# How To Evaluate MLOps Tools

27 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=GHK5HMW4XMA
https://www.youtube.com/watch?v=GHk5HMW4XMA

SUMMARY

*The talk focuses on evaluating ML Ops tools, particularly TensorFlow Extended (TFX), through the speaker's extensive experience in building ML tooling across various companies. The speaker emphasizes the importance of having a framework for evaluating tools to avoid becoming overly attached to specific technologies, which can lead to blind spots and hinder effective data science practices.*

- *The speaker has a diverse background in ML tooling, having worked with companies like Airbnb and GitHub.*
- *The talk was motivated by misconceptions about the applicability of PyTorch and the effectiveness of TFX in production.*
- *A framework for evaluating ML Ops tools is essential to navigate the overwhelming number of available options.*
- *TFX has useful components like TensorFlow Data Validation and TensorBoard, but many aspects are criticized for being limiting and not user-friendly.*
- *Iterating on data transformations in TFX can be slow and cumbersome due to its reliance on TensorFlow operations.*
- *The speaker argues that tools should reduce cognitive load and allow for easy interaction, which TFX often fails to do.*
- *There is a warning against becoming a "tool zealot," as it can lead to biases in problem-solving and hiring practices.*
- *The speaker suggests that while TFX may be suitable for specific scenarios, it is generally a poor fit for most users due to its complexity and documentation issues.*

okay so today i'm going to be talking about evaluating ml ops tools but just a little bit about myself so as chip mentioned you know i currently work at a startup but my general background is i've spent a lot of time building ml tooling so i've done work on this at companies like data robot i've been involved a lot with fast ai but also been building ml tooling at airbnb and github

and i've been involved with in open source as well and i've worked on both what quote cool applied ml problems at airbnb and github and then also like a lot of uncool applied ml as part of like when i used to be a consultant in things like retail telecommunications restaurants things like that so i've seen all kinds of different data science kind of flavors and that really informs my views on tools

okay so the so this talk is about this talks is about how to evaluate envelopes tools so i just want to share the motivation for this talk like how why why am i even given this talk so actually like recently there there's a person who's an ml educator author very respected in the ml community came to me and said hey you know like

hamill people don't use pie torch for production like come on like no real applied ml co like

occurs in pytorch they don't have like good tools and then also you know kind of on and on and kind of you know it's like a little bit of a provocative in a provocative sense and also like things like hey like certain problems are completely solved in in tfx and i thought this was really interesting and surprising and it made me realize that this actually can happen this actually is very common if it happened to me i'm

an experienced person who has building ml tools and i'm kind of it's kind of known that i've done a lot of work with pi torch or with fastai that someone would come to me and kind of say that it made me realize that this is happening to everyone all the time because this actually happens yeah this happens to me quite frequently and when i talk to other people it happens happens to them also and i think

as you as a student taking this ml systems course i think like one of the things that you have to think about is tooling i mean that's where the rubber meets the road that's an important consideration when you go out into the world and so you're going to encounter people who are very passionate about certain tools and you know what i would call zealots even and i think it's kind of actually natural

it's not necessarily i think it's human nature when there's a lot of entropy in a field so this picture on the left is kind of this map of all ml ops tooling and it's so gigantic you can't even see it without a microscope and i think it's natural human nature to say to try to make sense of that and say and want to kind of cut the noise and say well i found one tool that is the

best and all the other tools are wrong that's kind of a very appealing thing to to to believe in and it's something that you know it will kind of you will encounter in real life it's actually a very difficult thing to resist especially when you're first starting out and it's kind of like especially when there's appeal to authority like hey this certain tool is used at google or facebook or netflix or wherever it might be and i think as someone starting out

it's good to have some framework or some example of like how to think about tools so this is my criterion for evaluating tools i'm not going to even read it i just know that there's some criterion for evaluating tools you can revisit the slides later i don't want to read this but i will touch on all of these during the presentation okay so just coming back to tfx tensorflow extended and so like what i want to do is

go through a concrete example with tfx using tfx just because i think that it's really hard to learn or gain an intuition of how to evaluate mlaps tools in the abstract i just gave you a list of principles and rules it would be kind of hard to apply them without perhaps listening to someone else apply them or at least have like some concrete sort of example of someone walking through that so

