Iterative Design of a Figure

guidelines and an example

by: evantheboucy@github.io

Here contains high level guidelines on how to iterate on a figure for a scientific (specifically, computer science) paper to make it clearer, along with a real example of design iteration consisting of 6 rounds of back and forth between myself and a student.

Guidelines

Attributes and Semantics A figure contains elements of various attributes, such as color, stroke width, font size, ... These attributes can take on different values, such as blue/red, thin/thick, small/big, ... Make sure every variation of these attributes (such as difference in color) corresponds to *exactly one* semantic distinction of the objects in your figure.

For instance, varying color and using red for human-generated data, and using blue for synthetically generated data. Once this decision is made, and you notice something in the figure is yellow, you ought to ask yourself "does yellow here differentiate a difference of how the data is generated?" and if the answer is no, you need to either (1) remove the yellow color or (2) find a different attribute other than color to convey the information encoded by yellow.

You will find that by being explicit about *every* attribute variations in the figure, and trying to *minimize* the usage of different attributes in cases where a distinction of semantics is not there, the figure will contain far less visual information for the reader to process, thus, becoming clearer.

Drawing and Words A figure exists in context of the rest of the paper, which is made of words. Drawing and words convey complimentary information. It is good to ask yourself for a given piece of information, is it best explained in the figure, or should we explain it using text — either in the figure caption, or in the main body?

As a general rule, the figure should be front-loaded with the most prescient, high level information which the reader dive deeper using the text. You will find that by being explicit about whether a piece of information should be the first thing a reader would encounter in the figure or should it be discovered by the reader later in text, the figure will become simpler and clearer.

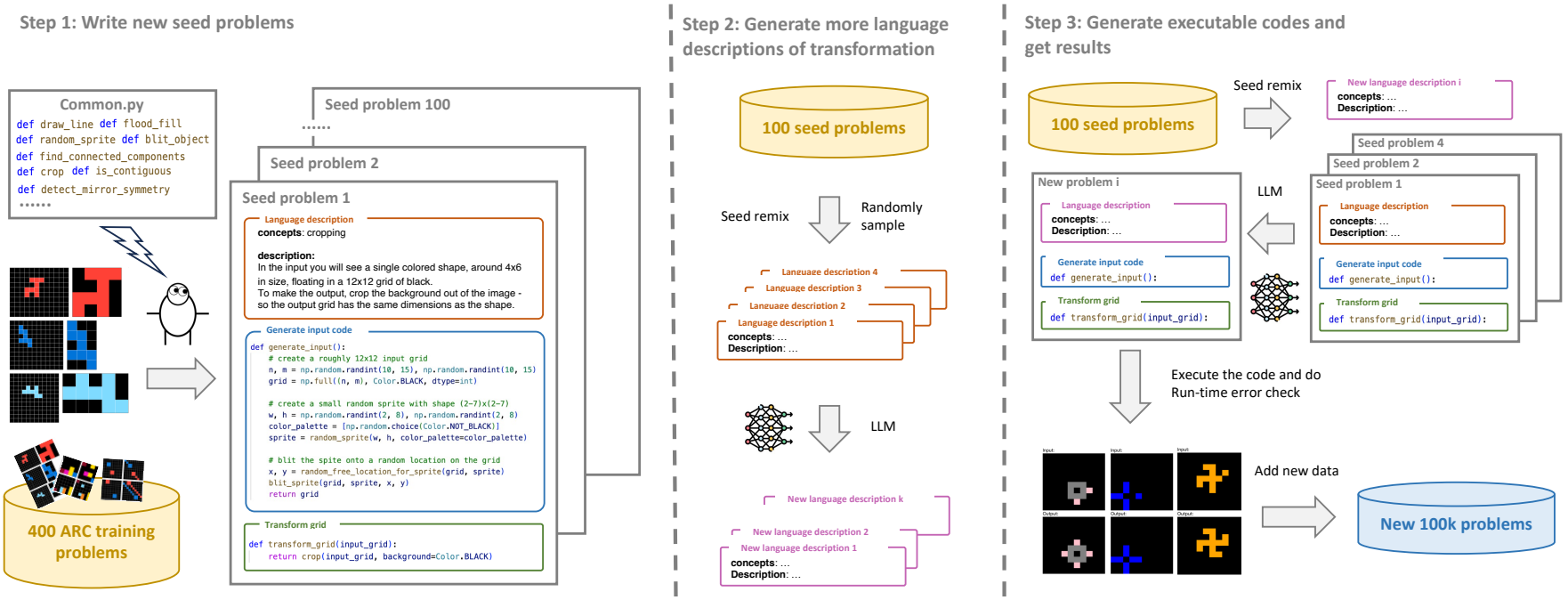
Redundant Information A piece of information is redundant, if after you remove it, the reader can easily infer it back from the rest of the figure. For instance, the words "100 objects" next to a count of "1, 2, ..., 100" is redundant, because clearly there are 100 objects evident from its count. For every piece of information, we should tentatively try to delete it, and think does the reader have sufficient information from the rest of the figure to infer it back? If they do, then you can safely remove it and the reader won't miss a thing.

Flow of Gaze Most humans read left to right, top to down. Therefore, the information being unfolded in the figure also have to flow from left to right, top to down. Figure that require a human gaze to move back and forth in a maze will be confusing. Always simplify the order of presentation of your figure, so the reader can easily consume the content in a linearized fashion.

