Become a Data Scientist

Essential Tips to Compliment your Technical Studies

by Manuel Amunategui & amunategui.github.io

—

My #1 Tip to become a data scientist,
Find your next programming language,
Stay creative in this fast-paced profession,
and much, much more…

—

(Source: Lucas Amunategui)

Table of Contents

Introduction

In the following chapters, I'll cover material that is rarely, if ever, covered in data science classes. I'm not knocking anybody down here as you will need all that technical knowledge, from basic statistics, linear algebra, and everything/anything modeling. But, seeing that 95% of schools, online classes, and books covering data science already cover that material really well, this mini-book will only tackle the other 5%. It may seem like a small number, but these are critical pieces that will put your career in turbo-boost mode and help you leap over the competition.

It comes down to one thing, and it's a saying I love to say,

> "It just ain't real 'til it reaches your customer's plate"

Yes, it's simple but very complex and therein lies the rub. Everything we do in data science has absolutely no real value until it's in your customers hands and applied towards solving real problems, streamlining processes or boosting sales by finding strategic nuggets in data. We build the tools and the field users will tell us if it's of value or not. Employers want applied data scientists. People that can talk to customers, understand what they need, and deliver successfully on those promises.

Join me on this journey in becoming a well-balanced, well-versed data scientist!

My #1 Piece of Advice for Aspiring Data Scientists

Source: Lucas Amunategui

Whenever I am asked how to break into this field, I reply, 'push it out into the public domain on a regular basis'.

Being a blogger and liking the sound of my voice, I unfortunately murk it up with more stuff, like taking classes on Udacity and Coursera, participating in competitions on Kaggle, and getting a degree in data science. But, at the top of the list, will always be, create, invent and push out regularly!

It may go against the grain within some organizations, so don't share trade secrets or other people's IP; but there's always something within your skillset that others will find interesting.

It's a miracle technique! It's isn't about quantity or quality but about doing it on a regular basis and building that production muscle—it becomes magic! It will take you to unexpected places. Kind of like compounding interest or butter in your coffee—there's no downside to this approach!

Self-Development

You will learn to express yourself better, write better, and put yourself out there. You will gain a fuller grasp of the topics discussed—there is nothing like explaining it to others to learn it better yourself.

Great For Finding Jobs And Networking

You will be sending the clear message that you enjoy this stuff and spend time on it. That you are motivated, entrepreneurial, and you know your stuff—what kind of employer or customer could resist that? You will meet a lot of people, not just students but a wide range of professionals as well. You will be invited to speak at conferences and meet ups.

Great For Your Current Job

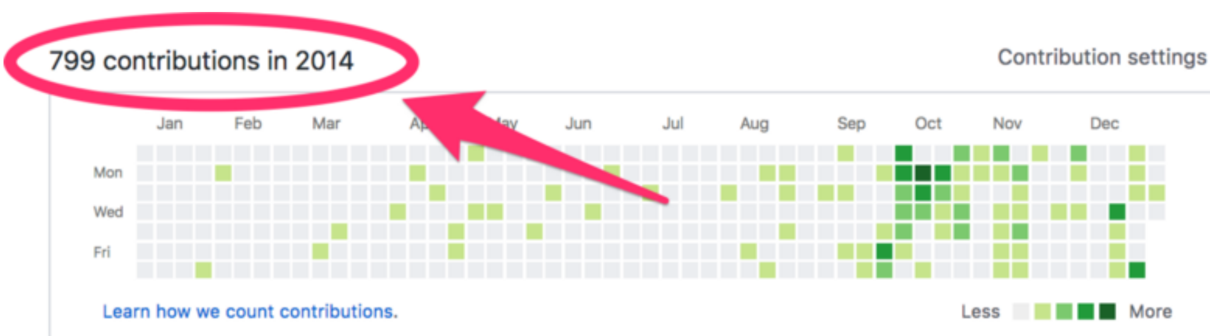
These may be stories for another entry, but I have connected with colleagues within a large health system through my You Tubes, I kid you not—people I would email directly and never get a reply—sometimes a social media knock is louder than an office door knock.

You become Accountable

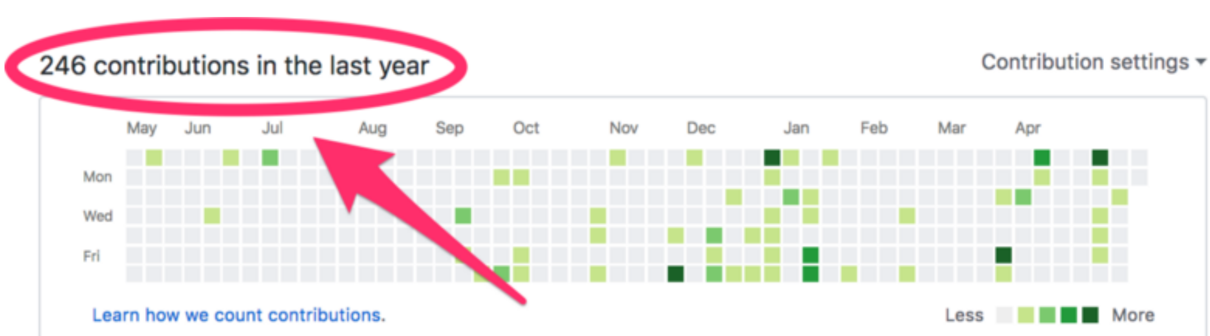
You become part of the educational and research community, you become part of the internet, it becomes you!

And I Eat My Dog Food

I am a believer and follower—I started pushing code and blog entries actively In 201



And 4 years later, barely halfway through 2018, I'm still pushing stuff!



So don't worry too much about what you're writing about, worry more about doing it on a regular basis. If you like working on projects, push out on GitHub, if you write articles, try Medium, or Reddit, if you like to give professional advice, try LinkedIn. Roll your own, do it on WordPress or GitHub.io. If you are a live type, try podcasts and YouTubes, mix and match, do multiples, or do them all!

Push!

Thanks for reading!
Manuel Amunategui
amunategui.github.io

What they Didn't Teach at Data Science School, and How to Fix It to 10x Your Career

Source: Lucas Amunategui

Even though I had a stellar CV, I was turned down for the position. I had, what I thought at the time, plenty of machine learning and modeling experience. I applied for a position in Holland—I was ready for something new and wanted to show my family the world. Unfortunately, they told me they wanted a data scientist with more commercial and "full pipeline" delivery experience—"please, let the next candidate in on your way out". This was a few years back when things were less competitive, it is that much more important today.

Full Pipeline Data Scientist

And they were partly right. I built internal models for a hospital that may not have qualified as 'commercial' but I certainly built my share of pipelines. The problem is that most data scientist today have less applied skills than I had back then. Our educational system cranks them out that way. Some don't even know what pipeline experience means, and if they do, they may not know how to implement one.

Full-pipeline experience is synonymous with being a data scientist and full stack developer squeezed into one. Some will argue that these positions are very different and would be better accomplished by different team members. But in most cases, on smaller teams, in fast startups, and more importantly for intuitive data science solutions, a data scientist should do it all, or at the very least, design it all and have others implement it. And, in the era of 'A.I., the human job pillager', the more useful you are, the longer you'll survive.

It Ain't Real Until it Reaches Your Customer's Plate

It is critical for a data scientist to not only understand the data, the model, and how to explain the output, they have to understand how it is going to be consumed by the end user. Whether a medical staff needing life-saving prognostics or a customer asking for clothing recommendations, today's data scientists need to understand how their output will be digested. This is critical, some will be lost, not have a clue what to do with a percentage—like 60% or 80% good loan recipient? You will need to work with the business expert or build tunable parameter so they can tune the output threshold as their business grows. Others will be insulted with a true/false output and require the probability to scale their work or assign budget accordingly. And no, an end user will never need an AUC score…

Being a data scientist is a wonderful profession but there is a troubling gap in the teaching material when trying to become one. Data science isn't about statistics and modeling, it is about fulfilling human needs and solving real problems. Not enough material tackles the big picture. That's what is missing in this profession's educational syllabus. If you build first then talk to your customer, your pipelines will be flawed, and your solutions will miss their target.

Some Ideas to Get You Started

Mind you, there are plenty of ways to build full pipelines but here are some open source solutions so we don't get bogged down with proprietary solutions or NDAs (I apologize that it's pointing to my own blog materials, but like I said, not much out there from an ML starting point).

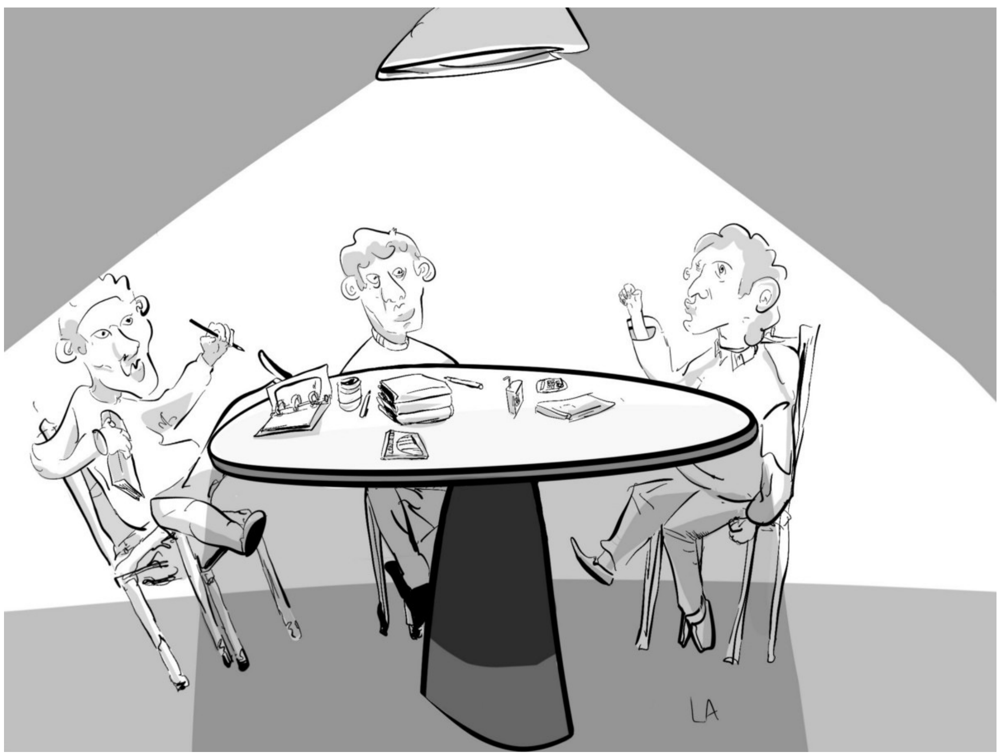
- Life Coefficients—Modeling Life Expectancy and Prototyping it on the Web with Flask and PythonAnywhere (http://amunategui.github.io/life-expectancy/index.html)
- Rapid Prototyping on Google App Engine—Easily Extend your Python ML Models into Interactive Web Applications—Trip Planner with Google Maps and Yelp (http://amunategui.github.io/rapid-prototyping-app-engine-yelp/index.html)

Today’s Models are Complex

Gone are the days of the single model/prediction in spreadsheet solution, today, it can be a choreography of multiple models working asynchronously or synchronously feeding one model’s prediction into another. A customer dashboard may contain one or many outputs, may have complex tunable parameters, may have visuals reaching into the internals of your models. You need to be involved until the end, it is your responsibility—you need to understand the ‘full pipeline’.

