LLM Fine-tuning (Workflow)

written by DaltonRuer | October 31, 2023

Prerequisites to the Why

You can't start a post by handing out work to be done before you readers can read it!

I would like to follow that convention, however, I find myself in a place where you really need a little background to truly understand what this post represents.

Assignment 1:

Read this Linked In article I wrote recently: Focused Generative AI

Assignment 2:

With those thoughts in mind, which are more than likely still very vague please watch at least the first few minutes of the most hilarious, yet meaningful, video I've ever produced. I need you to specifically understand the various personas as they relate to Qlik Application Automation. {Yes this post is about fine-tuning an LLM, but trust me you need this.}

Why

In my initial post on Generative AI was how I was able to fine-tune an LLM to generate Qlik Dork Script code. The goal was to help business users or developers who didn't know my

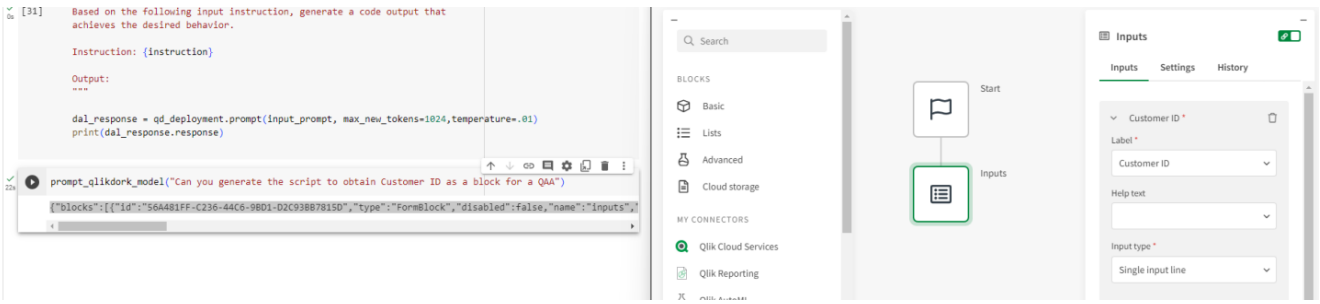
fictious programming language. They could ask in natural language and be handed a single line of code. So let's jump into the "why" for this post: "Can an LLM be fine-tuned in order to help users get started with automation tasks in Qlik Application Automation?" If you think about the images I shared via the Linked In post imagine for a second that developers certainly know how to select the right type of connector, and know how to select the right blocks. But the business users at the far left of the scale, likely would not. Hence, this attempt.

Dipping my toes in

Having succeeded at generating one line of code, I started my fine-tuning by attempting to generate 1 single block for QAA using the Predibase declarative ML platform. Notice in the output it's just 1 row, but a row of complicated JSON text not just simple words like "QDSum(Field)."

SUCCESS.

That just looks like gobbly-gook. I get it. It's the raw JSON that represents and input block for a Qlik Application Automation. If I wanted to try and eat the whole elephant I would worry about having to build a chat interface and embedding it. But I'm not trying to eat the whole elephant, I'm taking a bite. If I copy the JSON code, and simply paste it into my automation, voila the magic happens.

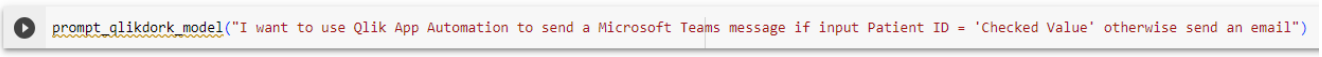


Guess I'm done testing, the article is over, and you can go back to your real work. Nah, I kept prompting in other ways. More complicated ways.

Diving in

One of the obstructions I mentioned in the Linked In article was that of business users not knowing to organize things. Qlik Application Automations involve a workflow, not just single blocks. So, naturally my real objective for this post was to see whether or not I could train an LLM to … generate a workflow with multiple objects that work together in a pattern. Could I really remove that obstacle from the path?

Would this actually work?