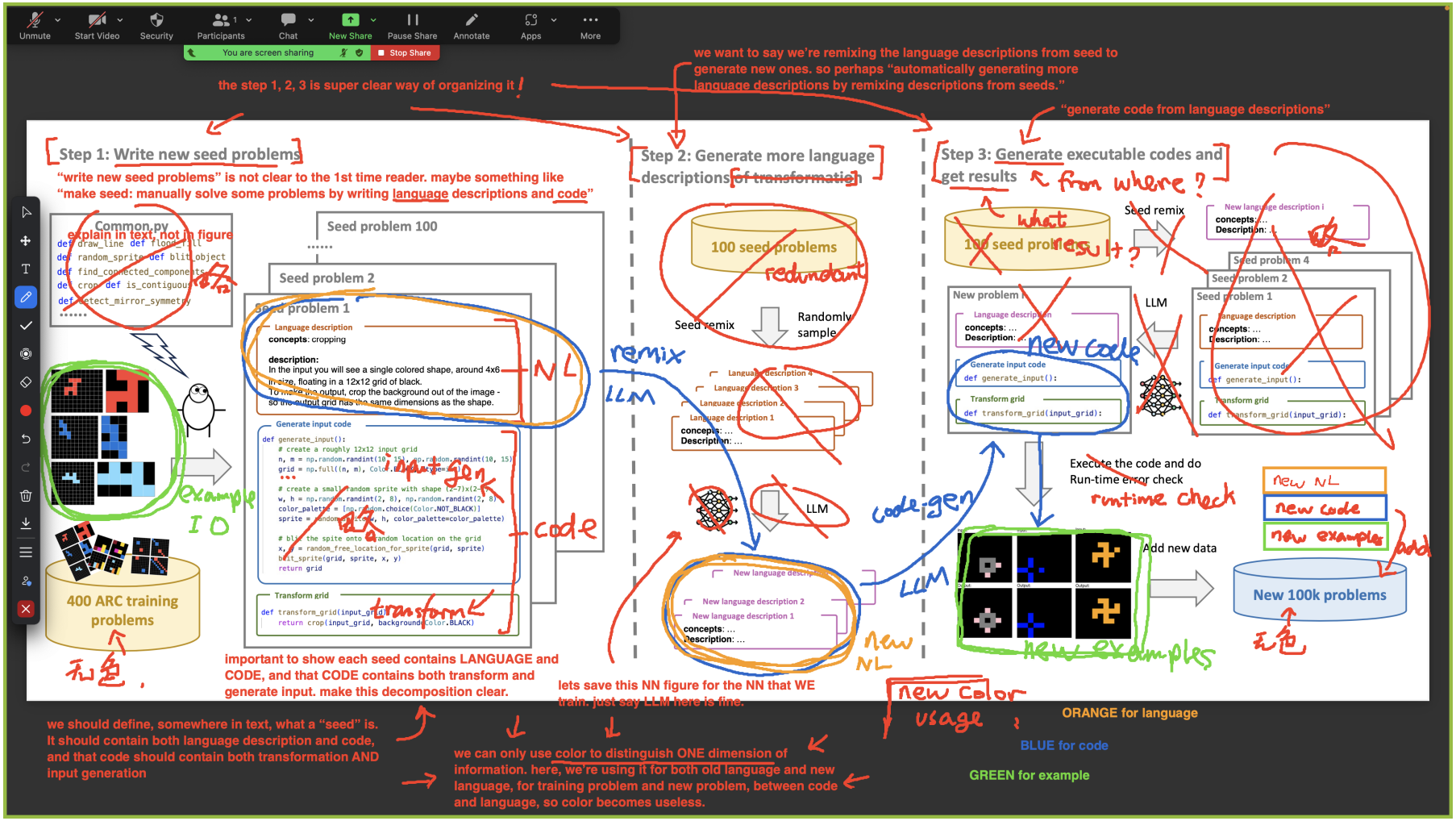
Use of Space Space is the most versatile delimiter in a figure. For the same reason we;do;not;delimit;our;sentence;using;extra;symbols;between;words, we may use space to naturally separate different objects, to suggest hierarchy — where the spacing between objects within a cluster is smaller than between clusters, and to avoid clashes of densely packed information (such as two rows of text very close by). By playing with space and alignments, we can remove many separation lines and boxes from the figure.

In Conclusion You may start with a figure with all the information you want on it. Then, simply the figure by making sure every attribute marks exactly 1 dimension of variations, figure out what part of the figure is better explained in text, remove redundant information, organize the flow, and finally, use space to organize things.

Iteration 1


Step 1: Write new seed problems
Common.py
def draw_line def flood_fill
def random_sprite def blit_object
def find_connected_components
def crop def is_contiguous
def detect_mirror_symmetry
400 ARC training problems
Seed problem 100
Seed problem 2
Seed problem 1
Language description
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Generate input code
def generate_input():
# create a roughly 12x12 input grid
n, m = np.random.randint(10, 15), np.random.randint(10, 15)
grid = np.full((n, m), Color.BLACK, dtype=int)
# create a small random sprite with shape (2-7)x(2-7)
w, h = np.random.randint(2, 8), np.random.randint(2, 8)
color_palette = [np.random.choice(Color.NOT_BLACK)]
sprite = random_sprite(w, h, color_palette=color_palette)
# blit the spite onto a random location on the grid
x, y = random_free_location_for_sprite(grid, sprite)
blit_sprite(grid, sprite, x, y)
return grid
Transform grid
def transform_grid(input_grid):
return crop(input_grid, background=Color.BLACK)
Step 2: Generate more language descriptions of transformation
100 seed problems
Seed remix
Randomly sample
Laneuage description 4
Language description 3
Laneuage description 2
Language description 1
concepts: ...
Description: ...
LLM
New language description k
New language description 2
New language description 1
concepts: ...
Description: ...
Step 3: Generate executable codes and get results
100 seed problems
Seed remix
New language description i
concepts: ...
Description: ...
Seed problem 4
Seed problem 2
Seed problem 1
Language description
concepts: ...
Description: ...
Generate input code
def generate_input():
Transform grid
def transform_grid(input_grid):
New problem i
LLM
Execute the code and do Run-time error check
Add new data
New 100k problems

Critique 1


Unmute
Start Video
Security
Participants
Chat
New Share
Pause Share
Annotate
Apps
More
You are screen sharing
Stop Share
the step 1, 2, 3 is super clear way of organizing it !
we want to say we're remixing the language descriptions from seed to generate new ones. so perhaps "automatically generating more language descriptions by remixing descriptions from seeds."
"generate code from language descriptions"
Step 1: Write new seed problems
"write new seed problems" is not clear to the 1st time reader. maybe something like "make seed: manually solve some problems by writing language descriptions and code"
Step 2: Generate more language descriptions of transformation
Step 3: Generate executable codes and get results
from where ?
what
Common.py
explain in text, not in figure
Seed problem 100
Seed problem 2
Seed problem 1
Language description
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
NL
Generate input code
def generate_input():
# create a roughly 12x12 input grid
# create a small random sprite with shape (2-7)x(2-7)
# blit the sprite onto a random location on the grid
return grid
input gen
code
Transform grid
def transform_grid(input_grid):
return crop(input_grid, background=Color.BLACK)
transform
example I O
400 ARC training problems
100 seed problems
redundant
Seed remix
Randomly sample
remix
LLM
Language description 4
Language description 3
Language description 2
Language description 1
concepts: ...
Description: ...
LLM
New language description 2
New language description 1
concepts: ...
description: ...
New NL
code gen
LLM
Seed remix
New language description i
concepts: ...
Description: ...
Seed problem 4
Seed problem 2
Seed problem 1
New problem i
Language description
concepts: ...
Description: ...
new code
Generate input code
def generate_input():
Transform grid
def transform_grid(input_grid):
LLM
Execute the code and do Run-time error check
runtime check
Add new data
new examples
New NL
new code
new example
add
New 100k problems
important to show each seed contains LANGUAGE and CODE, and that CODE contains both transform and generate input. make this decomposition clear.
lets save this NN figure for the NN that WE train. just say LLM here is fine.
we should define, somewhere in text, what a "seed" is. It should contain both language description and code, and that code should contain both transformation AND input generation
we can only use color to distinguish ONE dimension of information. here, we're using it for both old language and new language, for training problem and new problem, between code and language, so color becomes useless.
new Color usage :
ORANGE for language
BLUE for code
GREEN for example

Critique 1 cont.