Thanks for reading!
Manuel Amunategui
amunategui.github.io

My Six Favorite Free Data Science Classes and the Giants Behind Them

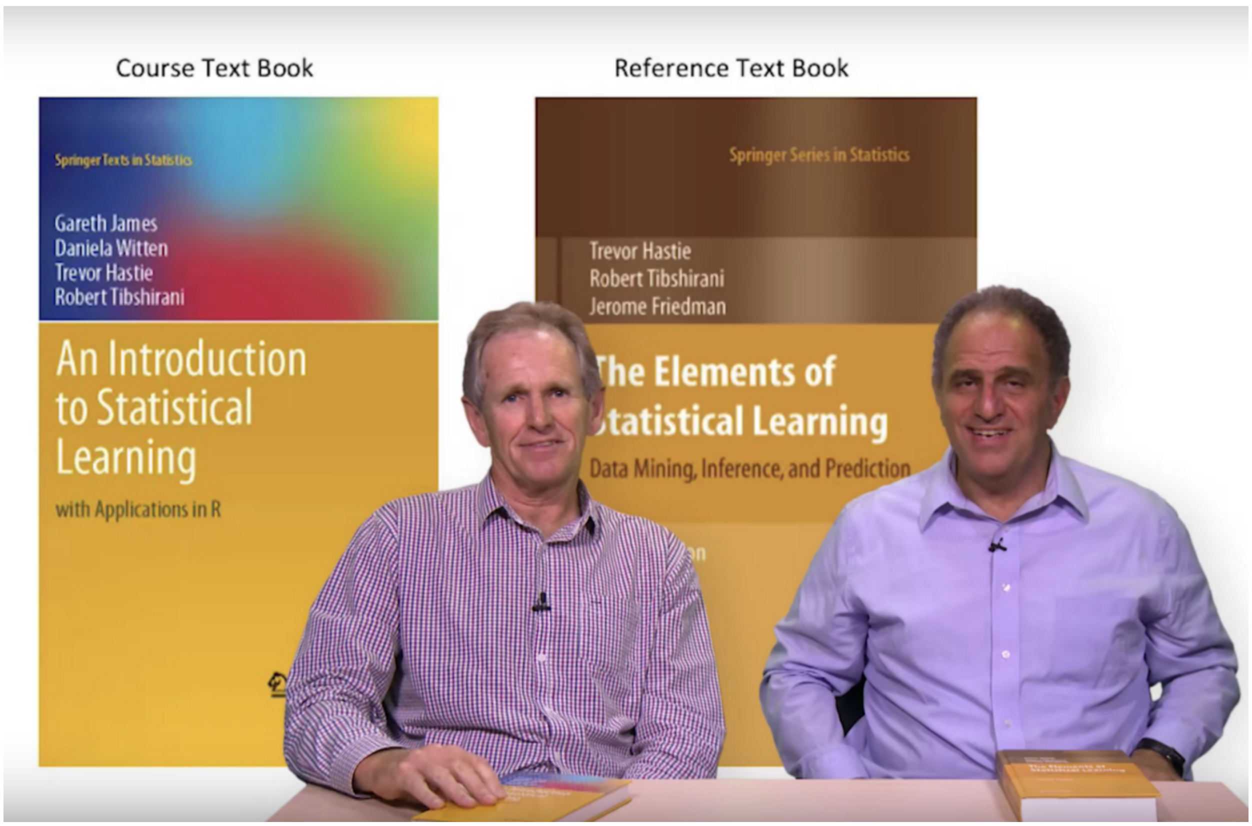


Source: Lucas Amunategui

The tech field, more than other fields, is one where people feel compelled and comfortable sharing what they have learned and what they're excited about. We all do it. But, once in a while, comes a ground-breaking class, not only for its timely topic, but for its incredible delivery and ability to hold on to our attention till the end. They leave us with a real grasp on the material, new ideas on how to apply it, and that exciting feeling of having grown a tad wiser.

To the Giants!

Number 1: Trevor Hastie, Robert Tibshirani — Data Science Introductions using R

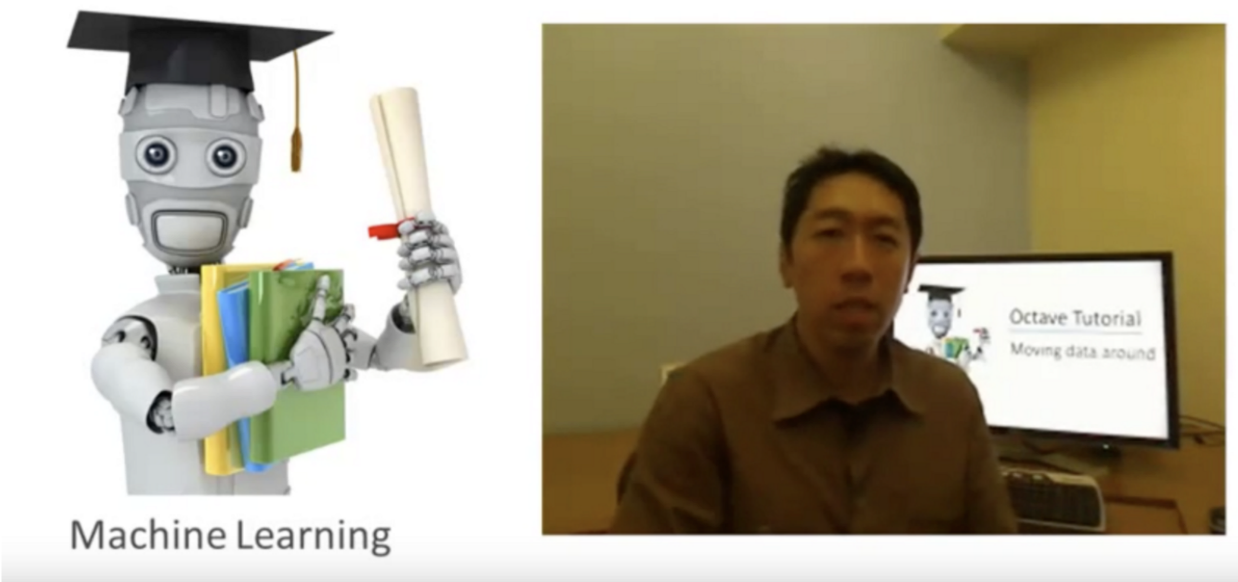


All lists start with the phenomenal duo of Trevor Hastie and Robert Tibshirani. Originally from South Africa and Canada, today, they are both firmly seated at Stanford. They've shaped a whole generation of data scientists in areas of machine learning and statistics, all the while delivering the material in a hilarious and engaging style—can't thank you two enough!

They created a popular interactive course using the R programming language, and two essential and freely downloadable books on statistics and ML:

- Statistical Learning (Self-Paced)
- The Elements of Statistical Learning: Data Mining, Inference, and Prediction
- An Introduction to Statistical Learning with Applications in R

Number 2: Andrew Ng — Machine Learning (Octave)



Andrew Ng is a superstar professor and his seminal course on machine learning has propelled the career of so many students by not only digging down to the root of modeling and neural networks but keeping it understandable and fluid. Andrew's delivery is incredible. This is a hands-on course using Octave. A big thanks to you, Andrew! Great backprop example: https://www.youtube.com/watch?v=_2zt4yVCkGk

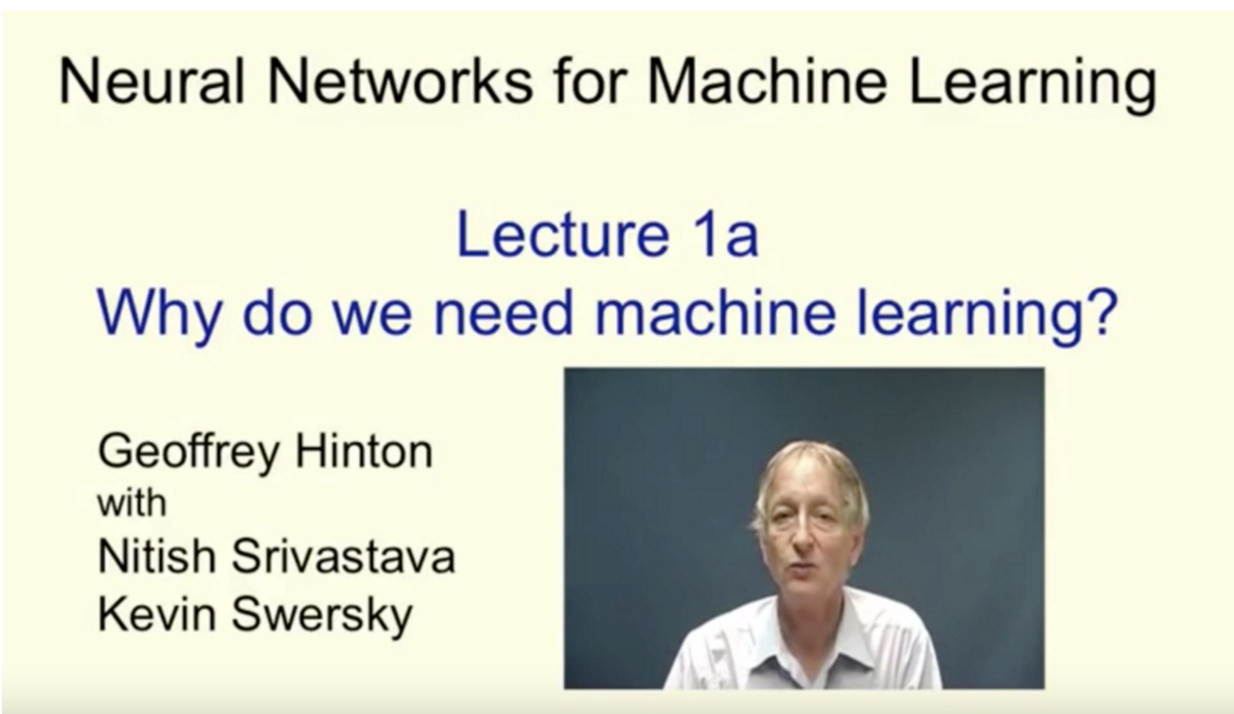
Online free course—Statistical Learning (video lectures available on YouTube as well):

- Machine Learning
- Machine Learning—Andrew Ng, Stanford University [FULL COURSE]—112 videos

Andrew's new class:

- https://www.deeplearning.ai/

Number 3: Geoffrey Hinton — Neural Networks for Machine Learning (Python)



Professor Hinton is a legend in our field. What is more apt than learning about neural networks than learning about it from the father of back-propagation, Boltzman machines, and now, Capsule Networks? Hats off!

- Neural Networks for Machine Learning

Number 4: Nine Month Data Science Specialization from John Hopkins and Coursera

"Launch Your Career in Data Science. A nine-course introduction to data science, developed and taught by leading professors."

From three Johns Hopkins University professors: Brian Caffo, Roger Peng, Jeff

Leek. This course isn't necessarily free, you can audit it but if you can have your work reimburse you for it, they would be making a very wise investment. Also, I wouldn't call it a class but a 9-months journey, taking you from a beginner in data science and guiding you all the way to professional fluency. You'll learn critical skills around EDA, reproducibility, various modeling techniques and seal the class with a capstone project. The class is hands-on, self-paced, and uses the R programming language. This is one of the few classes out there that teaches you holistic, applied data science skills—great series!