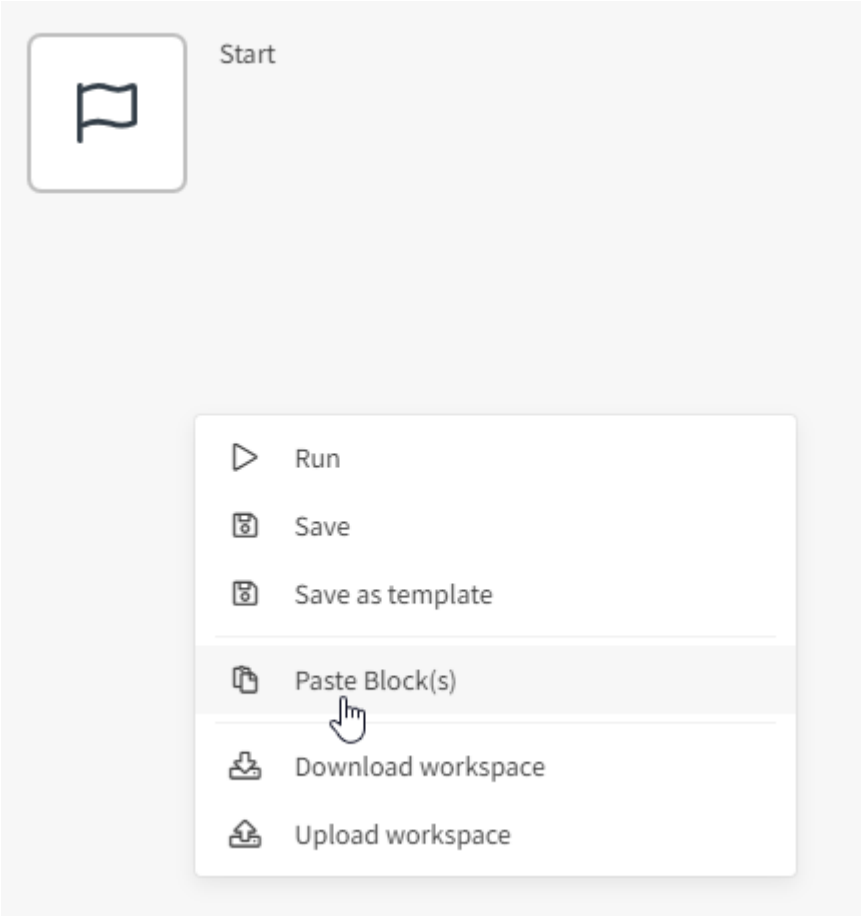
I needed it to generate all this:

> *{"blocks":[{"id":"56A481FF-C236-44C6-9BD1-D2C93BB7815D","type":"FormBlock","disabled":false,"name":"inputs","displayName":"Inputs","comment":"","childId":"1659B3FC-D3A3-4CC0-AF0E-D55E44DFD0B6","inputs":[],"settings":[{"id":"persist_data","value":"no","type":"select","structure":{}},{"id":"automations_censor_data","value":false,"type":"checkbox","structure":{}}],"collapsed":[{"name":"loop","isCollapsed":false}],"x":290,"y":140,"form":[{"id":"inputs-input-0","label":"ContactID","helpText":null,"type":"input","values":null,"isRequired":true,"options":{},"order":0},{"id":"inputs-input-1","label":"Contact Name","helpText":null,"type":"input","values":null,"isRequired":true,"options":{},"order":1}],"persistData":"no"},{"id*