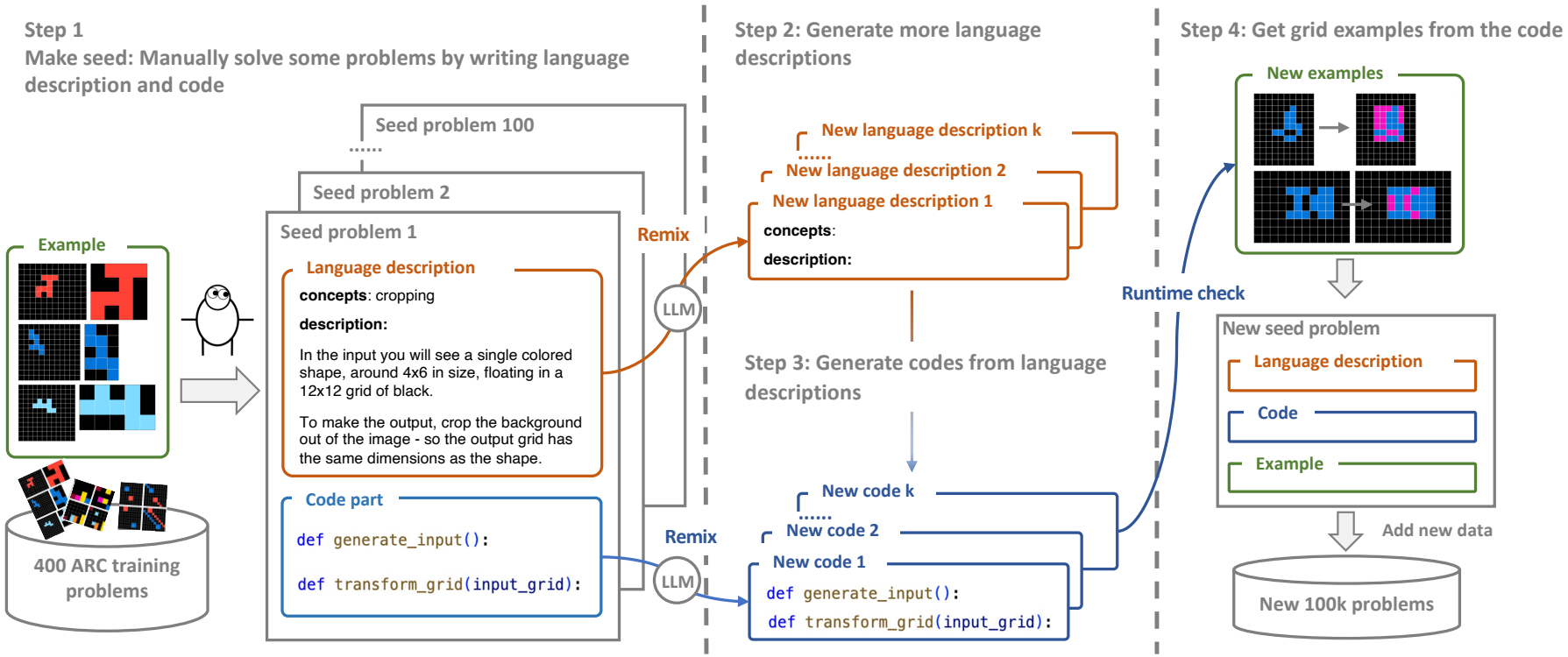
evanthebouncy Wednesday at 8:56 AM

big changes are:

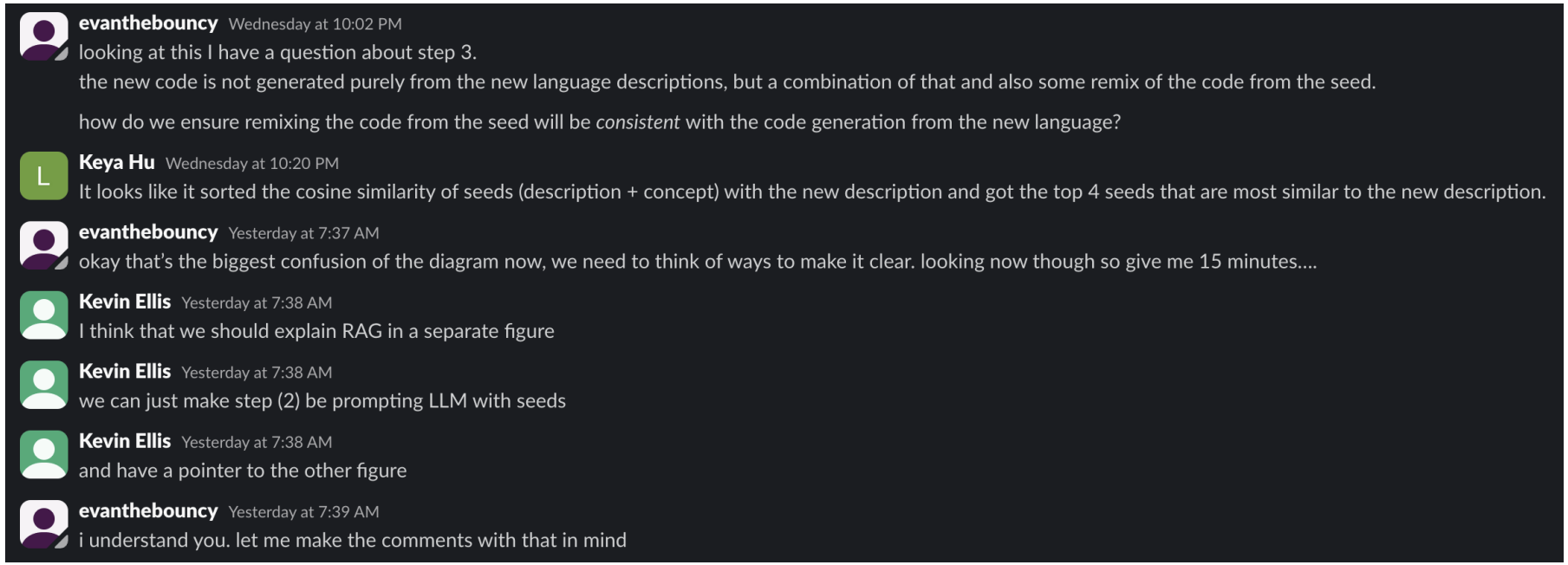
1. remove redundancies being duplicated across the figure, so it flows (with arrows) from left to right, rather than copying over what's already there
2. remove a lot of details that is better explained in the text. figure and text is complimentary, don't have to be "correct" or "complete" all the way in the figure. think about how the reader will incorporate information from figure, figure_caption, and the text itself. want the figure to contain most useful information, leaving some gaps for the caption to fill, and the rest for the text to fill in.
3. the use of COLOR is to distinguish information on ONE dimension only. color is powerful, 且惜且用！ here, the most useful place for color is on DATA TYPE, which for us is **examples**, **language**, and **code.** we only use color for those three distinctions, and nothing else.

(edited)

Iteration 2


Step 1
Make seed: Manually solve some problems by writing language description and code
Example
400 ARC training problems
Seed problem 100
Seed problem 2
Seed problem 1
Language description
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Code part
def generate_input():
def transform_grid(input_grid):
Remix
LLM
Remix
LLM
Step 2: Generate more language descriptions
New language description k
New language description 2
New language description 1
concepts:
description:
Step 3: Generate codes from language descriptions
New code k
New code 2
New code 1
def generate_input():
def transform_grid(input_grid):
Runtime check
Step 4: Get grid examples from the code
New examples
New seed problem
Language description
Code
Example
Add new data
New 100k problems

Critique 2 discussion



evanthebouncy Wednesday at 10:02 PM
looking at this I have a question about step 3.
the new code is not generated purely from the new language descriptions, but a combination of that and also some remix of the code from the seed.

how do we ensure remixing the code from the seed will be *consistent* with the code generation from the new language?

Keya Hu Wednesday at 10:20 PM
It looks like it sorted the cosine similarity of seeds (description + concept) with the new description and got the top 4 seeds that are most similar to the new description.

evanthebouncy Yesterday at 7:37 AM
okay that's the biggest confusion of the diagram now, we need to think of ways to make it clear. looking now though so give me 15 minutes....