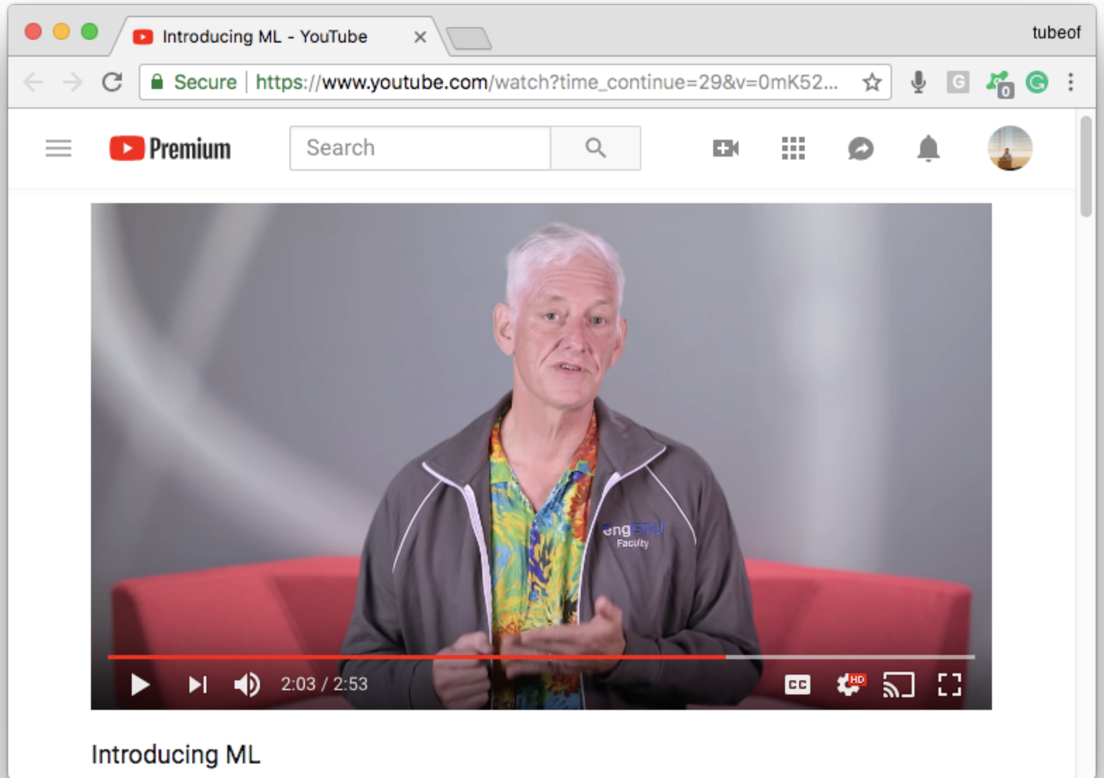
- 9-Week Data Science Specialization from John Hopkins and Coursera

Number 5: Sebastian Thrun and Peter Norvig — Intro to Artificial Intelligence

These two need little introduction—Sebastien was part of the winning 2005 DARPA Grand Challenge and co-founder of Udacity and Peter is Director of research at Google. Put these two thinkers and entrepreneurs together and you know that you're going to have a great and practical class! I'll also add that the Udacity "Self-Driving Car Engineer Nanodegree", though not a free class, was a great introduction to autonomous driving.

- Intro to Artificial Intelligence

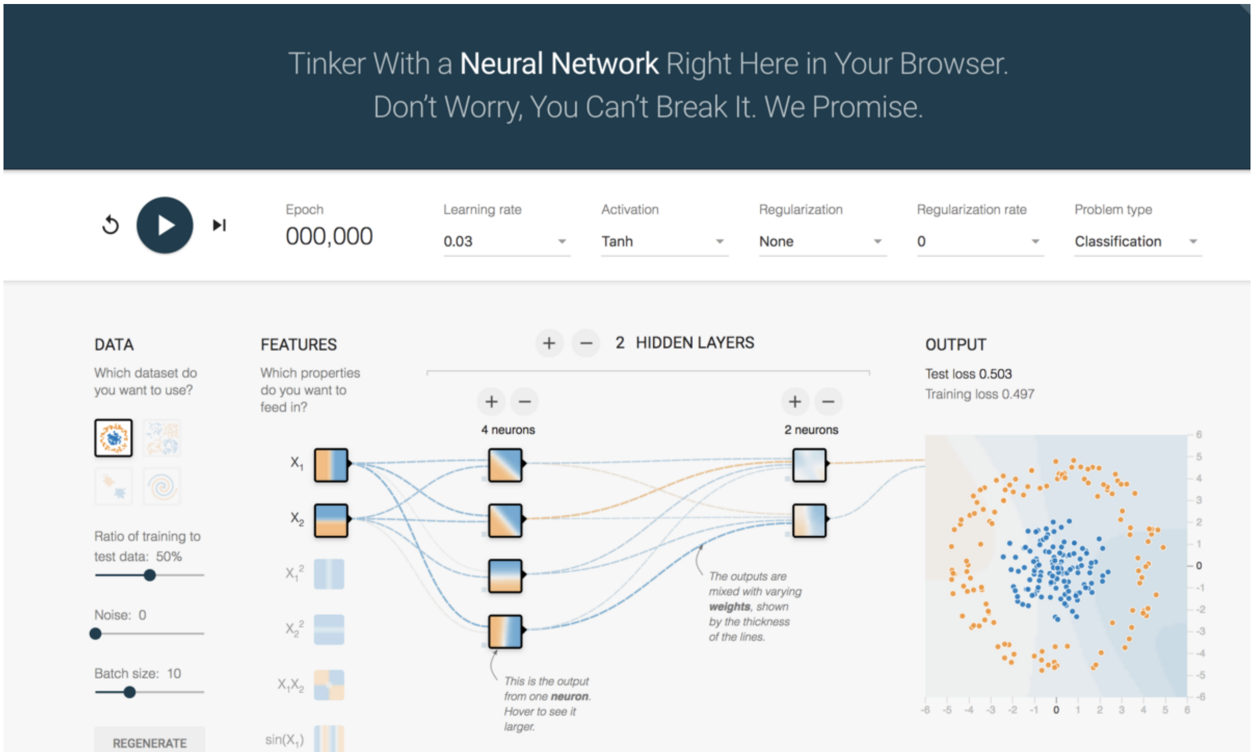
Number 6: Peter Norvig + team — Machine Learning Crash Course



Peter is everywhere! The Machine Learning Crash Courses from Google are top notch for those wanting to learn everything TensorFlow!

- A self-study guide for aspiring machine learning practitioners

A Great Resource: TennsorFlow Playground (Daniel Smilkov and Shan Carter)



Deep neural networks are sometimes hard to understand and can feel impossible to tune. This is a great interactive tool to help you learn both.

- TennsorFlow Playground

Conclusion

Props to the giants that have generously created these ground-shaking classes, made them available for free, and formed this new generation of motivated and self-starting data scientists. If you've found your way by lighting your path with any of these amazing educational beacons, cheers to you!

If you follow through with these classes until the end, take them seriously, do the homework and take smart notes, you will come out of the other end as a real data scientist! We've officially run out of excuses as to why we cannot learn data science— if you hear of somebody wanting in, point him or her to any of these classes!

Thanks for reading!
Manuel Amunategui
amunategui.github.iov

Work Fast but Never Rush the Creative Process

Source: Lucas Amunategui

Who's stuck in meetings all day long and chronically strapped for time? Yes, the executive. If you know one, lack of time to think and space to create are usually their top complaints. Unfortunately, this also applies to many others, from any walks of life, employed or not, and at all levels of the professional food chain.

I am a big self-management junkie—from tracking output, to do lists, white boards, morning stand-ups, Pomodoro, standup desks, frequent breaks, no breakfasts, etc. And they all have merit. The problem is that time isn't equal in the eyes of the creative process; some things just can't be rushed. If you don't understand that, you are doing yourself a huge disfavor.

Mechanical vs. Creative

The key is to differentiate between mechanical and creative. The mechanical is what you have to do with little thought—it can be part of your job, you've done it many times before, something with tons of steps and no thinking. On the other hand, the creative process can't be rushed, you need to give it time to percolate and age in order to be great. In the long run, if you don't protect that process, your inventions won't stack up to your expectations and others will notice. You'll be the person that just sucks at coming up with new stuff. They'll label you as a follower, as a drone worker… They'll demote you by not asking for creative input and assume that mechanical is your thing. If it is, then fine, no need to read further. If it isn't, if you want to create more and create more great things, keep reading.

The Problem

Therein lies the rub, if you allow the mechanical to suffocate the creative, you will be unhappy, one way or another—you won't like your output, your job, and your employer won't like you in the long run (but thankful you handled all that crap).

You need to give yourself the proper time and space to let the creative work its magic. Personally, when it comes to original thinking, presentations, writing, I always give myself a few days for a chance to sleep over my progress a few times. This works for me, at least it sucks a lot less than the first draft. It's an iterative process that takes time to cure, sometimes in the middle of that run, I'll throw it all away and start from scratch or keep a few ideas and graft it into something better, over the next few days. Not many shortcuts to this process. But the “few days thing” isn’t the point of the article, I know people who create phenomenal content in hours. The point is to not get asphyxiated by mechanical tasks.

And My Advice

At a high-level, tackle the mechanical immediately, don’t procrastinate, don’t dilly-dally, as it will weigh in your head and clutter the creative flow. Another issue is that mechanical work isn’t always entirely mechanical, the worse is when there is enough difficulty baked-in that it weighs heavily on you and stops you from thinking about/enjoying other stuff. This is insidious and not always easy to surmount, especially if you are like me and tend to procrastinate things when they get difficult… and that, my friend, is the virus that kills the creative process.

We can put it in another way, I tell my kids to get the homework done before the fun, ’cause the fun will be that much funner!

Realize it, tackle it, own it, finish it

Start the crap stuff immediately and get it done quickly. Give yourself the right space/time to think and create.

Now, once they’re done, don’t turn it in too early, stick to deadlines, as turning crap work early will probably engender more crap work. And I am certainly not gaming any system, on the contrary, the creative work you do for your company will be 10 times more valuable than any of the mechanical stuff. The low-level manager may not get it (and if the creative was in your job description, head for the door now), but the higher ups will, and they will respect you for it and rely on you that much more!

Thanks for reading!
Manuel Amunategui
amunategui.github.io

The Guerrilla Data Scientist and that “Can Do” Attitude

Source: Lucas Amunategui

It was lunch time and wonton Tuesday at the cafeteria, but our data scientist was not there. He was far away with dirt smeared over his face and hiding behind a stack of red construction bricks trying to blend in. It was a hot day, like looking for shade under a magnifying glass kind of hot. His mind was racing, and he was working hard at staying composed and strategic, a rash decision at this point would end it all. He needed the data. It was there, right in front of him, practically within arm’s reach, yet a chasm made of trigger-happy militiamen, drugged-up illicit traffickers, exploding fires, and a lazy co-worker, made it all feel so grinding and discouraging…

Yes, a lazy co-worker, forget the other stuff. How many times did I need some data for a project or study only to be turned down by a gate keeper explaining that an SQL join of that size would bring the system to its knees, or the multi-departmental permission form was still missing 2 out of 5 signatures. And no, I wasn't requesting this on a Friday at 3:30PM or during lunch on wonton Tuesday. But I'm not here to complain, no, I'm here to celebrate the ingenuity of colleagues not letting a lack of common curiosity and common urgency stand between them and their data.

Never Mind, I'll Collect It Myself— Guerrilla Data Scientist #1

A colleague was attempting to measure how staff's late-cancelations and no-shows affected the outcome of a hospital department. At the time, the data needed was not recorded in any centralized location, actually, it wasn't recorded in a digital one either. He had a choice, shrug it up and share the merits of collecting this data during the next company meeting, or, roll up his sleeves, and go hunting!

This was no easy feat; he built a VB.net application to simplify data collection and shared it with the nurses in charge. He not only had to convince them to add this extra step in their hectic workflow, somewhere between life-saving medical procedures and giving a crap for what may have seemed like a nerdy pet-project to most, but also had to press the importance of being systematic and consistent in recording the data if it was to be of any use.