so i'm going to be kind of focusing on tensorflow extended today and again it's not that i hate tensorflow extended or anything like that it's just i think it's a really good motivating example for this general problem i described so first i'm going to talk really quickly about things i like about tfx i'm not going to spend actually too

much time about on the things i like i'm going to spend more times with things i'm critical on

because i think that's where it's more interesting so just really quickly so tensorflow extended has something called tensorflow data validation it's very decoupled from tensorflow you can pass in a data frame and you get all you get very nice eda for example in a notebook and it does a lot of boilerplate exploratory data analysis for you and it's pretty nice you don't even like i say you don't have to use tensorflow and just two lines of code it's very nice

and similarly the same similar library helps you validate statistics validate schema so you can quickly check the schema of your data do some like minimal testing of your data at least and it's very lightweight another thing i like about things under the tfx umbrella is a tensorboard it's really nice you can use it with any framework people really like it generally and it's nice and easy to use as well documented and then also

tf serving that's another thing under the tfx umbrella that i really like it's very well documented this helps you serve models and in a nice easy to understand way it abstracts things very nicely and it's also well documented and has really nice ap well-designed thoughtful apis so things i don't like about tfx is everything else and i'll talk more about that so so the first thing that i want to mention is tools need to

get out of your way especially in critical parts of the workflow so one critical part of the workflow is iterating on data so this is andrew wang and as you know andrew wang is a proponent of the data-centric modality of iterating on problems i know that chip actually touched on this earlier in the course not to say that data centric modality is or this data centric view is the right view but i do think that

either way iterating on your data is very critical and so kind of looking at tfx what we see is in the ta they have a something called tfx transform which is the library used to do feature engineering and transforming your data the problem with this api is you have to use a limited limited set of tensorflow ops so you so if you want to do a simple mathematical

operations like greater than you can't really use plain python code you have to use you have to call a function for example that says tf grader and so this is this actually can slow down feature engineering quite significantly this is because you know you if you have played python code or using pandas you have to refactor that or let's say if you're using spacey to do some pre-processing or another library you have to refactor all of that into tf

ops so it can be quite limiting another thing is if you want to iterate on data transformations a lot of these things are it can't be performed interactively in tfx so even the official tutorials they they write a python file to disk and then execute that python file via another thing that calls it and and you know iteration is really slow so

you know if you want to change some feature engineering in certain parts of the workflow you have you can't you can't do work interactively and also the hello world pave paths kind of you know in the documentation when you're using tfx they jump directly into distributed computing and things like apache beam and apache

beam is a very complex framework for distributed data processing

and it can actually be very difficult to reason about ins and i'm a big proponent of what's in the documentation is is the product so while there might be another way to do things if the documentation you know in their hello world examples pushes you in one direction then and it's really in they don't really present other options very clearly then it clearly is what the product provides another thing i want to mention is with

regards to feature engineering and sort of understanding your data is it's important to visualize your data i think we can all agree or probably understand that by now but what you see here is aniscom's quartet basically all the data you see here has the same descriptive statistics so same mean variance in correlation and you know you need to visualize your data to really understand it

so tensorflow tfx has a sub project called tensorflow model analysis in tensorflow model analysis the basic crux of it is you define these config files so you define these json files that you see here and that defines your visualizations and again it's very difficult to iterate it's not interactive if you want to make a change to a visualization first of all you have to learn this special dsl for

visualizations and it it can very much slow you down and so this is what the output looks like it's you know visualizations and it's of various kinds but in the end of the day it's not really anything terribly new with regards to visualization in python and this is a similar related library of tensorflow model analysis again you write these giant config files json format and you get

you know you slice models and visualize them on various dimensions of your data but again it's very difficult to reason about is a very steep learning curve and so one thing i want to mention in this case like python has really great data visualization tools things like matplotlib okay seaborn altair alterhaps is my personal favorite but kind of you can't use this in in tfx you have to use you can't use

the tools that you're comfortable with and you know that that can be problematic one thing i want to mention also is like metrics on slices of your data is something that you can do with existing tools like pandas and even if your data is at a larger scale there are other things like pandas that offer consistent or similar apis that are easier to reason about and i'll discuss some of those later okay so another thing i want to

mention that you have to kind of really be wary of or kind of pay attention to is tool myopia can lead to blind spots with when especially in applied machine learning so when a tool offers a solution to a problem can in you know if you're if you're really dedic if you're really in love with the tool you might think that you have solved that problem so one thing i want to mention is skew detection in tfx