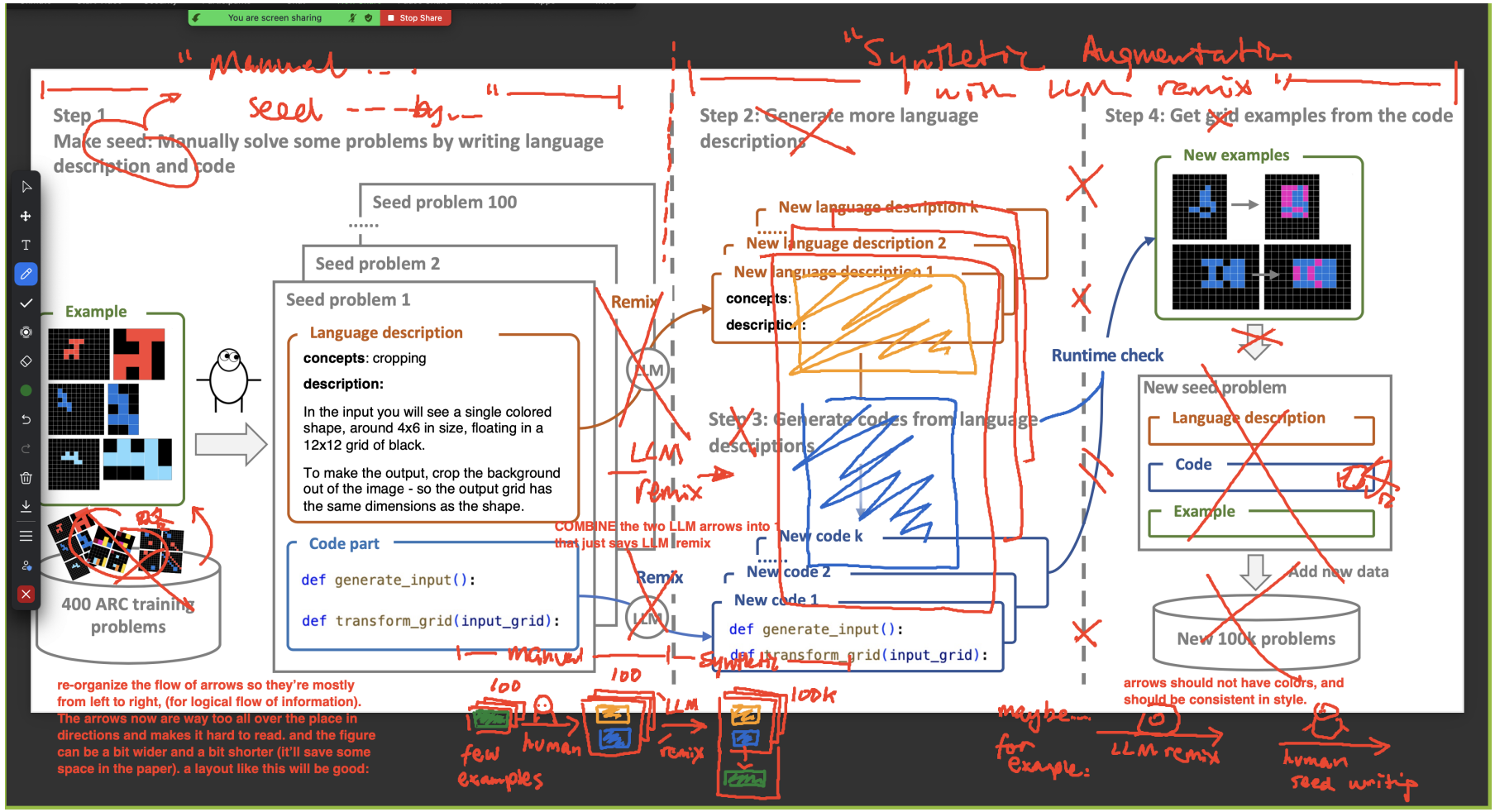
Kevin Ellis Yesterday at 7:38 AM
I think that we should explain RAG in a separate figure

Kevin Ellis Yesterday at 7:38 AM
we can just make step (2) be prompting LLM with seeds

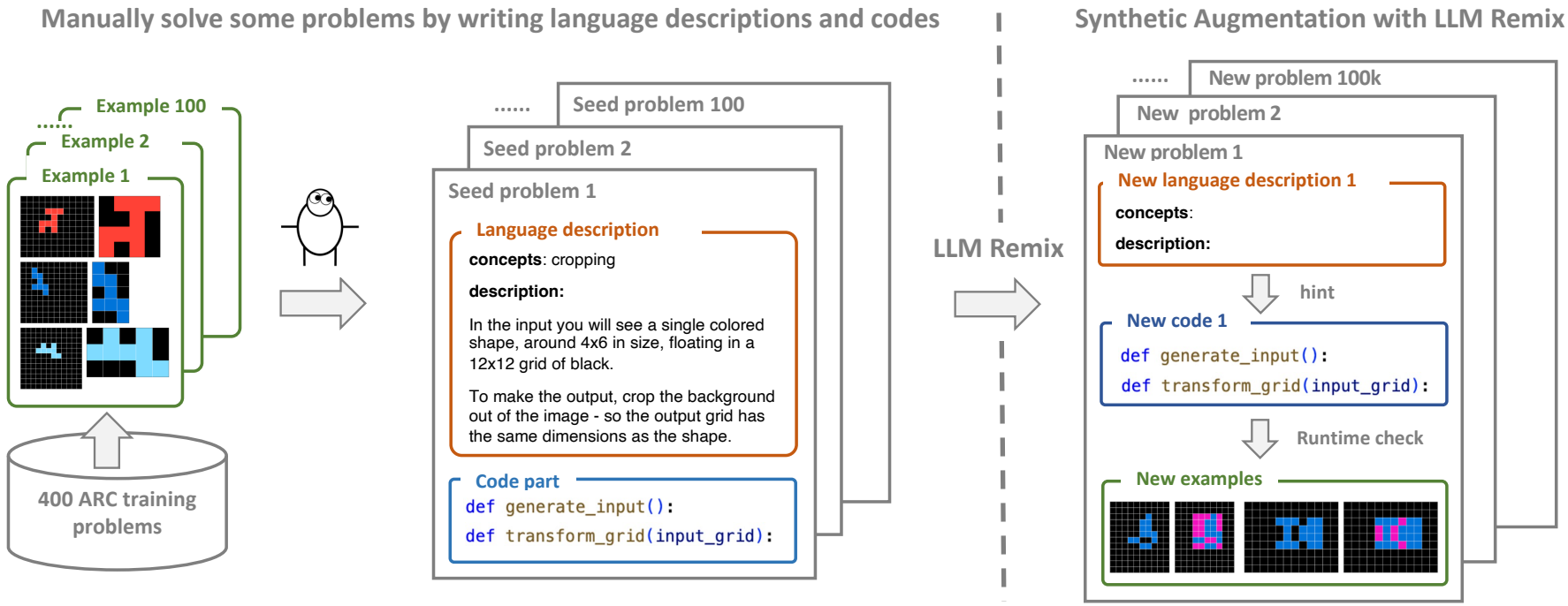
Kevin Ellis Yesterday at 7:38 AM
and have a pointer to the other figure

evanthebouncy Yesterday at 7:39 AM
i understand you. let me make the comments with that in mind