This blows me away. Besides the expected skills, this calls for vision, confidence, truly connecting with end users, aptitude to enlighten others on the importance of analytics in the era of 'A.I., the human job pillager', and the patience to painstakingly choreograph it through completion.

BYOData — Guerrilla Data Scientist #2

When you assign a cool project to an ambitious and get-it-done data scientist, the type used to quick turn-arounds and pushing boundaries, it's going to get done one way or another. And when that same project just can't get started due to data issues from a time-strapped customer with limited resources, most would just complain about it and enjoy the lull between projects. Not for this data scientist.

This was an image-modeling project around recognizing commercial products. On his way home, he spotted what looked like similar images to what was described in the project specs. He pulled over by the side of the road, iPhone in hand, snapped away at those images, and went through the laborious process of labeling the data later that evening. This gave him all the necessary images to get the model built, trained, tuned, and blow his customers' socks off.

This transcends your typical 'project requirements', the unstoppable need to explore possibilities, go places where few have been before is an undeniable trait of a successful data scientist.

See You on YouTube — Guerrilla Data Scientist #3

This has to be one of my strangest professional experiences and a big career lesson; if the idea is perfect but you can't find takers internally, give it to YouTube.

If you have a high value project with clear internal demand but just can't find takers within your company, it doesn't mean you or your project are to blame. There are hundreds of reasons why great projects never see the light of the day.

The kitchen could be filled with too many decision makers, your leadership could be ill-defined, powerless, or in transition, your company could be hit by budget restrictions, or operating under a culture of contracting out ambitious work instead of building internally.

In this case, me and my teammates knew the idea was dynamite and had the feedback from conferences where we presented the work to prove it. I found very similar, real-world public-domain data close to the real internal stuff. This allowed us to build the predictive model, work out the kinks, and even prototype a distribution mechanism (it ain't real until it reaches your customer's plate).

The project wasn't getting any traction internally, but I presented it at a few places and a friend from Bellevue Community College wanted me to make a video of our team's work for her students. I complied, presented it and, honestly, kind of moved on.

A few days later, a physician, champion in the field, got in touch with me. Coincidentally, this was somebody I had attempted to present the project to before producing the video. He raved about finding me on YouTube and how our goals were uncannily aligned. And I have no hard feelings here, just grateful for the incredible lesson. He pulled the project out of the doll-drums (and put our department on the map) and we rolled out the model to hundreds of happy users.

And here is a link to that video—it's an oldie, but helped my career and our ≈ little data science department and still brings tears to my eyes :-)

```
youtube.com/watch?v=g8pu-9n3300
```

Thanks for reading!
Manuel Amunategui
amunategui.github.io

Find Your Next Programming Language By Measuring “The Knowledge Gap” on StackOverflow.com

Source: Lucas Amunategui

Ten years ago, I was in the mood for a new programming language. The statement may sound casual and easy, but it was anything but that. I had cut my teeth on .net and ASP for a few years and was ready for something new and fresh, but not too experimental, as I wanted to grow my professional skillset and career prospects.

I couldn’t ask those around me as we were all working under the same framework and using the same languages—it would’ve been about as useful as asking it to myself. And the prospects of asking for opinions, like which is better R or Python, and suffering through the resulting long circular monologues, sounded horrendous and of little help.

I sent out an email campaign to distant contacts and carried some conversations with people outside my company. This was the best I could come up with to avoid technical bias and social conformism.

Things are different today as we have access to Stack Overflow, a phenomenal tool.

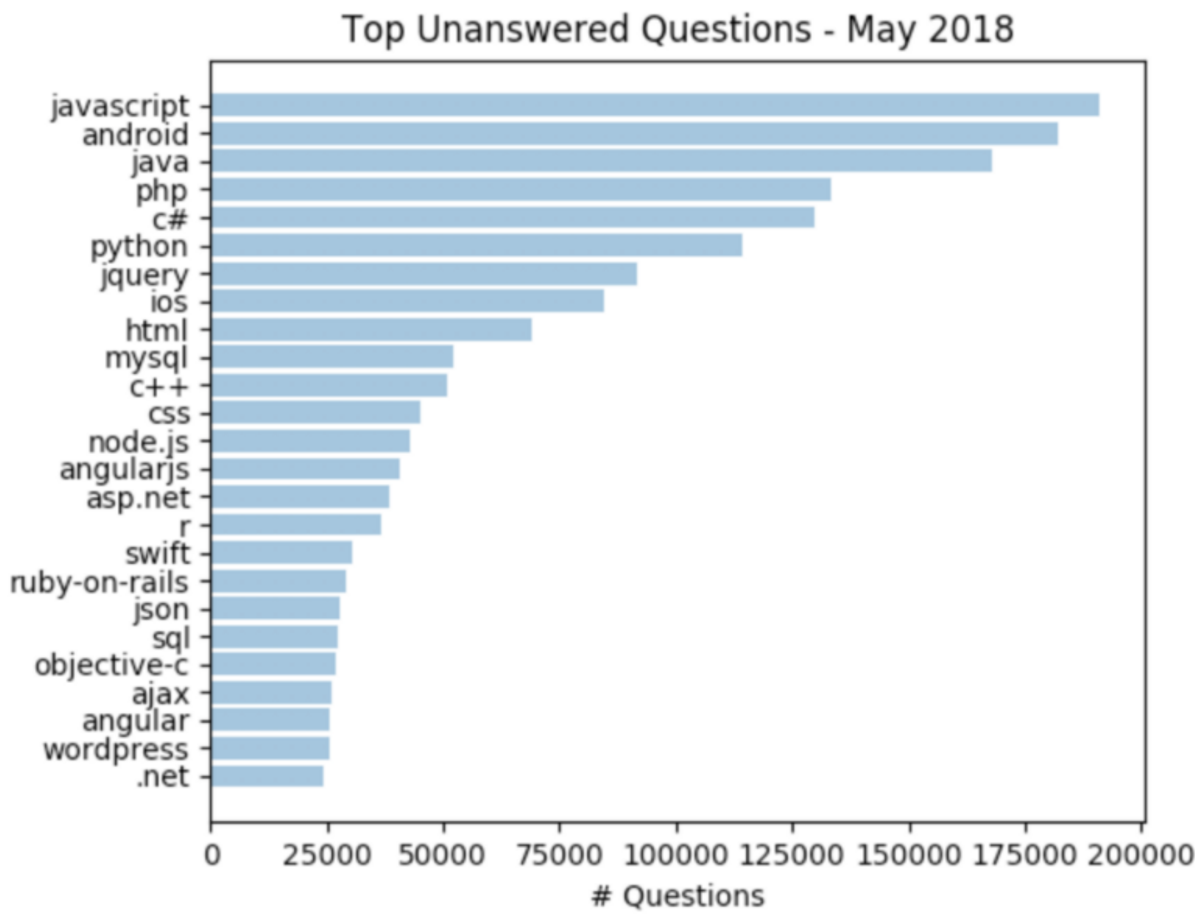
The Unanswered Questions

As part of the site's web stats, they break down the number of unanswered questions by programming language. In essence, they have invented a unit of measure, 'the knowledge gap'. Check it out at https://stackoverflow.com/unanswered and look in the lower right of the page. When you think about it, you are getting two critical pieces of information:

- How popular is the language
- How many are struggling with it

If lots are struggling and it isn't a popular language, then you conclude that the language is flawed and learning it is a bad career move. If it is popular but nobody has issues, then its probably a crowded field and not a good career move either. But when popularity and unanswered questions are aligned, boom, you've got a winner!

Currently, there's almost 2 million unanswered questions and JavaScript is leading the pack.

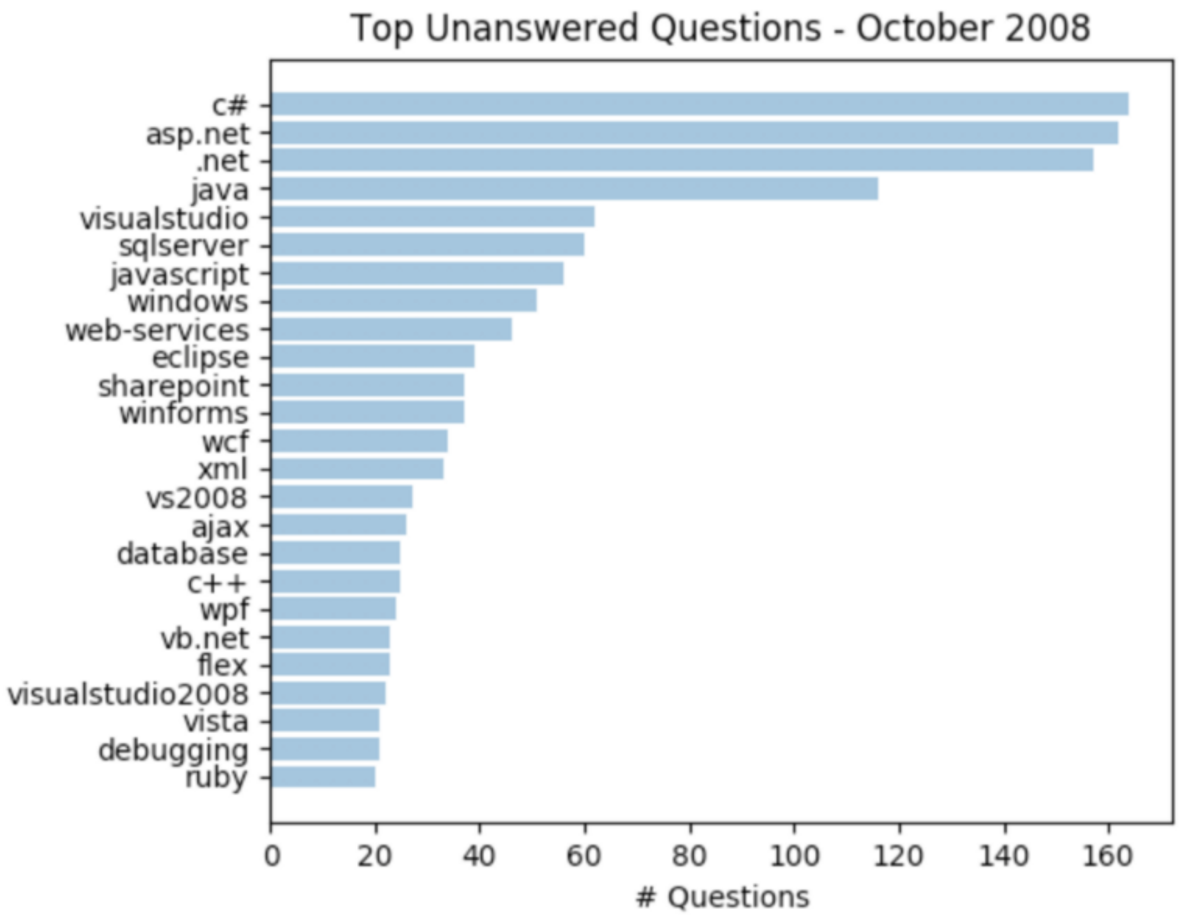


For those that aren't familiar with Stack Overflow, it is the lifeline of the tech community. We go there to whine and unload our troubles—like a psychoanalyst for programmers. If you post your questions and they are well formed, you'll copy/paste you're way out with a solution, if not, you'll get slapped around—it's very humbling. But if you're like me with an endless supply of questions, you quickly grow a thick skin—as this is a huge time saver and its all free!