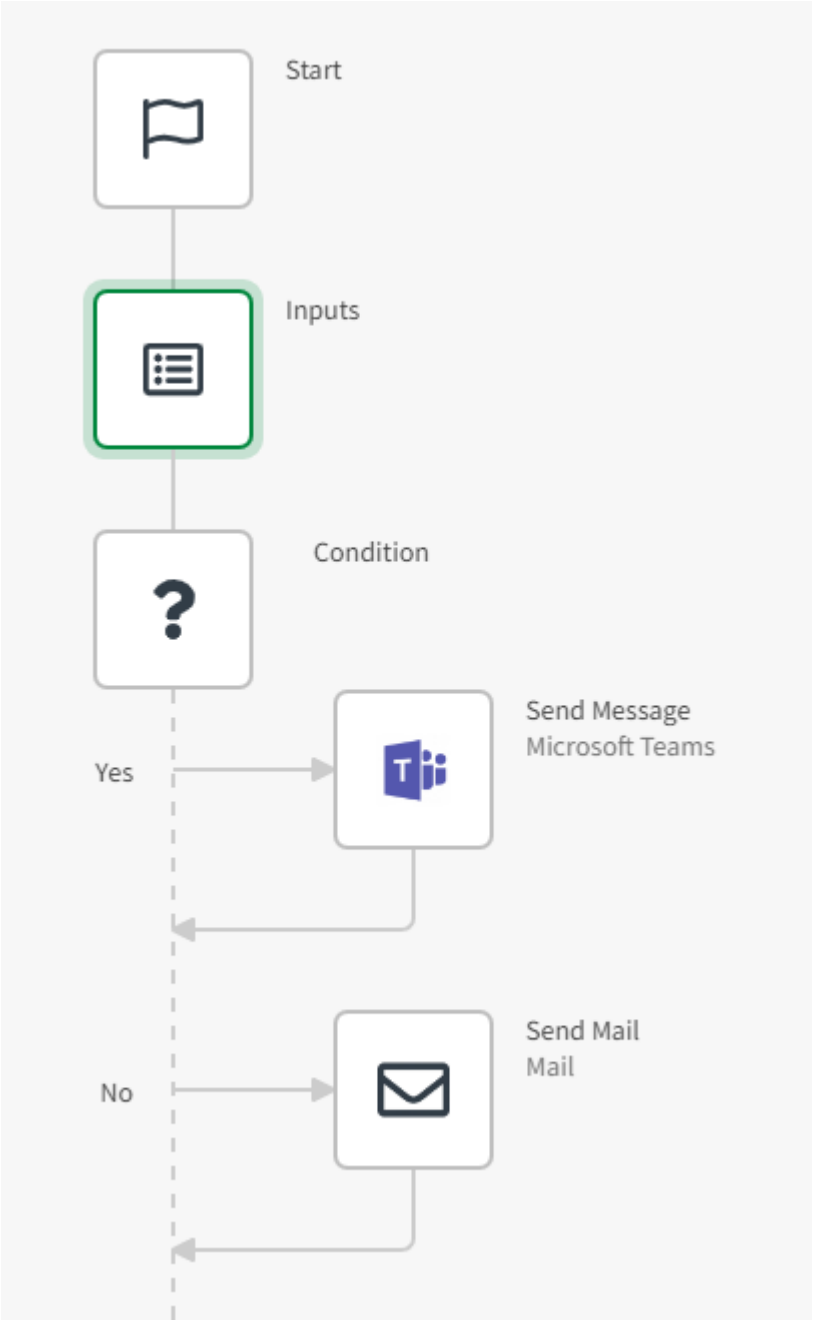
```
":"1659B3FC-D3A3-4CC0-AF0E-
D55E44DFD0B6","type":"IfElseBlock","disabled":false,"name":
"condition","displayName":"Condition","comment":"","childId
":null,"inputs":[{"id":"conditions","value":{"mode":"all","
conditions":[{"input1":"{$.inputs.ContactID}","operator":"c
ontain","input2":"'Qlik
Dork'"}]},"type":"custom","structure":{}}],"settings":[],"c
ollapsed":[{"name":"both","isCollapsed":false},{"name":"yes
","isCollapsed":false},{"name":"no","isCollapsed":false}],"
x":-370,"y":133,"childTrueId":"5DF8A4AD-DE24-4CD3-B090-
AF27D0CD03E1","childFalseId":"C771B857-EF1B-4B54-
A842-9B5E1C3DF990"},{"id":"5DF8A4AD-DE24-4CD3-B090-
AF27D0CD03E1","type":"EndpointBlock","disabled":false,"name
":"sendMessage","displayName":"Microsoft Teams – Send
Message","comment":"","childId":null,"inputs":[{"id":"ce80c
8c0-d33a-11eb-a1be-
fb85dcd48977","value":null,"type":"string","structure":{}},
{"id":"ce8be4d0-d33a-11eb-
aa74-37bc58fc7293","value":null,"type":"string","structure"
:{}},{"id":"ce9788f0-d33a-11eb-
a754-31a1e548ae07","value":null,"type":"string","structure"
:{}},{"id":"cebb4240-d33a-11eb-86c4-
cf0a4f744ace","value":null,"type":"string","structure":{}}]
,"settings":[{"id":"datasource","value":null,"type":"select
","structure":{}},{"id":"blendr_on_error","value":"stop","t
ype":"select","structure":{}},{"id":"automations_censor_dat
a","value":false,"type":"checkbox","structure":{}}],"collap
sed":[{"name":"loop","isCollapsed":false}],"x":-367,"y":266
,"datasourcetype_guid":"0d87af8f-27c0-11ea-921c-022e6b5ea1e
2","endpoint_guid":"ce5ffe30-d33a-11eb-
b284-3955388e8c59","endpoint_role":"create"},{"id":"C771B85
7-EF1B-4B54-
A842-9B5E1C3DF990","type":"EndpointBlock","disabled":false,
"name":"SendMail","displayName":"Mail – Send
Mail","comment":"","childId":null,"inputs":[{"id":"442845a0
-cd00-11eb-9aa5-
b966a68573b7","value":null,"type":"string","structure":{}},
```

