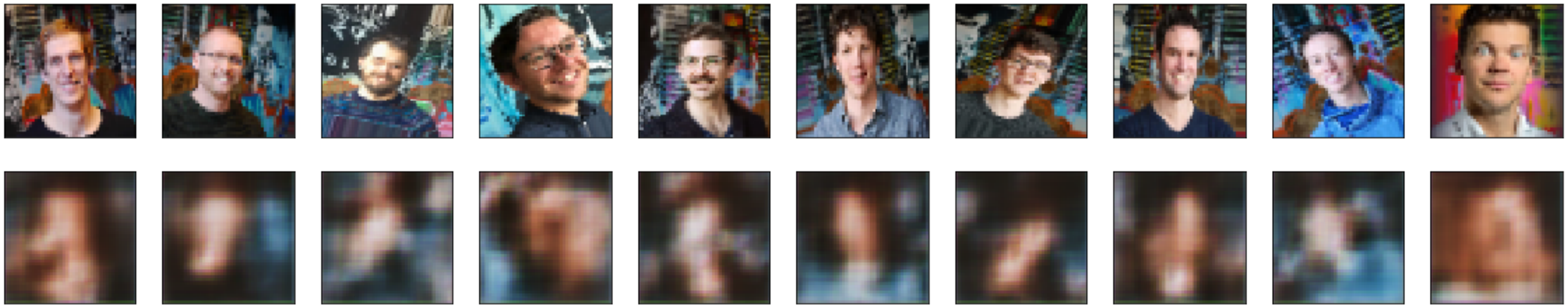
It was the pooling.

$$f(x) = \begin{cases} \text{MaxPool} > \text{AveragePool} & \text{if classification} \\ \text{MaxPool} < \text{AveragePool} & \text{if encoding} \end{cases}$$

```
x = Conv2D(8, (3, 3), activation='tanh', padding='same')(input_img)
x = BatchNormalization()(x)
x = AveragePooling2D((2, 2), padding='same')(x)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = AveragePooling2D((2, 2), padding='same')(x)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = AveragePooling2D((2, 2), padding='same', name='encoded')(x)
encoded = Flatten()(x)
x = Reshape(target_shape = [_.value for _ in x.shape[1:]])(encoded)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(4, (3, 3), activation='tanh', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='tanh')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)
```

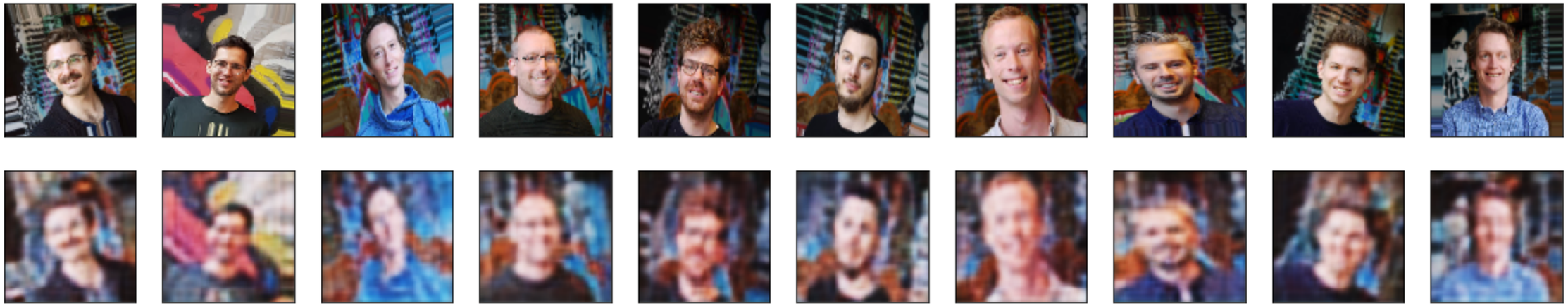
It went ... better

But the faces are still a bit blurry.



It went ... better

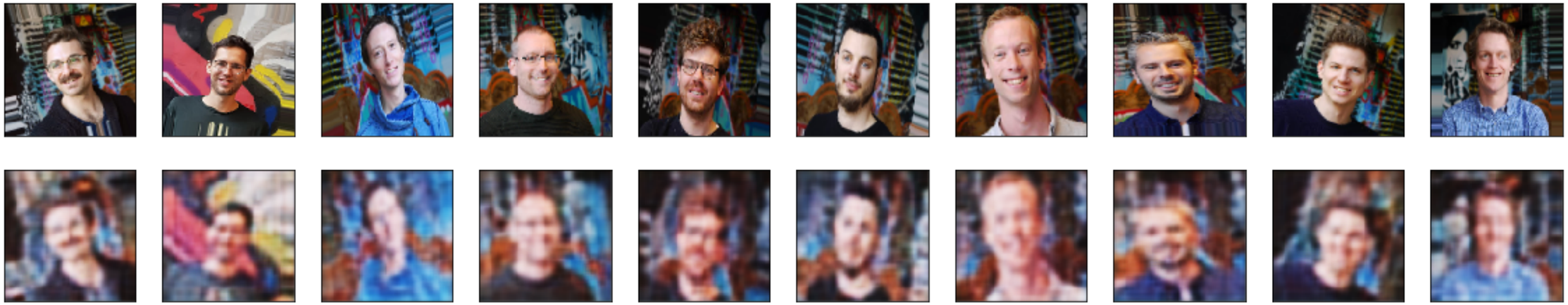
So can anybody tell me why?