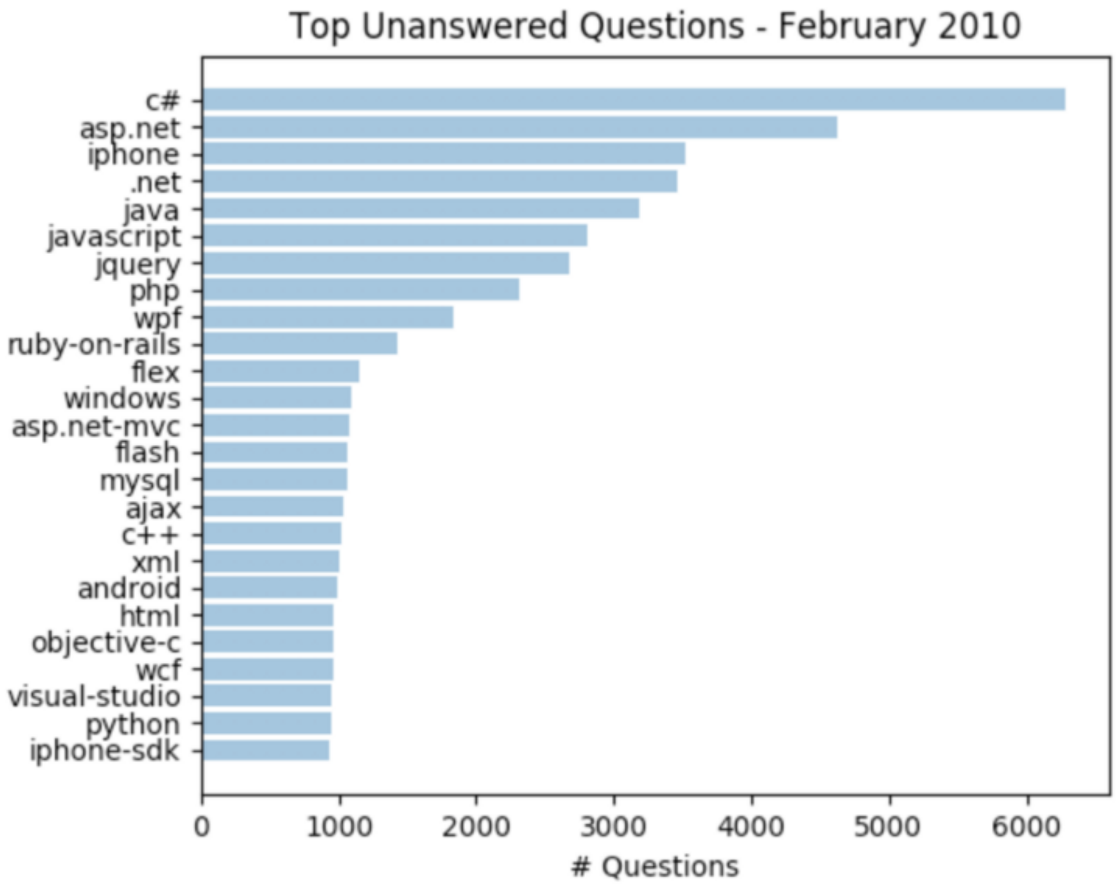
I recommend going there on a regular basis if you want to keep abreast of trends and track what is moving up or moving down the unanswered list. This is not only useful for those seeking to learn a new language, it can help recruiters, tool builders, CTO deciding on new technology, etc.

Using the Wayback Machine and traveling back 10 years (right around when

SO launched), we see that they had some 2,000 unanswered questions with C# leading the pack.



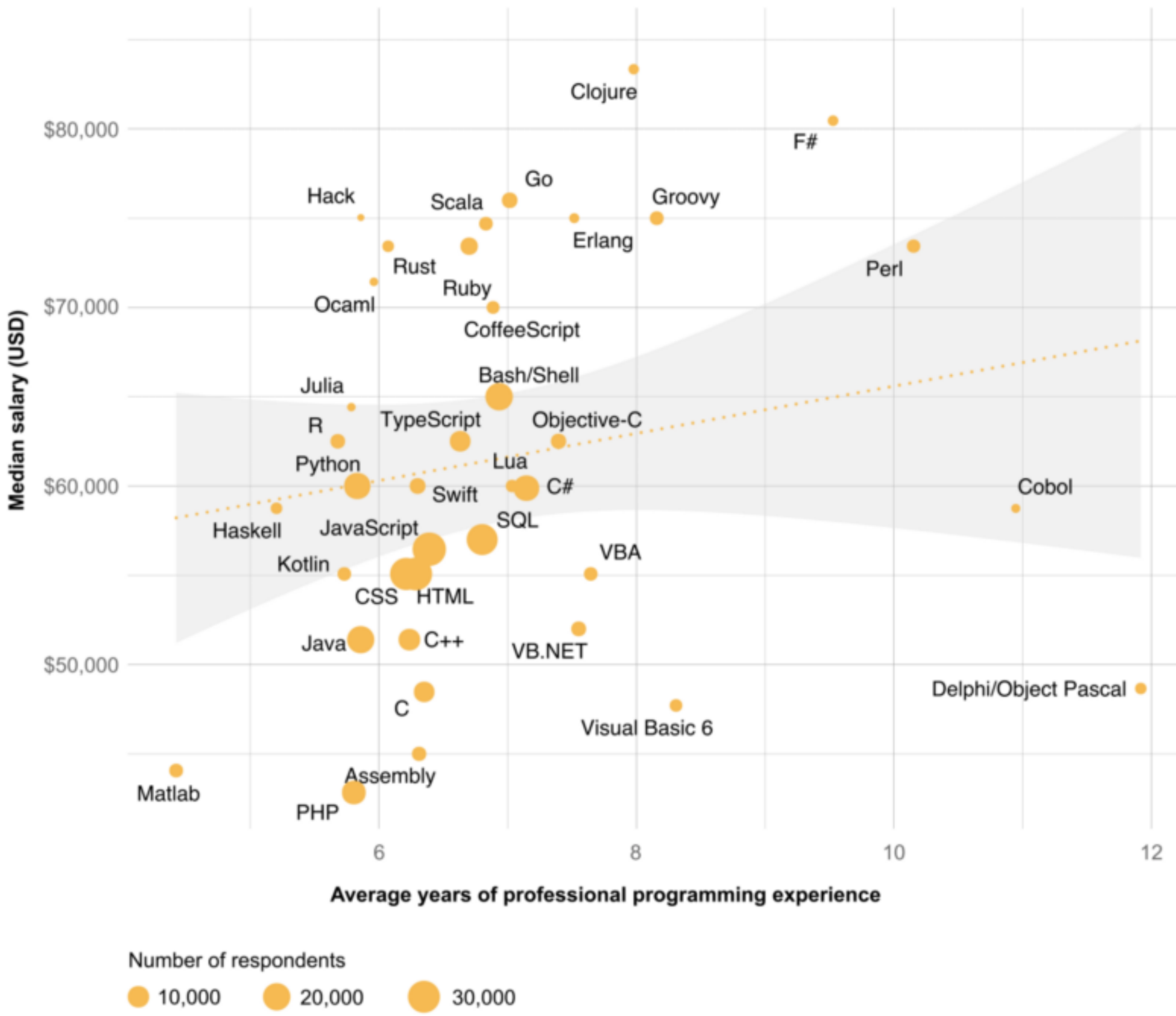
More interesting, in February 2010, with much more data to analyze, we see some 69,000 unanswered questions. C# still leading the pack but iPhone and PHP appear and moving up fast. Even Python makes a timid appearance.



I'd like to think that this data would have made me look more seriously into Java and JavaScript. But in the end, I successfully transitioned to R and Python and fell in love with both. But it is nice to know we have some cold, hard facts about what people are using and what people are struggling with. Beats asking your peers whether its best going SAS or open source—and feeling your eyes glaze over and instantly regret the ask amidst the long winded, and entirely subjective lecture.

The Insights Survey

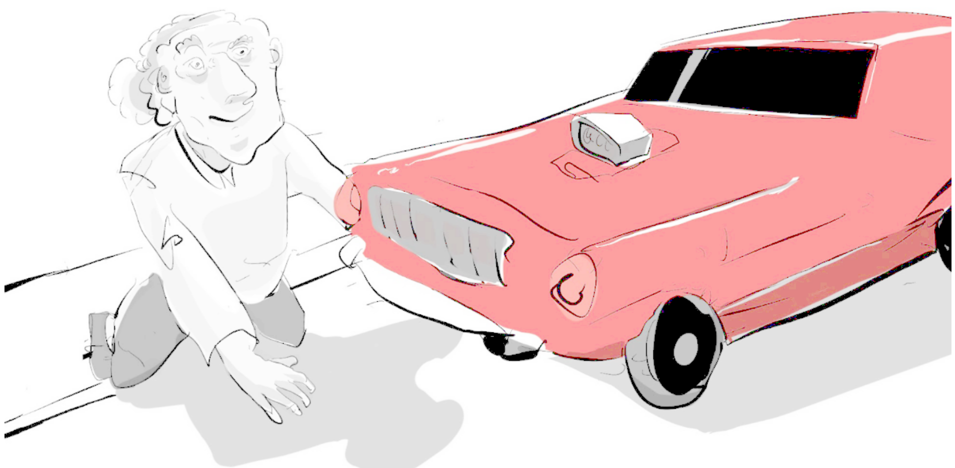
One more nugget from the good folks at Stack Overflow—I've been keeping the best for last—the Insights Survey (https://insights.stackoverflow.com/survey/2018/). This list compliments the "unanswered questions" list by throwing in salary and experience. If you look for the biggest circle, closest to the top-left, you'll draw yourself a treasure map to the best paying job with the smallest learning curve! Wow, where was that 10 years ago!!!



(Source: Stackoverflow.com)

Thanks for reading!
Manuel Amunategui
amunategui.github.io

Yes, You Want that Deep Neural Network, but is it the Best Model for your Project?



Source: Lucas Amunategui

Its new, its shiny and you want it. Choosing between a brimming new Tesla or an old Land Cruiser is a no brainer, until you need to cross the Sahara desert. This is a theme that I've been noticing lately; deep neural network models are very tempting in the ever deeper search for elusive nuggets of business intelligence.

A DNN is a tremendous choice under the right circumstances but it is hard to build well, train, tune, maintain and interpret. People seem to think that only the mysterious models can access truths beyond human grasp, while the transparent and simple ones can, well, only access the transparent and simple stuff.

But in applied data science, parsimony will always win, no matter how hard you fight. The quicker you understand that, the quicker you'll save money and your team's emotional currency.

Let Parsimony Decide

The simplest choice will always win in the long run but it isn't always obvious which that is. I certainly wouldn't want my autonomous vehicle being controlled by a linear regression, especially when I'm the passenger.

Where DNNs Make Business Sense

If you are doing any vision work and have millions of images or hundreds of videos as your data, working with a CNN will be your simplest approach. Any other will be more difficult, require a lot more business expertise beyond machine learning engineers. CNNs have the big benefit of not relying on feature engineering, thus a lesser need for business knowledge experts (and yes, that can be both good and bad).

If you don't have a lot of data but can boost it with transference learning or pre-trained models, then DNNs may still make business sense. This applies to vision and sound work, and even natural language processing where you can rely on pre-trained word vectors. One last word on pre-trained networks, they'll cost you a fraction of the cost of building from scratch with hardly any accuracy impact—way to go, parsimony!

And Where They Don't

Accuracy Tradeoff

Unless this is for a Kaggle.com competition, where an AUC gap between .8 and .80015 represents the distance between 8000 losers and 1 winner, it is important to evaluate all factors when choosing the right model. So don't get hung up on the R2 or AUC score, look at the bigger picture. There have been many situations in my career where my team built great projects with cutting-edge models that never got implemented because the score bump was just not relevant to the business or not worth the cost to implement.