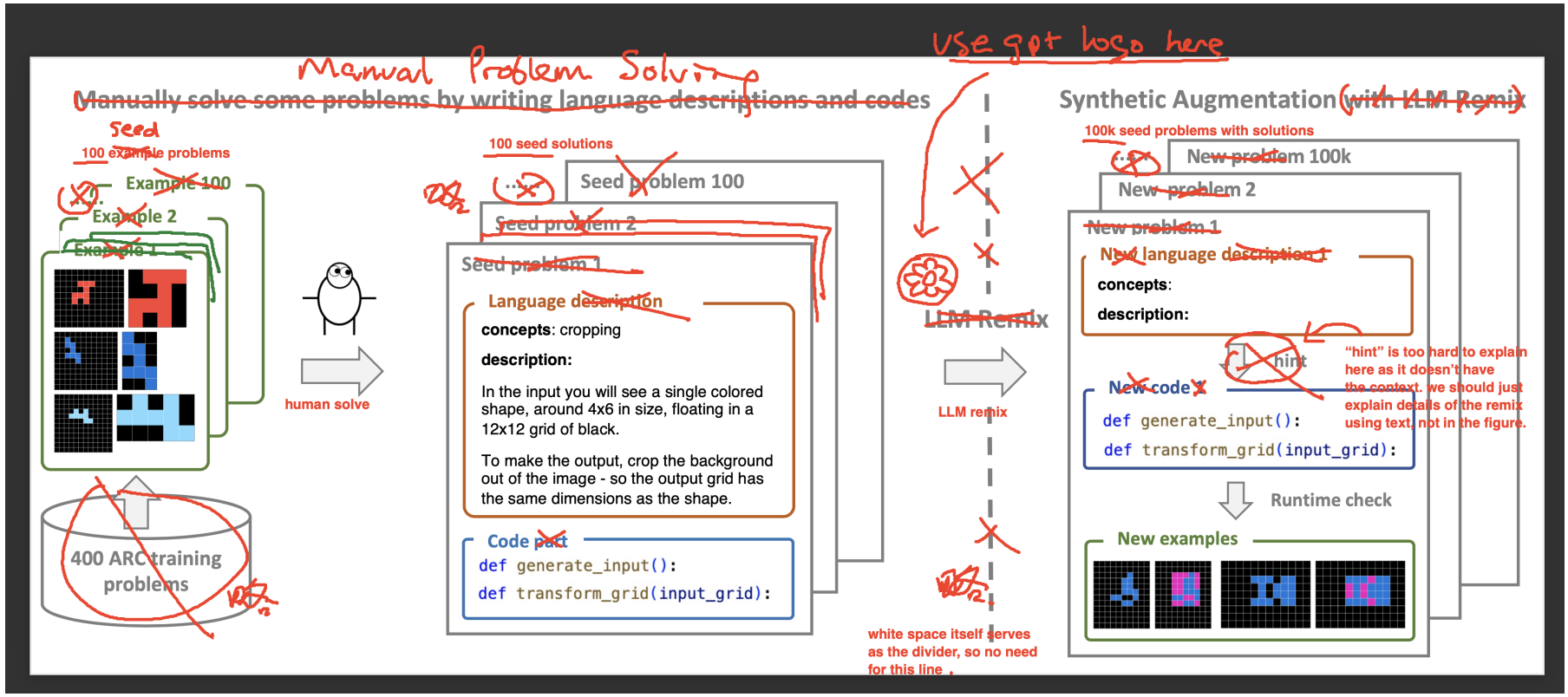
Critique 2


You are screen sharing
Stop Share
"Manual ... seed ... by ..."
"Synthetic Augmentation with LLM remix"
Step 1
Make seed: Manually solve some problems by writing language description and code
Step 2: Generate more language descriptions
Step 3: Generate codes from language descriptions
Step 4: Get grid examples from the code
Example
400 ARC training problems
Seed problem 100
Seed problem 2
Seed problem 1
Language description
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Code part
def generate_input():
def transform_grid(input_grid):
Remix
LLM
New language description k
New language description 2
New language description 1
concepts:
description:
New code k
New code 2
New code 1
def generate_input():
def transform_grid(input_grid):
COMBINE the two LLM arrows into one that just says LLM remix
Runtime check
New examples
New seed problem
Language description
Code
Example
Add new data
New 100k problems
re-organize the flow of arrows so they're mostly from left to right, (for logical flow of information). The arrows now are way too all over the place in directions and makes it hard to read. and the figure can be a bit wider and a bit shorter (it'll save some space in the paper). a layout like this will be good:
arrows should not have colors, and should be consistent in style.
few examples
human
100
100
LLM
remix
100k
maybe... for example:
LLM remix
human seed writing

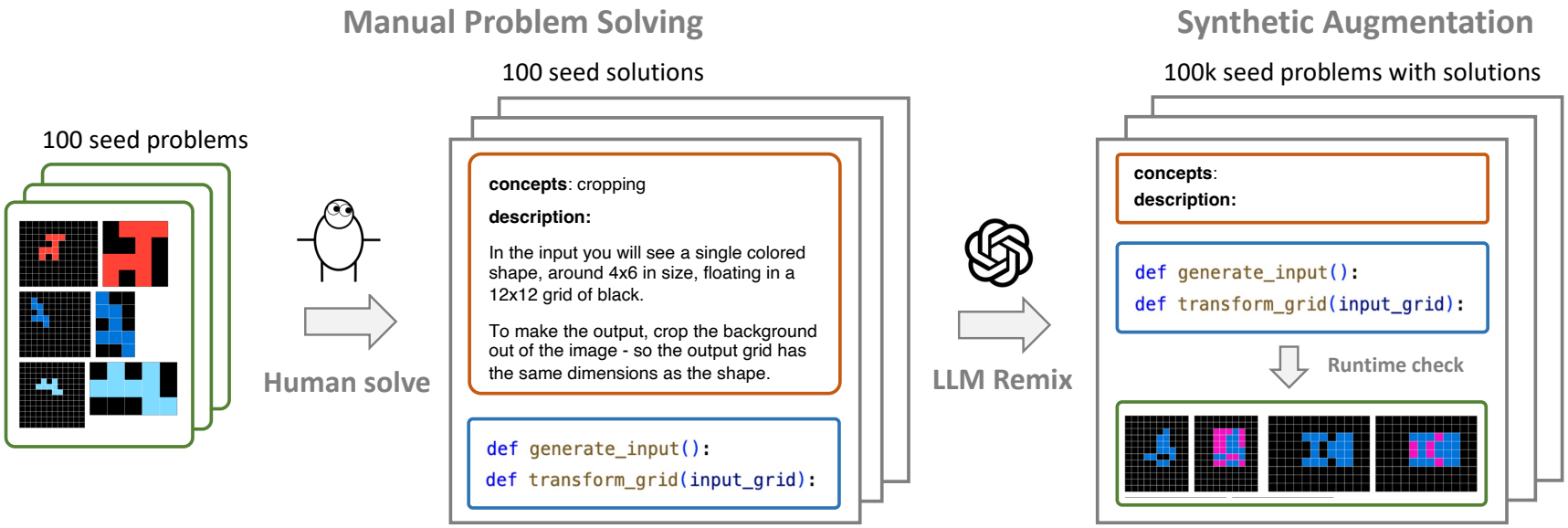
Iteration 3


Manually solve some problems by writing language descriptions and codes
Synthetic Augmentation with LLM Remix
Example 100
Example 2
Example 1
400 ARC training problems
Seed problem 100
Seed problem 2
Seed problem 1
Language description
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Code part
def generate_input():
def transform_grid(input_grid):
LLM Remix
New problem 100k
New problem 2
New problem 1
New language description 1
concepts:
description:
hint
New code 1
def generate_input():
def transform_grid(input_grid):
Runtime check
New examples