so skew is when your trading distribution differs from your serving distribution or you know even like your chain distributions differs from your holdout distribution or any partition or data that you want to measure so tfx comes bundled with some apis for helping you detect skew they they offer these like categorical skew as

measured with this l infinity norm and numeric skus with this jensen jensen shannon divergence the details of

that doesn't doesn't matter what i want to really highlight is it's not really clear the documentation of how these are calculated and what summary statistics are included in the element family norm for l infinity norm for example but really like the main problem is it's not very actionable so you might get a report but you don't really know what to do next you get like a you know a score of some kind and you know something something is

above a certain threshold but you don't really know you're not really sure like where it comes from or what to do next and you know you and then another thing is this is done in a very univariate way so it's one feature at a time in isolation and so there's other things that can give you more visibility so this is a screenshot of great expectations great expectations offers you similar things but it's a lot more human

readable and understandable another thing that i want to mention is skew detection there are there are more like practical approaches to skew detection that can be you know just as robust or even more robust so there's something called adversarial validation that's when you train a model to discriminate between your data partition so train let's say training and servings serving set and if the model has any predictive power

at all you know that is indicative of of drift or sku and so the good thing about this is you can use all of your familiar machine learning interpretability machinery to then drill into like why that's happening and so you don't need any new infrastructure or tools to do this and then also like this is not limited to it's not a univariate test basically you know you can catch complex interactions in your data that are indicative of skew

okay so the next thing i want to talk about is ergonomics and so like one of my favorite machine learning tools is keras and i know when you get confused between tfx and keras i think chaos is actually very different in you know its design and philosophy even though it's kind of somewhat related in the larger tensorflow ecosystem and and i think it's like one of the most loved ml apis so one principle is like documentation

is more important than features and i think this is really like this is really true this really resonates with me you know something's not documented it doesn't exist and you know even francois would go as far to say is like documentation is more important than features and and i agree and tfx you know documentation is often missing a lot of the apis i see are not documented at all and you know it's and sometimes it's

kind of funny like you'll get documentation like this and you'll even be asked if this is helpful or not another thing another principle that francois outlines is the apis should reduce cognitive load in in boilerplate so you know you shouldn't have to look up things in a tutorial documentation to do very trivial things so this is an example of how you remove an item from a set so this is a

in tensorflow data validation you get a feature and then you need to use a not environment you know you need to use this non-environment method and append it to that so it's very unintuitive this is nothing like how you would remove something from a set in python but you know this is a dsl that you know you would this is the

cognitive load of kind of what the kind of kinds of things you have to deal with

similarly configuring a metadata store is pretty there's a lot of boilerplate involved you know this is just an example but you have to do a lot of things that you wouldn't have to do in another framework let's say like ml flow or metal flow that have sensible defaults and even doing something like a train validation split involves tons of boilerplate and kind of like a like a bit of a new api that you have to reason

about like you have to think about hash buckets and this is like a lot you know this is quite different than sort of what you're used to and it forces you to like take on more cognitive load another thing is a progressive disclosure of complexity so ml ops tools should make it really easy to get started on like very easy things but then scale up with you to do more complicated things and and allow you to learn incrementally and i

mentioned this before but like when you jump into tfx it goes straight into apache beam which is quite complicated it's a it is a a different dsl that kind of mutates the python language for one thing but also you have to reason about distributed computing right out of the box which is which is kind of it is it is a lot to take on so kind of after all these grievances it

would you know like people that this person would say wait there's reasons for all this stuff all the stuff you you that you described there's there's reasons why this is this is the way it is so the first reason and the thing that's kind of thrown out there quite a lot is a scalability argument so you know hey like we need scale like apache beam and things like that allows us to scale and i think it's really important to

keep in mind that scaling when you don't need to scale is is really counterproductive and this famous quote by donald knuth is true i mean premature optimization is the root of all evil and so if you try to scale from the beginning like using apache apache beam you might never get off the ground but then also i just want to mention like this other solutions like using large single large nodes and using multi processing works for a lot of applied ml problems

and then things like dasc as well so there's other ways to achieve scale and data processing without cognitive overload another argument that's sort of put forth is if you don't try to tie your transforms to your model so so for example you know if you don't use those tfops like the tf greater thing you know if you don't the reason we do that is so you can tie your transforms to your model the reason for doing that