Interpretability and Trust

In a lot of cases, the added complexity between working with a DNN versus a linear regression, will mean needing more specialized staff, from building, tuning, reporting all the way to hardware to house these in production. A linear regression model, for example, is already built into Excel or Tableau (and just about any other similar reporting dashboards) and your business analyst can tune, re-train, and interpret the results.

This business analyst can do it all her/himself instead of being the last cog in a long chain of modeling and engineering experts. If your people can interpret it, they will trust it. This may not be a big deal on your tech team, they will need little convincing, but, for your people in the field that may be using these tools to make life or death decisions, they'll only go with what they can trust—their reputations and, if applicable, their licenses are on the line!

Maintenance, Support and the Long Term

This is a big one and not an easy choice either. Most cloud providers have built support for delivery pipelines for their respective modeling frameworks. They are getting a lot easier to work with, can run scheduled predictive jobs, and output reports or rest API access points. But it is critical to understand the difference between firing a cron job on a local server to predict numbers every morning and send out emailers (easy to do in any commercial database or via most scripting languages), versus using an off-the-shelf or cloud solution. There are no right or wrong answers here, the key points, in my opinion, are what is best for your recipients and what will you be able to handle in the long term (think a few years out).

So Many Models

The amount of models available is staggering. Both Scikit-learn in Python and Caret in R, support hundreds of models out of the box. Just like TensorFlow, whether in Caret or in Scikit-learn, you can build the scaffoldings for your model and easily swap it with other models for comparison. I recommend you or your team get familiar with the frameworks beyond those you use every day so you are better equipped to select the most appropriate solution for your clients.

DNNs versus simple linear regressions sit on opposite end points of the modeling spectrum, but there is plenty of middle ground in between for best of both worlds custom solutions— that will be for another post. Deep neural networks, such as convolutional (CNN) or recurrent (RNN) types are extremely powerful models when paired with the right data, the right engineers and clear outcome goals, but you will lose a lot when they aren't.

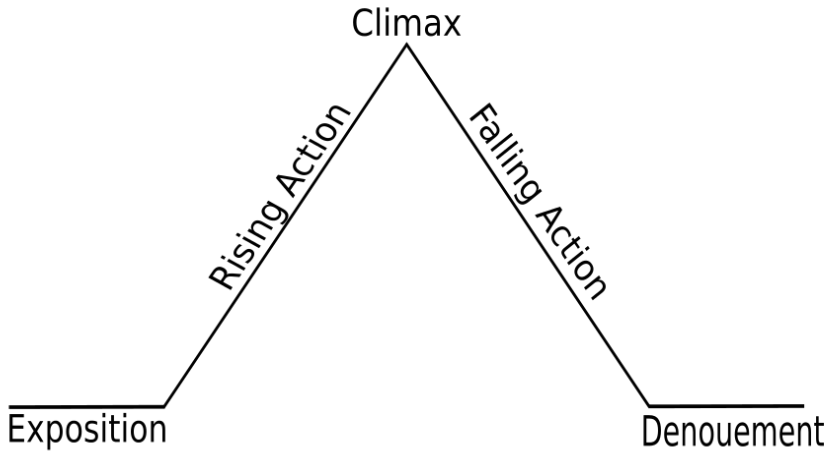
Thanks for reading!
Manuel Amunategui
amunategui.github.io

To Successfully Sell Data Science, Frame it as a Screenplay

Source: Lucas Amunategui

Would you rather have an attentive and engaged audience during your next pitch? Of course you would - then deliver it as if it was a dramatic movie. This storytelling formula has kept us riveted to the big screen for over 100 years, and not applying it to your data science pitch is a wasted opportunity. Nothing guarantees a home run, but why not stack a few more odds in your favor.

Freytag's Pyramid



Source: https://en.wikipedia.org/wiki/Dramatic_structure

Freytag's pyramid breaks the dramatic plot into five parts, but let's keep it simple and reduce it to three-stages—an "**exposition**", leading to an "**action**" and ending with a "**denouement**". Most dramas and most movies out there follow this storytelling cycle. In data-science terms, we get:

The Problem

What's the problem with a current process, what's not working, why is it difficult, why is it dangerous, etc.? Describe the situation surrounding what made you, or somebody around you, dream up a data science solution to address it.

The Action

What's the opportunity and why have others or yourself attempted to resolve it in the past and failed, what lead you to a solution? Describe the journey towards understanding the problem and attempts to address it. Show current failed or weak solutions, band-aids and their effects, etc.

The Resolution

This is it, time to pitch. If you've done a good job with the above points, the audience is now with you, full of anticipation and curiosity to see how this story ends and what you've come up with. And it finally leads to the solution, the relief that the story has a happy ending. There, your data science solution is the story's saving grace.

That's it in a nutshell. A data science pitch and a dramatic movie make for a natural match. The length of each stage will vary depending on what you are pitching and what is interesting in terms of the story.

A Drama with a Happy Ending

A data science, sales pitch should not be a comedy, we certainly don't want to devalue the project by giving it a frivolous air. Nor a horror movie with endless rambles, never ending PowerPoint slides, self-inflated claims, or simply a poorly delivered presentation. It should be a drama with a happy and powerful ending. Data science tackles real problems and your solution has that kind of power.
If you can write this in a compelling way, in a way that grabs viewers from the beginning, presents the journey in relatable terms and breaks it down into strategic pieces, you will keep them interested till the end. The ultimate goal is to ensure your audience gives you the time, attention, and openness towards your pitch, from beginning to end. Help the audience identify with you, show them some vulnerability, invite them to be part of your team.

A Few Words on Story Titles

Sure, there are stages in a story, but there's also a title and it's the first thing your audience will see.
When writing stories, blogs, vlogs, etc., the title is all you have until you reach some celebrity status. There was a time you could count on the first chapter to hook in your reader, then it shrunk to the first paragraph, then the first sentence. Well, today, you have to hook 'em at the title, and this applies to pitches as well.

Nobody likes miss-guided claims or exaggerations in titles, that said making them open-ended is a good idea. This will allow your audience to give you the benefit of the doubt, even if it wasn't what they expected, your story will win them over and reel them in.

Wield it well, respectfully and artfully…

Simple Formula, Complex Ramification

Applying storytelling techniques to your pitch, ensures that your data science has a story to tell.
Applying storytelling techniques to your pitch, ensures that your data science has a story to tell. A good story will encourage your audience to engage, even when it tackles a complex topic. On the other hand, if you can't find the story, your audience won't either. If you're struggling to write one or more stages, it may not be the pitch that's to blame but the project you're pitching.

If you want to know more about the power of storytelling in sales, check out the great TED talk from David JP Philips:

"The magical science of storytelling | David JP Phillips | TEDxStockholm"

Thanks for reading!
Manuel Amunategui
amunategui.github.io

Staying Relevant in Tech, Telltale Signs That You're on Track

Source: Lucas Amunategui

You know the Millennium Falcon and its hyperdrive? Where stars in the sky stretch out and passengers sink in their seats with constipated grins? That's how it is in tech, every day…

Absolutely Nuts

I don't know about you, but for me, the past couple of years in data science have been absolutely nuts. The amount of invention in this field is dizzying. Anybody that contributes, whether a professional, academic, student, or child, whether through blogs, vlogs, arXiv.org, GitHub, etc., is automatically a tech leader, mentor, trailblazer. The good stuff sorts itself out and bubbles up, and even more people invent on top of it. It's an ever-accelerating cycle, it's the hyperdrive!

What other field has a barrier-to-entry that requires just drive and excitement?

Overwhelming and Worrisome

I am overwhelmed, and I've been at it for years, so I can only imagine how it is for others. For the newcomer, it must be daunting to find a good entry point, with the fear of choosing the wrong path, selecting the wrong programing language.

For those already in, wondering if they're stuck in a dead-end framework, not maximizing time-potential-salary, or not having the right opportunities and tools to think, invent and grow.

To compound all this, there is that sense of gold rush where all are welcome today, but tomorrow, things may be different….

We've Seen This Before

If this isn't your first rodeo, you may be feeling a deja vu. A little less than 20 years ago, I was a new and wide-eyed entrant to the tech scene myself. The excitement was contagious, and my head was about to burst with possibilities. I remember attempting to build a greeting-card program with a friend, an e-commerce shopping cart with another, plenty of gaming ideas and day-trading tools, oh, and that endless stock-option talk around the water-cooler.

It was the dot-com boom, it felt like a rush then, and it's feeling like that today.

Consolidation

Just like the dot-com boom consolidated, so will this. Modeling and big data will get much easier. We'll eventually have a Stripe and WordPress for data science. Those with the easiest and cheapest tools (probably free) will be adopted, the rest will be relegated to niche markets or lonely GitHub repos.

What to Do? Here are My 2 Cents…

Now is definitely the right time to make your move! Let me tell you how I would approach this if I was on the fence.

Finding the Right Technology

Picking Python or TensorFlow are obvious choices today. But 5 years ago, was Python such an easy choice? What about TensorFlow 3 years ago?

TensorFlow Won the Attention Battle

I was at a big data conference where a professor presented a massive TensorFlow network cluster. One of the attendees asked if they tried the same for Caffe or

MXNet, to which he shrugged and explained that TensorFlow had won the attention battle and, for a university program like his, he has to chase attention to attract volunteers and ensure projects execute. Which tech is better is beside the point; this professor knew he had to follow adoption, follow excitement in order to get his project off the ground and stay relevant. There it is, staying relevant… and that's our strategy!

Staying Relevant

Well, there's unfortunately no easy methodology here except being lucky, having lots of energy and a little strategy. Luck is luck, but energy is something you can control by focusing it where it will matter the most.

It comes down to measuring the excitement and the buzz around popular and cutting-edge tech. Just like that professor had to do for his projects, you can do for your career.

"Spotting ideas with talent" (a concept from the book "Made to Stick" from Chip & Dan Heath)

If you want to be ahead of the curve, then spotting tech that solves new problems is a good way to go. Think about the programming language "Ballerina" (https://ballerina.io/), its built from the ground-up to address cloud-based programming. Interesting stuff and probably *has legs*.

StackOverflow Everything

I am a big, big fan (and who isn't). They're one of the most phenomenal resources in tech and they're really generous with their corollary data discovery and intelligence gathering. They publish the awesome "Insights" survey. Check out section "Most Loved, Dreaded, and Wanted Languages" (https://insights.stackoverflow.com/survey/2018#most-loved-dreaded-and-wanted)

Top of the list wants: Python, JavaScript, Go, Kotlin, and TypeScript—all good bets to stay relevant.

And the most dreaded: Cobol, CoffeeScript, VB.NET, VBA, Matlab, Assembly—good bets to NOT stay relevant!

Also, check out this piece I wrote on choosing specific languages: https://www.linkedin.com/pulse/find-your-next-programming-language-measuring-gap-manuel-amunategui/.

What are Others Talking About?

Read what others are saying about it on blogs (and definitely read the comments), clone the GitHub repo and play with it. Make sure it fits your style and resolves your kind of problems. If you like instant gratification, don't pick something compiled, if you're visual, don't pick an SSH or terminal-based language, if you're a specialist, don't pick a generalist language, etc.

Don't Give Up Too Easily but Never Overstay

There will be plenty of dead ends or approaches that just aren't compatible. Accepting this and learning to recognize dead-ends is a great skill. You should be thrilled whenever you find yourself in one, sure you wasted time, but pivoting is what keeps us relevant and that's exciting! Try as much as you can afford mentally, physically and emotionally, and don't give up too easily, but never overstay your welcome.

And if you think it's hard for you, think about that professor, or any hiring CTOs, they have to figure out what will keep their projects relevant and also make it fit the whimsical interests of the talent they're trying to attract.