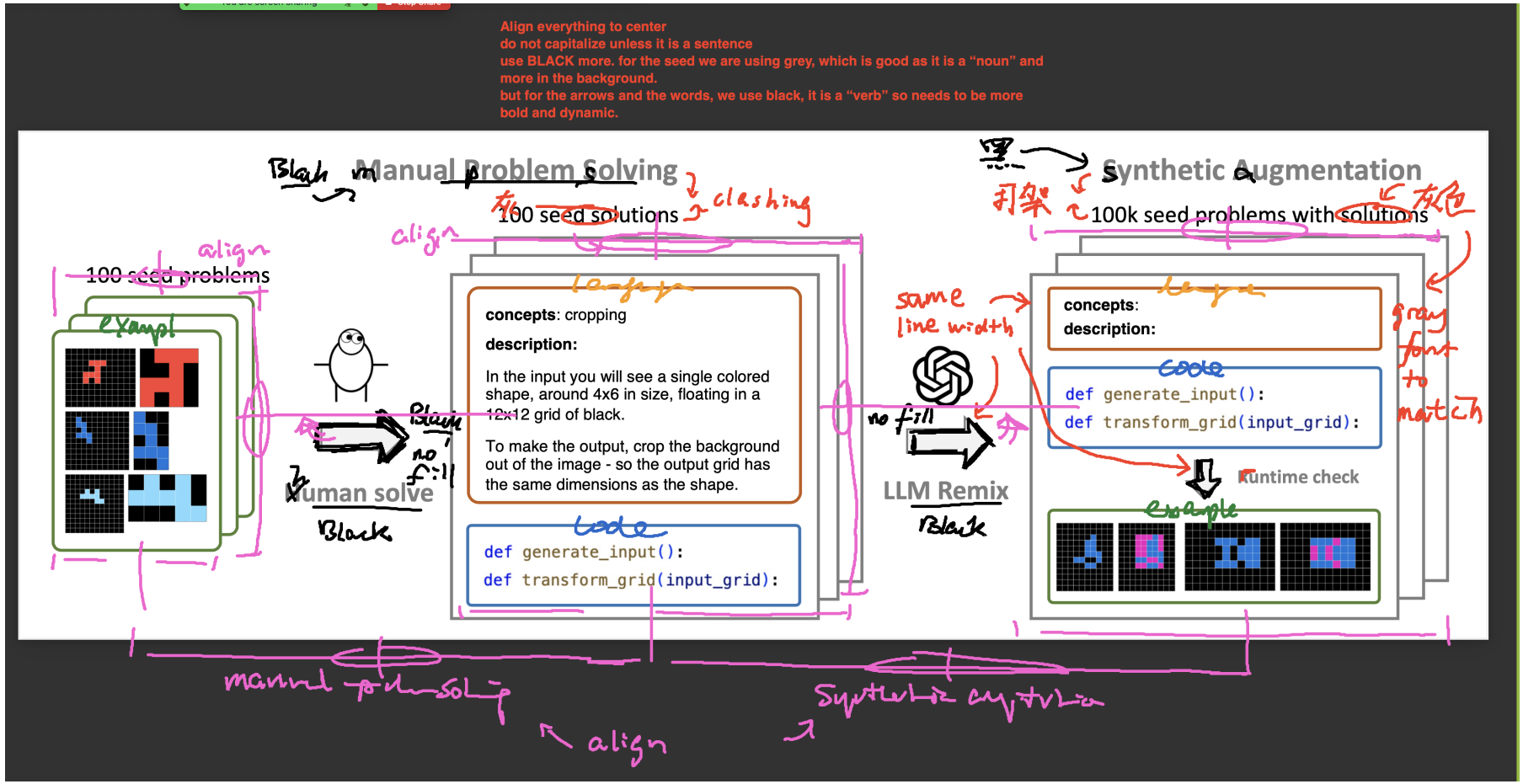
Critique 3


Use gpt logo here
Manual Problem Solving
Manually solve some problems by writing language descriptions and codes
Synthetic Augmentation (with LLM Remix)
100 example problems
Example 100
Example 2
Example 1
400 ARC training problems
human solve
100 seed solutions
Seed problem 100
Seed problem 2
Seed problem 1
Language description
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Code
def generate_input():
def transform_grid(input_grid):
LLM Remix
LLM remix
white space itself serves as the divider, so no need for this line .
100k seed problems with solutions
New problem 100k
New problem 2
New problem 1
New language description
concepts:
description:
hint
"hint" is too hard to explain here as it doesn't have the context. we should just explain details of the remix using text, not in the figure.
New code
def generate_input():
def transform_grid(input_grid):
Runtime check
New examples

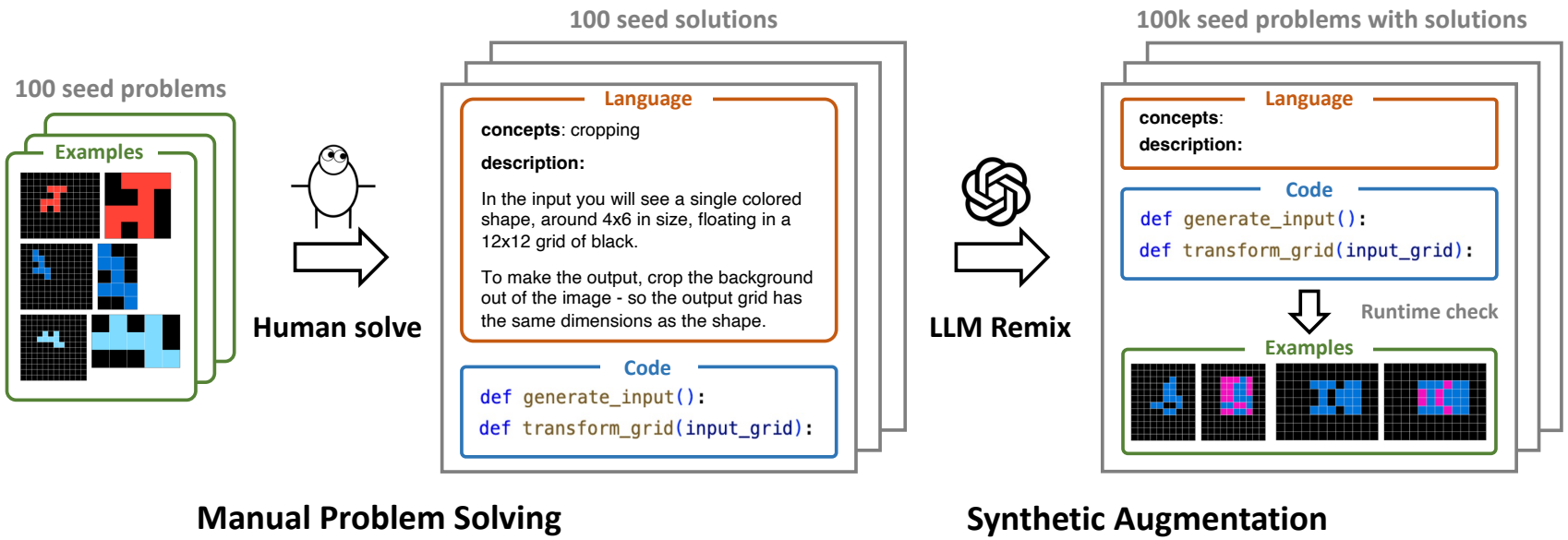
Iteration 4


Manual Problem Solving
Synthetic Augmentation
100 seed problems
100 seed solutions
100k seed problems with solutions
Human solve
LLM Remix
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
def generate_input():
def transform_grid(input_grid):
concepts:
description:
def generate_input():
def transform_grid(input_grid):
Runtime check