{“id”:”543f08c0-cd00-11eb-b9c6-af8e23cc2439″,”value”:null,”type”:”string”,”structure”:{}},{“id”:”5cbdb1d0-cd00-11eb-8afd-ddd4ecc002ac”,”value”:null,”type”:”string”,”structure”:{}},{“id”:”72cad1e0-cd00-11eb-aa1b-3f42c1d22e38″,”value”:null,”type”:”string”,”structure”:{}},{“id”:”79af8c50-cd24-11eb-a019-dfdc9a7f7317″,”value”:null,”type”:”select”,”structure”:{}},{“id”:”7d225d00-cd00-11eb-8493-055760e8b45a”,”value”:null,”type”:”longtext”,”structure”:{}},{“id”:”bb841f10-3341-11ec-b9a3-9d6de0bb7866″,”value”:null,”type”:”custom”,”structure”:{}}],”settings”:[{“id”:”datasource”,”value”:null,”type”:”select”,”structure”:{}},{“id”:”blendr_on_error”,”value”:”stop”,”type”:”select”,”structure”:{}},{“id”:”automations_censor_data”,”value”:false,”type”:”checkbox”,”structure”:{}}],”collapsed”:[{“name”:”loop”,”isCollapsed”:false}],”x”:-367,”y”:9,”datasourcetype_guid”:”05407a80-ccfd-11eb-b098-7140f6ac27f4″,”endpoint_guid”:”84a79bf0-ccff-11eb-a508-7d415a725ea7″,”endpoint_role”:”create”}],”variables”:[]}

So that when I did this:



It would do this:



Unfortunately my friends I hit the wall on my first, second, third and fourth attempts. While I ignored the Predibase model training outputs for my first tests, I started paying attention to them, because obviously something wasn't right. My training wasn't working.

Well talk about gobbly-gook the training status was speaking in words I didn't understand. It was right there in front of me, like that JSON code was in front of you, and it made no sense. I like bleu cheese, but what did it have to do with machine learning? What's perplexity. More importantly why were

my next tokens lower than my regular tokens? Or should they both be lower?

```
Created model repository: <Qlik App Automation>
Check Status of Model Training Here: https://app.predibase.com/models/version/4655
Monitoring status of model training...
Compute summary:
  * GPU: T4 (x1)
✓ Queued 00:08:49
✓ Preprocessing 00:09:08
```

epochs	time	feature	metric	train	val	test
0	01:07:59	combined	loss	0.4693		3.0104
		output	bleu	0		0
			loss	0.4693		3.0104
			next_token_perplexity	13680.83…		18209.83…
			perplexity	31840.97…		31837.69…
			word_error_rate	23.4823		18.7091
1	02:07:55	combined	loss	0.0008		2.9894
		output	bleu	0		0
			loss	0.0008		2.9894
			next_token_perplexity	11782.65…		18204.96…
			perplexity	31852.63…		31836.96…
			word_error_rate	24.3143		18.5273
2	03:07:54	combined	loss	0.0007		2.9936
		output	bleu	0		0
			loss	0.0007		2.9936
			next_token_perplexity	11780.75…		18205.43…
			perplexity	31852.57…		31837.63…
			word_error_rate	24.7518		18.5273

```
✓ Evaluating 05:17:30
✓ Visualizing 05:47:31
Ready
```

Rookie mistakes

As you can imagine after digging in so that I could understand the values, I realized the numbers were horrible. Basically told me “based on the data you are trying to train with, you could flip a coin 100 times and it will never land on the heads you want it to.” I broke a cardinal rule of utilizing output from Generative AI.

Like in the previous post I asked for alternative ways to ask my questions because I knew I needed a lot of variety. Several of the many ways involved phrases like this “Get the input for Field1 and if ***it*** =” which is completely reasonable english. That blindly started using. However, as I used my Qlik Sense application to replace Field1 and Field2 everywhere in the code, and to replace the value ***it*** was supposed to equal for

condition ***it*** remained by itself. Which was confusing for the training. I needed “If Field1 = ” or “If CustomerID =” or “If PATIENT ID = .”