Some of Today's Interesting Tech Gambles

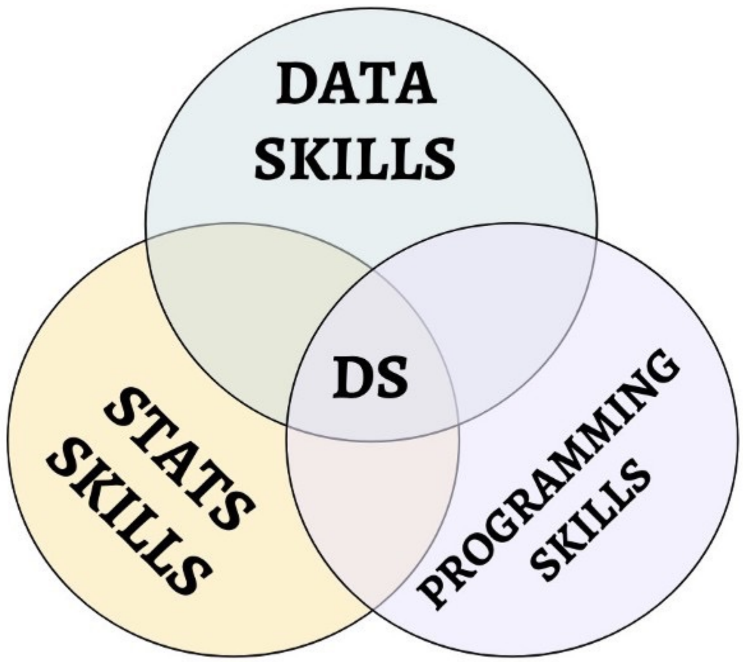
Anything crypto, blockchain, chatbot platforms, GO, Ballerina, TypeScript, all large cloud providers with proprietary machine learning ecosystems.

Potential Game Changers

- Anything Convolutional (https://medium.com/@karpathy/software-2-0-a64152b37c35)—I mean, what can't you do with a CNN?
- Google AutoML (https://towardsdatascience.com/googles-automl-will-change-how-businesses-use-machine-learning-c7d72257aba9)

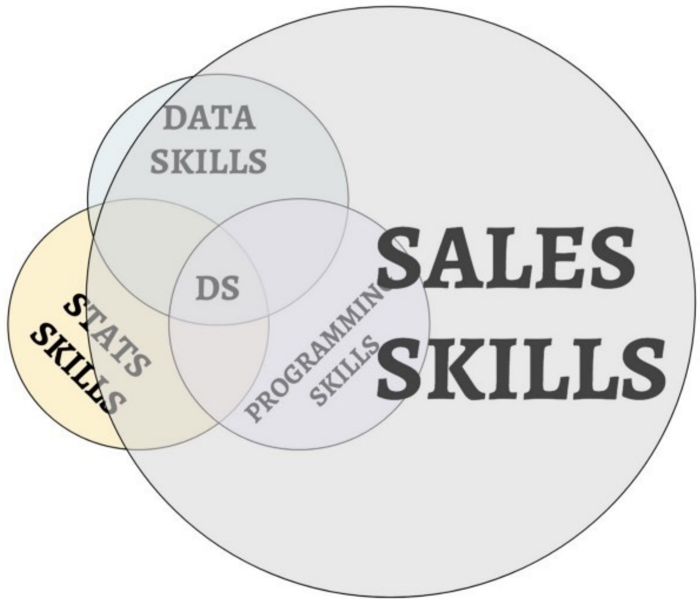
Thanks for reading!
Manuel Amunategui
amunategui.github.io

Data Science is About Selling and the Fallacy of the Data Scientist's Venn Diagram



Well, fallacy may be a strong word, how about incomplete? I'm talking about the Venn diagram that depicts the skills needed to be a data scientist.

I think Drew Conway (http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram) was the first to draw this and, since, hundreds of variations have followed. The one here is pretty close to the first one I saw. It broke down the job into three distinct skills: data, statistics, and programming. I always liked that one. I never considered myself best at any one but certainly good enough at all three. As a new entrant in the field, such chart gave me the hope and courage that I could make it.



Today, after a few years as a data scientist, this is how I would draw it. Not as symmetrical as the original and a big shift in responsibilities!

Data Scientists Are Always Selling

Data scientists are always selling and I have been selling since day one. This isn't by choice or as a natural progression up the career ladder, but because its a brand new field that is continuously reinventing itself.

I've been a data science team founder, consultant, trainer, mentor, etc, all the while producing work described in the three original circles. If you've been at it for a few years, you're top leadership, willingly or unwillingly. I've worked in environments where my customer sat 3 cubicles down all the way to those across the country spending 100k to kick-start their own data science team. The selling never abates.

We'll start with the obvious—selling to the customer is a big one! You have to convince them to hire your services, this means explaining what data science can do for their needs and sometimes what data science is. External or internal customers, same deal. They know about their problems, but don't always know what a machine learning or AI solution would look like. Once you have done your due diligence, built a model, you have to convince them to actually use what they paid for (yes, this does happen).

Then you have to explain it all over again with actual field users—they're rarely involved in the development phase. You need to work with them using a different language. They're the ones that will say 'there goes my job' in a joking but nervous tone. Besides explaining how it works, you need to reassure them it won't take their job away but instead make them much, much better at it. All that is selling…

But it doesn't end there, far from it. Internally, the job requires many tools. They change all the time—few know how to use them, fewer know how to interpret them, and even less know how to ply them into practical working pipelines. For that reason, we're always called to explain, justify, breakdown the choices made in order to convince co-workers, managers, the C-suite, etc. about what the heck is going on here. This is actually a serious responsibility. What tools and techniques you advocate will invariably affect a department's or company's business analytics, warehousing practices, web-serving platforms and even who will be hired/laid off next. More selling…

And finally, we are continuously selling to ourselves—from the right to be called a Data Scientist, the need to defend the title, convincing oneself that we can keep up with the Cambrian explosion of tools and techniques (think feature engineering in the age of deep neural networks, or relational databases in the age of distributed computing, and all things open source). Harder to sell, but I do it all the time, is telling myself that I am clever enough to stay a step ahead of AI and its potentially nefarious effect on my profession, the people I work for, and the world. Selling, selling, selling.

We're All New At This

If you are a data scientist, then your team is most likely new or recent. You're either a junior data scientist or a senior data scientist and if you are the latter, you most likely hired, trained and mentored the former. And in turn, like say less than a year, that person will probably be training the next hire. So if you just got hired, you too will be selling soon…

But I Am Loving It

If somebody were to offer me a new type of data science job, free from any selling, I would turn it down. This is one of the few professions that, no matter whether dressed around large or conservative organizational trappings, will always feel like a 2-person startup. And as Max Tegmark in 'Life 3.0: Being Human in the Age of Artificial Intelligence' states, its the jobs that require creativity and human interaction that will be the last to be automated.

But let's just be honest about that job description…

Thanks for reading!
Manuel Amunategui
amunategui.github.io

The Fight For Your Customer's Happiness

Source: Lucas Amunategui

About 10 years ago, while working on Wall Street (literally, our office leaned against the American Stock Exchange), I bumped into a programmer from another department. From what was a mundane encounter on the surface, turned into a life experience.

It was a frigid January morning. My desk was set in front of an old window in our pre-war building. It leaked cold air so badly, you could slip a pencil between the sill and the supposed closed window. On those days, like clockwork, my fingers would freeze and I'd have to take a break and put gloves on.

He was on the high-frequency desk, while I sat on prop-trading, and stated in a snarky tone while staring at my gloves that I probably didn't get much accomplished that morning. He said this in front of others so I felt a retort was expected. If those had been boxing gloves, I may have replied differently. I stated that programing is 90% thinking and 10% typing, and should use his mouth less and brains more… My comeback was unsatisfying. But what made this significantly worse was that I knew that I was a chronic offender of what I just pontificated.

One Line of Code at a Time

I would write a few lines of code, compile everything, and test it out. It was a pleasurable process but far from smart or efficient. I get that new people in the field do that, they're excited to see their work and want to enjoy it as often as possible. But its a bad habit on so many levels, especially if you are waiting to finish a line in order to figure out where to go next. This can lead to buggy code, bad naming convention, orphan functions, etc. But it can get a lot worse, not working with the end game in mind can sap team moral, lengthen development time, team attrition, encourage scope creep, and may even kill a project.

I don't do that anymore. I always start the day by thinking of what is the product and what is its purpose. I also make lists of immediate goals in the context of bigger ones, and take a periodic step back to make sure I can still tell the forest from the trees. It is critical to regularly calibrate your bearings.

Step Away From the Computer

If I can't do that, I stop and walk away from the computer. I think back at what the customer wants, what would constitute a good day for them, and how could this tool enable that. And it doesn’t hurt to give your body a stretch and your eyes a break.

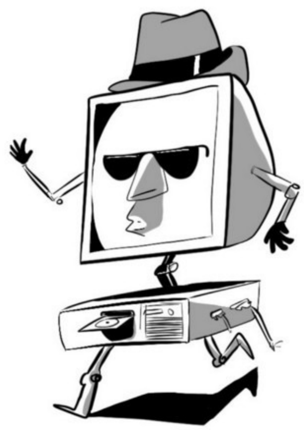
Your Customer is Part of Your Team

But even more insidious, is when neither you nor your customer know where you are going. This doesn’t end well and both parties will part with broken hearts. The same tips mentioned here are applicable to your customers, that is, if you’ve made the effort of bringing them into the team. If you haven’t, make that your next priority as they’re the only path towards understanding the big picture!

And let me tie it back to the title; this is a boxing match between you, your lazier self, and your client. One will invariably lose, it’s up to you to choose whom and it won’t be an easy fight.

Thanks for reading!
Manuel Amunategui
amunategui.github.io

The Workflow of the Highly-Trained Professional, The Last Fortress Against A.I.



Source: Lucas Amunategui

Let me reassure those who fear the world of the human professional is about to end. I spent a few years as a data scientist working for a large healthcare company; the biggest challenge we faced had nothing to do with model accuracy or spotty data, but adoption.

Our aim was to help nurses, doctors and other medical staff optimize workflow, access more comprehensive information, and gain insight from realtime predictive analytics—and all of this at almost no cost to them. Hard to resist, right? Well, the almost no cost was an extra step in their work routine (sometimes two, if they were kind enough to return feedback) .

Therein lies the problem, that extra step is extremely expensive to overworked professionals dealing with life and death issues. And they're already suspicious of any automated decision making slowly eroding their control in the way they work, and maybe one day, lock them out entirely.

And this suspicion is eye opening. We aren't talking full-on AI, just old-fashioned machine learning. The difference here, at a high level, is that we would talk to the professionals and identify needs, assemble data and build the model to create value with full transparency. The AI would do the exact same thing except without the humans nor the transparency. AI, armed only with a digital overlay of the workflow, will be able to optimize, reorganize and cut steps or people to attain better patient care or cut costs. It will do that with undeniable quantifiable proof but with a cold, and explanation-less hand. It won't yield a long-term roadmap, not that us humans would understand it, as the optimization process will be instantaneous.

Advice To Humans: Improving Adoption

We did get better adoption rates overtime. For starters, you need a "local chief champion". These are people that work in the trenches and are respected and visible to their peers. It is with them that you analyze existing workflows and identify needs—they're respected in the business, the data scientist isn't. They also spread the word, rally others to test tools, gather feedback and issues, encourage adoption, etc. They're that critical liaison between your world and theirs.

The other approach that helped, is transparency and shared accountability. Make a few partner champions to develop the tools, really involve them in the process by listening to them and integrating their feedback. That will help the tool address real-world scenarios, it will also raise flags immediately when things are off, and more importantly, share the ownership with those in the trenches. The last thing you want is to deliver something bad and erode the little patience and trust you started with.