is to bake it into one portable unit is so you don't get trained serving skew so they can have reproducibility and so i think that is a weak argument because you have other tools that facilitate this like weights and biases ml flow metaflow and many other things that help you with reproducibility of your workflows and then when i press this person on all of these items i mean i did show this person many of these slides

and i said hey like how do you even get like tf transform i mean how do you even do this like how can you expect people to get started so the answer was hey like just don't use it until you're ready to use it and then just

refactor everything to to to tf transforms like when you need it and you know this is kind of the definition of technical debt you know you're just kicking the can

down the road and you're going to have to pay a pretty high cost to to using a platform that forces you into this paradigm and then finally kind of the last like one of the arguments put forth was hey like you know things like apache beam and other things are made really easy for you on google cloud and and this is a general kind of principle that might be put forth by a lot of mlops tooling providers out there

it's like hey if you use this cloud or use this thing it's really easy and i would be really careful of that so in this case hey like google cloud only has 10 market share so do you really want to gamble with your career by like investing all your time and time and skills and learning something that only applies to that's only really effective on google cloud i would think about that and then kind of the ultimate argument

which actually traps a lot of people and it's very seductive is hey is this appeal to authority so for example hey like google is really smart if they're doing it then it must be it must be the right answer like you can you know of course they were successful with ml just use their tools and you'll have the same success and this is you know it sounds it might look silly on this slide but it actually you will encounter this quite

frequently in real life and so what i would say is like this is a religious argument this is not a principled argument that makes any sense and actually like this kind of this is a kind of a very traditional marketing tool that you can be like somebody if you just use their tools so this is an air jordan okay so some final thoughts here so so google actually produces really good research on technical debt and machine

learning systems and kind of really has helped push the field forward and in like making us all aware of the importance of having tools the one thing i want to say is like that doesn't mean that you should use their tools and i don't think like they're telling people that you need to use their tools either i think there's a lot of a lot of people misunderstand and they think they draw an equivalence that hey like these people produce the

research therefore you should use their tools and so that's another kind of trap that people fall into in this so that's great so you might be wondering okay hamill just gave the stock when would i recommend tfx so actually like it's not binary like i hate it or love it i actually love some components and dislike others but i do feel it's a bad fit for a vast majority of people just based on what i've observed and some of the things i

mentioned but despite that it might be a good fit if you're you know really committed to tensorflow if you're running on google cloud already or like if you need to deploy to edge things like edge devices if you want if you really need that portability or if you are operating at a scale where somehow the optimization is like really important to you but even then i'm not very convinced it's a great fit for many people based on

just things that i've seen and you know things like documentation which is almost makes you know lack of documentation is makes it unusable but if you if you take away one thing from this talk it's it's this slide and one

thing i want to emphasize is don't become a tool zealot it's really easy to become a tool zealot and you know as you form data science teams as you you know go into the workforce and start working at all these

companies you're gonna have to make decisions about what tools to use and so one thing just always keep in mind is like the tools that you choose especially ones that are more monolithic and don't operate well with other tools you know it will narrow the way you think about problems and also introduce bias towards working on certain things like if you know the the the places where the tool introduces lots of friction let's say for in this

case feature engineering you know it will bias you towards working less on feature engineering and more towards working on other things and then like very importantly like if you are too myopic in your tools and you know that will prevent you from hiring diverse talent so for example i don't think i could work with that person that was very convinced that tfx was the only right tool on the planet to use for machine

learning and you know that i think that is really important too as you go out there and think about these things but then also like you know what tools you use can also lead to blind spots and and really like i think it's important to try tools often especially ones have different approaches to things i think that can you know kind of the counterpoint to blind spots and you know all the things i mentioned just now i mean these are the same

reasons you should learn other programming languages but even if you don't have time to learn different ml ops tools and different things one thing i really want to emphasize is to keep an open mind and then lastly like i hope that what you take away from this talk is kind of an example of someone evaluating a tool and providing some reasons why it didn't work for them and then kind of seeing which one of these things i

mentioned resonates with you and having more confidence to either push back or sort of challenge certain certain kind of zealots that may come at you or even have better discussions about amalops tooling because in practice i think this subject is very important all right thank you and hope you enjoyed this talk