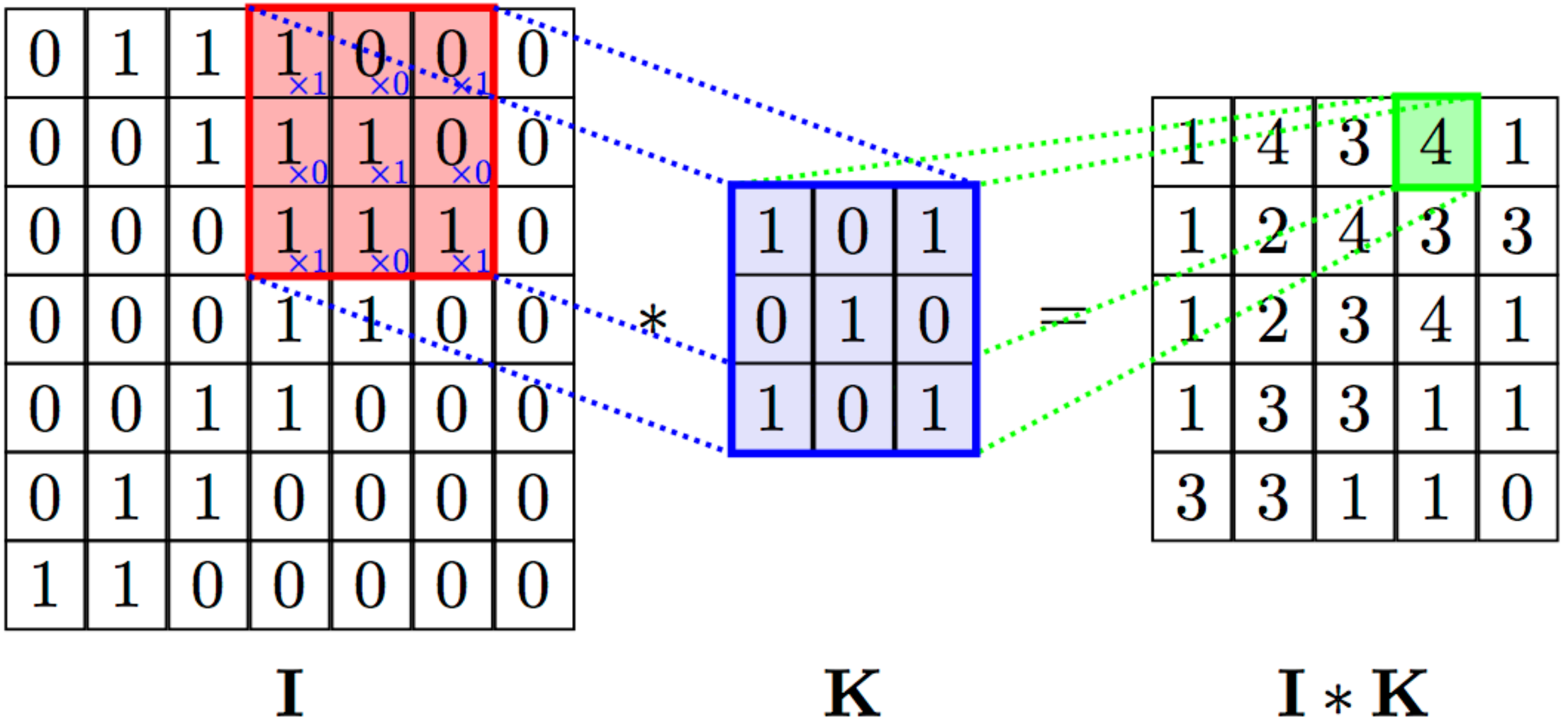
Next up, I figured it might be better to make ever smaller stepsizes as I train more and more epochs.

```
adam.lr = 0.01
autoencoder.fit_generator(godata_gen_col, epochs=10, ...)
adam.lr = 0.001
autoencoder.fit_generator(godata_gen_col, epochs=10, ...)
adam.lr = 0.0005
autoencoder.fit_generator(godata_gen_col, epochs=10, ...)
```

It went ... better



Trollololol, actually I increased the input image size!


0 1 1 1 0 0 0
0 0 1 1 1 0 0
0 0 0 1 1 1 0
0 0 0 1 1 0 0
0 0 1 1 0 0 0
0 1 1 0 0 0 0
1 1 0 0 0 0 0
×1 ×0 ×1
×0 ×1 ×0
×1 ×0 ×1
*
1 0 1
0 1 0
1 0 1
=
1 4 3 4 1
1 2 4 3 3
1 2 3 4 1
1 3 3 1 1
3 3 1 1 0
I
K
I ∗ K

Learnings

The main thing that made the difference was actually learning how convolutions work. It is very easy, fun even, to download some code from gitlab and paste it in a notebook to see what it does but not understanding it makes it tricky to really work with it. I think my natural thinking was a lot faster than a giant hyperparameter search would have been.

Spend the few hours learning what you're doing and you can start to think along much much better. It also

Appendix

— Hein Fleuren does a lot of work with the World Food Program; part of his work is described here.

— The kaggle competition comment can be found here.

— There's an amazing article from the 80ies about the fall of operations research that discusses many similar topics.