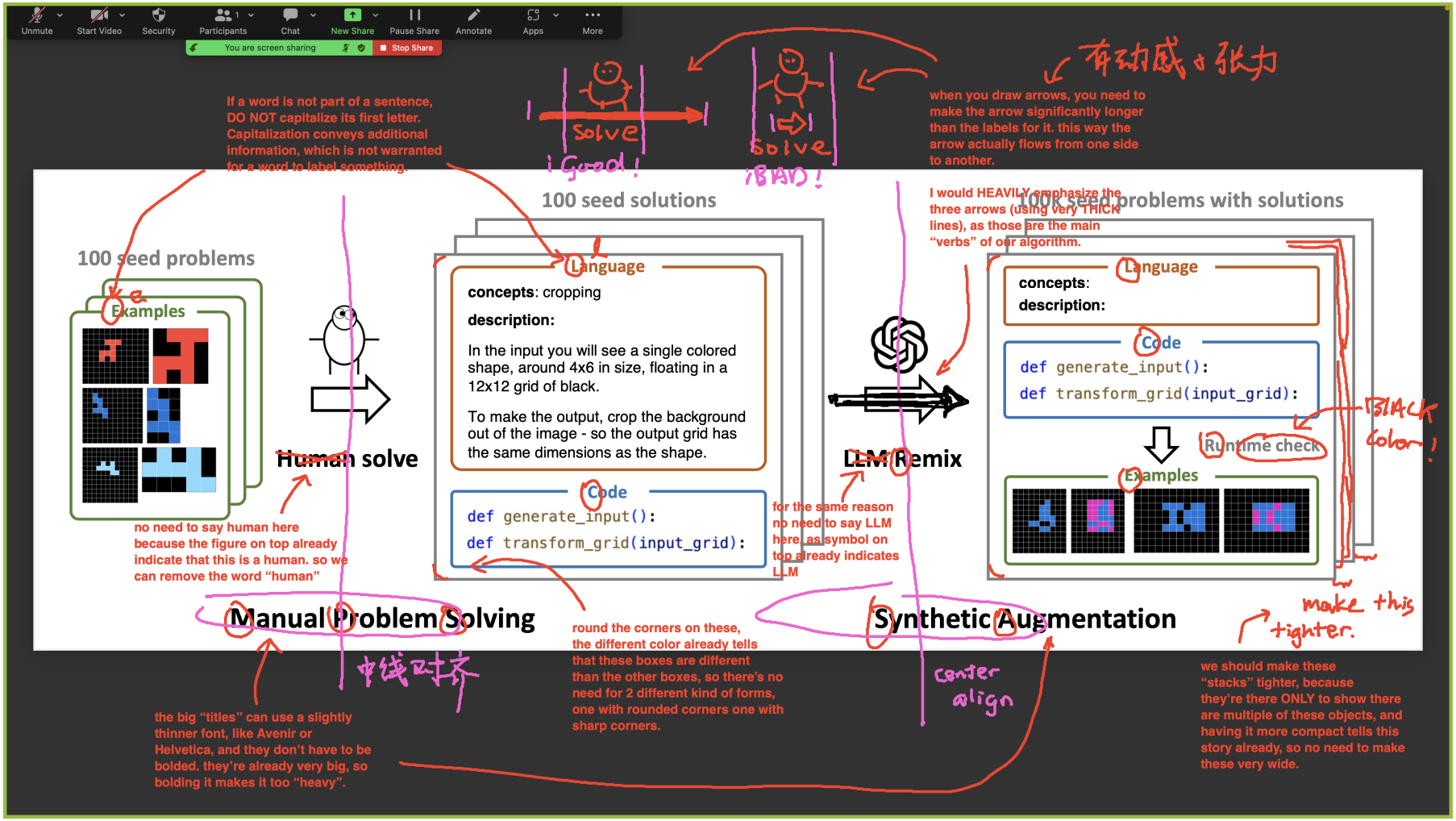
Critique 4


Align everything to center
do not capitalize unless it is a sentence
use BLACK more. for the seed we are using grey, which is good as it is a "noun" and more in the background.
but for the arrows and the words, we use black, it is a "verb" so needs to be more bold and dynamic.
Manual Problem Solving
100 seed solutions
clashing
align
100 seed problems
Example
Human solve
Black
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
code
def generate_input():
def transform_grid(input_grid):
Synthetic Augmentation
100k seed problems with solutions
same line width
no fill
LLM Remix
Black
concepts:
description:
code
def generate_input():
def transform_grid(input_grid):
Runtime check
example
gray font to match
align

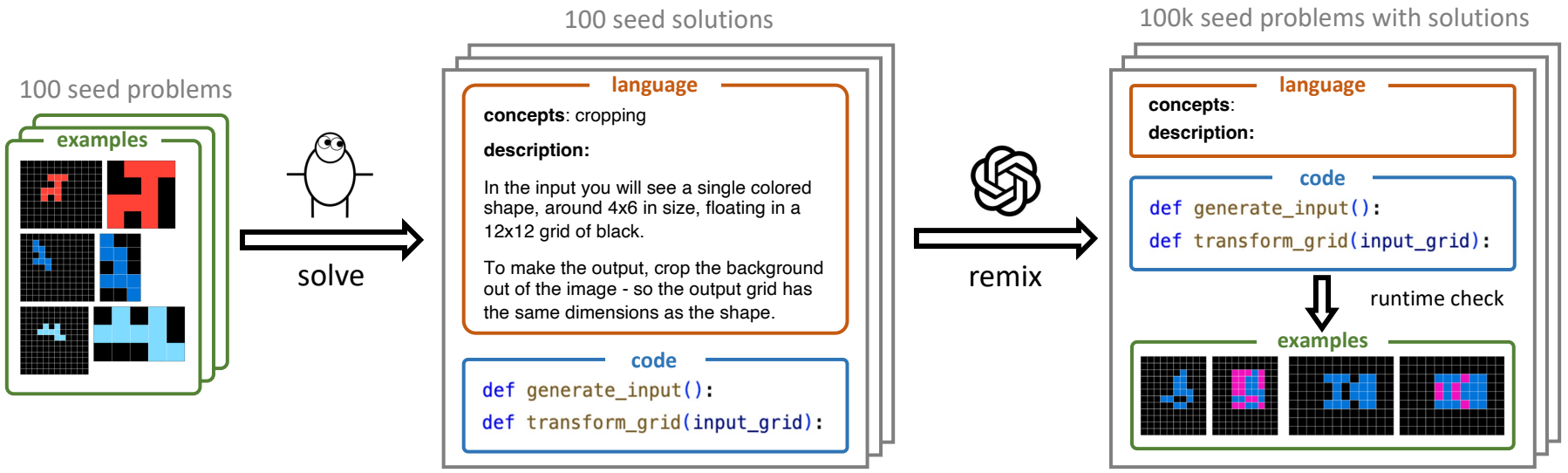
Iteration 5


100 seed problems
Examples
Human solve
100 seed solutions
Language
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Code
def generate_input():
def transform_grid(input_grid):
LLM Remix
100k seed problems with solutions
Language
concepts:
description:
Code
def generate_input():
def transform_grid(input_grid):
Runtime check
Examples
Manual Problem Solving
Synthetic Augmentation