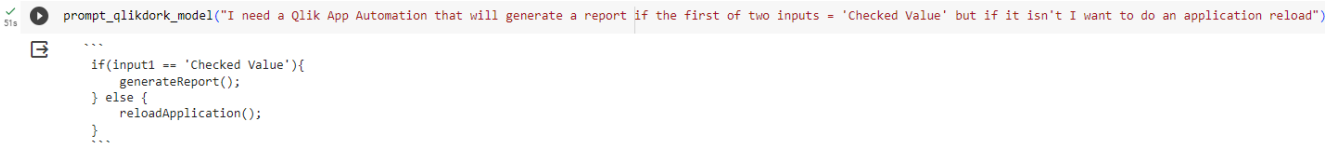
Worse than that. I’m a shame to even admit this rookie mistake I had copied rows repeatedly and then change the wording for how the question was phrased. Well it turns out that I missed an awful lot of rows and never changed them. And simply compounded the problem trying to use the Replace function to replace “Qlik app automationn” with “Qlik Application Automation” and “QAA” etc. Notice that I had 2 n’s at the end of automation.” It never found that in my sample rows and thus never replaced them. Woohoo more duplicated instructions.

So although the output from my load said that I had 2,000+ rows, the truth was that a ton of them were duplicates. As the Qlik Dork, you would think I would have remembered a very basic rule when loading data, and especially generating data, use the Data Model Viewer to see how may DISTINCT values there actually were.

After the first training I knew full well that the 393 distinct rows that got output were not enough to train anything. Especially for a highly bespoke task like I was asking.

Output is only as good as the input?

The plethora of rookie mistakes impacting the quality and uniqueness of the training data I kept applying, yielded hallucinations over and over. A reminder of exactly why I had used Qlik Dork Script to test in my first blog. I knew full well it had plenty of other coding languages to draw from. I wasn’t trying to simply enhance those. I was attempting to create a completely bespoke coding model for a very targeted, never been before trained task.



After correcting my rookie mistakes my jaw hit the floor. JSON formatted code that could be copied and pasted into Qlik Application Automation. Before determining how impressed to be … be sure and notice that GUID at the front of the block.

My training involved using the same basic block of code for the target over and over. In other words, code that had the same GUID in it over and over and over. If you scroll back to what I showed was the full expected output, it is NOT the same GUID. The training data involved the same starting GUID, but the rest of the script has others. During the training, unlike the key words and syntax like “{“blocks” : [{“id”:” it recognized the pattern was indeed a GUID and each time I prompted it was able to generate new GUIDS, and yet utilized the same new GUIDS where needed within other blocks.

Why so important? Because my goal was just to get it to reproduce the syntax needed and fill in the field names and values I entered in the prompt. Ensuring unique GUID’s if I prompted again and again was going to be project for a later date. (Or someone much smarter than myself to resolve.) So I

Lessons Learned

While I was thrilled beyond measure that I was able to generate a single block, obviously I was left feeling sorry for myself that I couldn’t remove that next obstacle. Hadn’t I learned anything in post one? Why was I producing hallucinations on multiple training attempts? But as I shared in my About and my first post: I am going to be writing about my learning process. My growth.

1. Last week I didn’t even know what “bleu” or “perplexity” were. Now I do.
2. Repeated all the other learning lessons from my first post. Especially that I will fail and that’s ok.
3. It’s humbling at times when you are so heads down on the really hard parts, to realize you are making rookie mistakes with the data.
4. That the quality of the data you are trying to utilize for training, can be complicated and you better ensure it is clean and unique and would make sense to a human. So while it’s easy to produce thousands of rows of synthetic data to feed AI, you should take the time to validate it first.
5. Unlike my simple training for “QDSum(Field1” the training models for these very large blocks of code took between 17 and 24 hours. Of all of the lessons learned for this complexity, patience is perhaps the lesson I had to learn most.

Recognition

I would like to recognize, and thank, my friend Christian Eckler for inspiring me through his work with Qlik partner ancoreSoft. If you care to watch my Dork Cast with him, you will see that he generates blocks/flows for Qlik Application Automation that can be easily pasted in. Until he showed me that QAA was so open and just used JSON structures I didn’t even know it. And certainly wouldn’t have thought to use that as my second goal. I may get back to my original goal, after I made this work. In the video Christian’s code produces all of the values for all of the needed parameters for each of the blocks. If you think back to my list of obstacles, do you think even understanding the parameters might be an issue for certain personas? Will I ever be able to train the model enough to infer certain values form the text prompted by a business user? Honestly I have no idea. But it seems like I have even more work to do. And this wasn’t even my desired

next step. So much to learn.