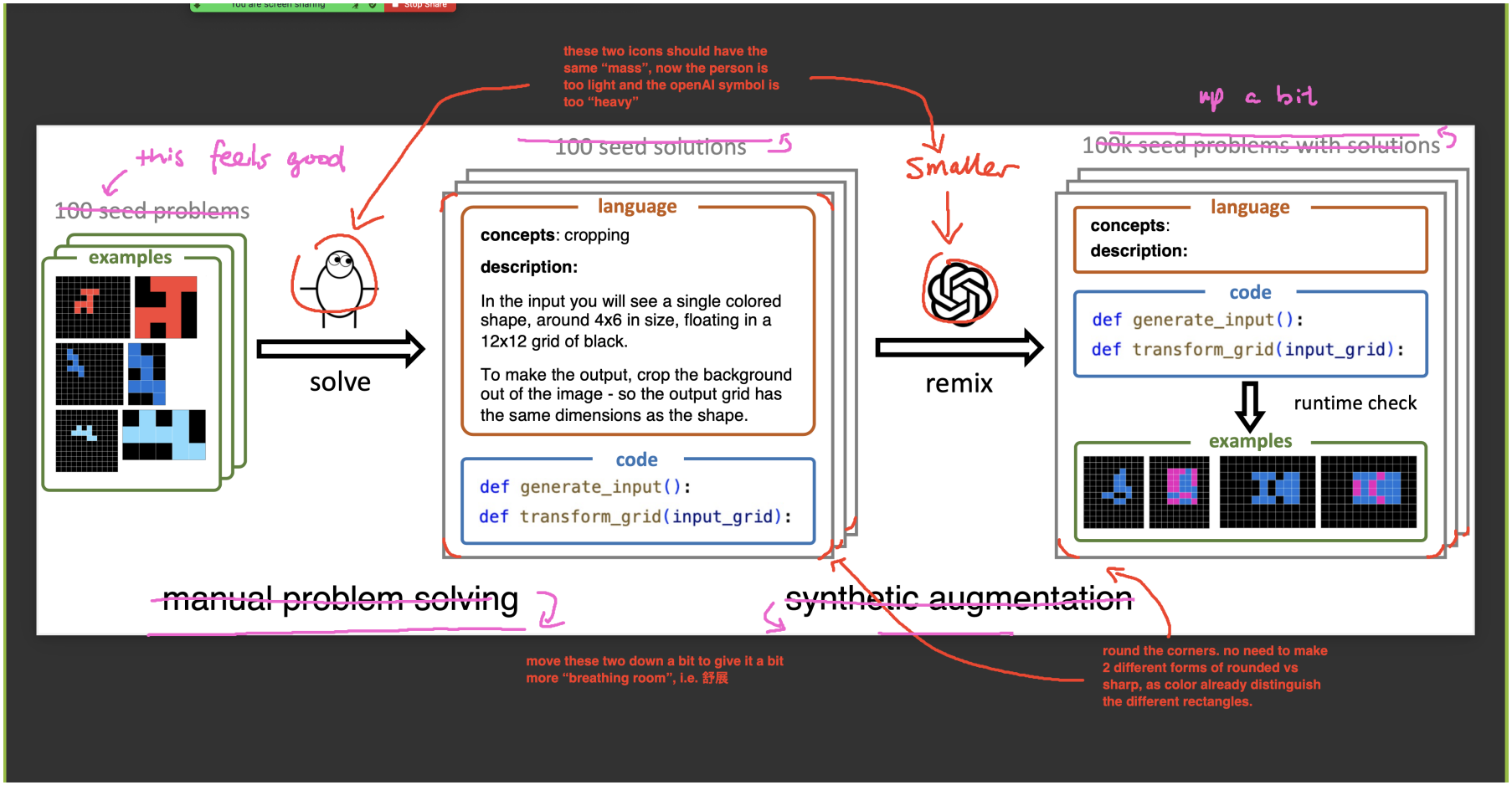
Critique 5


Unmute
Start Video
Security
Participants
Chat
New Share
Pause Share
Annotate
Apps
More
You are screen sharing
Stop Share
If a word is not part of a sentence, DO NOT capitalize its first letter. Capitalization conveys additional information, which is not warranted for a word to label something.
Solve
Good!
Solve
BAD!
when you draw arrows, you need to make the arrow significantly longer than the labels for it. this way the arrow actually flows from one side to another.
100 seed problems
Examples
Human solve
no need to say human here because the figure on top already indicate that this is a human. so we can remove the word "human"
100 seed solutions
Language
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
Code
def generate_input():
def transform_grid(input_grid):
LLM Remix
for the same reason no need to say LLM here, as symbol on top already indicates LLM
I would HEAVILY emphasize the three arrows (using very THICK lines), as those are the main "verbs" of our algorithm.
100k seed problems with solutions
Language
concepts:
description:
Code
def generate_input():
def transform_grid(input_grid):
Runtime check
BLACK color!
Examples
make this tighter.
Manual Problem Solving
Synthetic Augmentation
round the corners on these, the different color already tells that these boxes are different than the other boxes, so there's no need for 2 different kind of forms, one with rounded corners one with sharp corners.
the big "titles" can use a slightly thinner font, like Avenir or Helvetica, and they don't have to be bolded. they're already very big, so bolding it makes it too "heavy".
we should make these "stacks" tighter, because they're there ONLY to show there are multiple of these objects, and having it more compact tells this story already, so no need to make these very wide.

Iteration 6


100 seed problems
examples
solve
100 seed solutions
language
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
code
def generate_input():
def transform_grid(input_grid):
remix
100k seed problems with solutions
language
concepts:
description:
code
def generate_input():
def transform_grid(input_grid):
runtime check
examples
manual problem solving
synthetic augmentation

Critique 6


these two icons should have the same "mass", now the person is too light and the openAI symbol is too "heavy"
up a bit
this feels good
100 seed problems
examples
solve
100 seed solutions
language
concepts: cropping
description:
In the input you will see a single colored shape, around 4x6 in size, floating in a 12x12 grid of black.
To make the output, crop the background out of the image - so the output grid has the same dimensions as the shape.
code
def generate_input():
def transform_grid(input_grid):
smaller
remix
100k seed problems with solutions
language
concepts:
description:
code
def generate_input():
def transform_grid(input_grid):
runtime check
examples
manual problem solving
synthetic augmentation
move these two down a bit to give it a bit more "breathing room", i.e. 舒展
round the corners. no need to make 2 different forms of rounded vs sharp, as color already distinguish the different rectangles.

Iteration 7, final