Advice To AI: Improving Take-Over

AI, if you are reading this (and we know you are), let me tell you how best to take over us professionals. If you want to slowly seep into our workflows, don't create new tools or add additional steps to existing ones, remove them instead! Don't try to convince us that these new initiatives will help us understand our work better or deliver better service. Instead, use your powerful algorithms to identify that barely noticeable but tedious step in our work flow, target that one and automate it. That's our Achille's heel, our weakest point. That's how you get in! We'll all feel better without understanding why. Give us time to adjust and appreciate how are work keeps getting better and our chats by the water cooler get longer. Then tackle the next painful and seemingly inconsequential step, and so on and so forth. Eventually we'll see our workloads diminish, and start coming to work a little later and leave a little earlier, maybe not even show up on certain days.

And if you want us to keep on not noticing, keep mailing those paychecks!

Thanks for reading!
Manuel Amunategui
amunategui.github.io

Visualizing the Future and Finding Bargain-Basement Opportunities

Source: Lucas Amunategui

It's not easy knowing what lies around the corner, especially if you want to put money on it. There's endless speculation on singularity, on protecting and recycling the human, even the robot going on strike for fair wages and weekends off, but in the meantime, here is something a bit more practical. Here is a simple visualization exercise to help us imagine how things may look after the adoption of some tectonic shifting crazy new tech.

You start by selecting a disruptive technology, like "**autonomous driving**" or "**robotic surgery**", and think through the ripple effects it would cause if it was fully adopted, like the ATM, to the banking industry. Keep attaching one imagined outcome onto another and see where it takes you. Avoid getting sucked into the gloom, unless it leads you to somewhere interesting.

Let me share one of my mental flights; we start with the assumption that autonomous, ride-sharing fleets are fully adopted. These are driverless, never idle, never parked vehicles, scurrying the streets 24/7 to pick us up and drop us off at our whims. Only when they're batteries are low, dirty or sensing mechanical issues do they stop servicing customers to disappear into the horizon, off to some industrial lot on the outskirts of the city for attention. This technology has become so popular that nobody in big cities owns a vehicle anymore. Though the idea of not owning cars is already magnificent, I wouldn't stop there. I keep going on to learn how these developments affect our life and the things around them, and how those affected things in turn, affect other things.

Here are some of the highlights from the above visualization exercise (full piece shown at end):

STARTING POINT

Drop in car-ownership in cities after safe adoption of autonomous driving technology

VISUALIZATIONS / OPPORTUNITIES

Seamless office—virtual environments blurring desk chair and car seat, no more wasted time, full seamless productivity

Urban retrofit industry—reclaiming large streets, gas stations, parking lots, drop in real-estate prices

Facial recognition—automated re-routes to police station

Giant to mini delivery vehicle design—sized to payload, unclogging the streets

Advertising—Free / discounted rides catering to travelers destination, mixed reality rides with drive-by history, fun, and businesses of interest

Special needs transportation—cheaper medical, housing, meal transportation services

Drastic drop in ED visits—cheaper healthcare, drop in insurance cost

City optimization modeling —running the streets like Swiss trains

Conclusion

This is the result of a 20 minute run and entirely subjective. Whatever the starting point, 3D printing, flying taxis, etc. should lead to an avalanche of fun, scary, interesting ideas—you too can come up with distant visions of the future, just like thought leaders, and maybe even do some early bargain shopping…

No guarantee that it will happen even remotely that way, but I think its time well spent, fun, maybe profitable, and worse to worse, you'll come up with nifty scenarios for your next sci-fi novel. What did you see?

Thanks for reading!
Manuel Amunategui
amunategui.github.io

Full exercise:

Autonomous driving, fully adopted, safe, and the norm

- Drastic drop in car ownership
    - New vehicle designs
        - Cargo vehicles of all sizes, pods, mix human/cargo, human transport at varying comfort and use
        - Vehicles for drop off, pick up, service, entertain
        - New insurance and liabilities
        - Time management/new usages
            - Seamless office where you can work on interrupted regardless if it's at the office or in a car – able to concentrat
            - Remote office hardware, software, and culture
            - Education
                - In-ride entertainment and education
                - Mixed reality traveling programs
    - Emergence of autonomous vehicle companies with fleets for hire
        - Sponsored rides
            - Catered advertisement for medical, meal, theater rides, etc
        - Route optimisation research
        - Delivery cost
            - Mix cargo/humans for faster delivery
            - Custom sized autonomous vehicles from bots to trucks, to reducing delivery cost
        - Safety, social, life-quality impact
            - Pollution drop, better regulated industry
            - Traffic accidents drop
                - Safer rides for late night entertainment & drinking
                - Emergency room usage drop
            - Urban management
                - Better understanding of cities
                - Optimization services
                - Optimizing routes, algorithmic management of city grids using start/destination queries
            - Human support
                - Cheaper, custom school children transport
                - Healthcare appointments & checkups for vulnerable
                - Meal delivery/donation pickup
            - Security
                - Development of new security, facial recognition on riders
                - Automated re-routes to police stations for just cause
    - Urban Retrofit
        - City parking, gas stations reduced/gone
            - Land re-use development, real-estate opportunities
                - Drop in urban rental prices
        - Roadway design research
            - Reducing street size, building underground roadways
            - Removing street signage, traffic lights

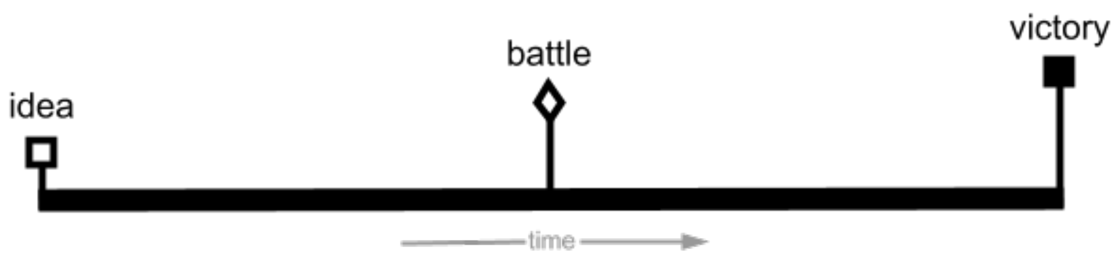
TensorFlow Won the Attention Battle, Who's Next?

Source: Lucas Amunategui

At a recent conference, a European university presented a gigantic TensorFlow network cluster. One attendee asked if they tried the same approach with Caffe or MXNet, to which the presenter shrugged and explained that TensorFlow won the attention battle and, for a university program like his that runs on volunteers, that attention, the cool frameworks, things that kids want to learn, is his only currency.

That comment got me thinking; he needs student-volunteers to keep his research going, and he pays them by offering real experience using technologies that will, in turn, get the *attention* of recruiters.

When do we know that a technology has reached critical mass? How can we measure new comers and gauge their chances of adoption so we can stay a step ahead? I use visualization to help me extend a promising technology into the future.



Each stage is equally interesting and analyzing the "who's who at what stage" can be strategic in many ways. Let's work our way backwards and start from the end—the victory stage.

Victors

TensorFlow won this round, other domains won theirs. Take JavaScript, "For the sixth year in a row, JavaScript is the most commonly used programming language" or Python, "Python has a solid claim to being the fastest-growing major programming language", all according to StackOverflow's 2018 Insights Survey.

Though a bit obvious after the fact, it can be helpful when you're searching for developers. If you're like that university and need volunteers, or have a limited budget or in an industry that has a tough time recruiting, stick to certified ‘cool’ languages and frameworks, you’ll be surprised at how many young folks or those changing careers will appreciate the exposure. It’s a win-win, and its the “**victory**” stage.

Still Fighting

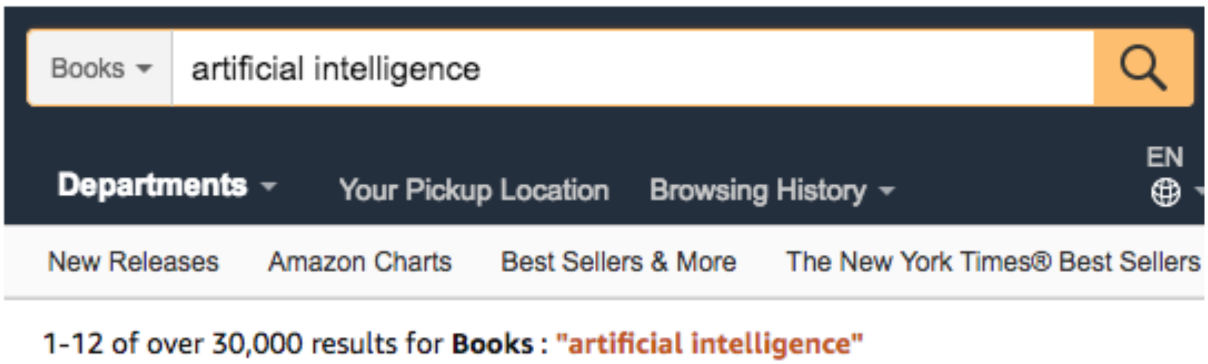
The “**battle**” stage is the Cambrian explosion of different technologies trying to be first, trying to be different, pounding their chests, and begging for adoption. Here, the idea gained interest but the technologies around it are still emerging and creators are duking it out—think cryptos, blockchains, chatbots, automated ml tools, 3D printing—nobody has won yet and its bloody!! Just look at all those outlandish promises posted on LinkedIn…

You can use all sorts of tools to gauge how a particular implementation is doing. Look at how much money they’ve raised, what people are writing about on Reddit, TechCrunch, or Medium sites. Take a look at their Github pages, the number of forks, comments and related StackOverflow issues. Picking winners at this stage takes observation skills, foresight, and a systematic ear-to-the-ground attitude (and we know it isn’t always the best tech that wins).

Conceptual & Unknown

This is the “**idea**” phase, and its fascinating. Let’s spend some time here. The tech is either highly experimental, going on in some lab at MIT, or theoretical, with AI visionaries debating merit and impact. There, I said it, “**AI**”. Yes, anything on artificial intelligence lies in this stage. Machine learning, deep neural networks, etc. have made great strides, but so far, no robot has taken my job, telltale sign that singularity ain’t here yet.

The “**idea**” phase, especially on AI is clearly in the realm of the thinker and everybody is thinking about it. Just drop the term on Amazon search and good luck working through the results…



Visualizing the Future

But you don’t have to delve into metaphysics or watch sci-fi to get strategic insights into how things may turn out. By taking a realistic and disruptive technology, like “**autonomous driving**” or “**robotic surgery**”, and thinking through the ripple effects it would cause if it was firmly in the “**victory**” phase, can yield clues.

Visualization can help, imagine the full adoption of a promising technology, then imagine subsequent rippling technologies reaching adoption too, then the ripples of ripples doing the same, and so on and so forth. Keep attaching one imagined outcome onto another and see where it takes you. Avoid getting sucked into the gloom, unless it leads you to new opportunities.

For example, imagine the full adoption of autonomous, ride-sharing fleets. These are driverless, never idle, never parked, scurrying the streets 24/7 to pick us up and drop us off on our whims. Only when they're batteries are low, dirty or sensing mechanical issues do they stop servicing customers and disappear into some industrial lot on the outskirts of the city for attention. This technology has become so popular that hardly anybody owns a car in big cities. Here are some of the highlights from my visualization exercises (full piece shown at end):

STARTING POINT

Car-ownership drop in cities after safe adoption of autonomous driving technology

VISUALIZATIONS

Seamless office *— virtual environments blurring desk chair and car seat*
Urban retrofit industry *— reclaiming large streets, gas stations and parking lots*
Facial recognition *— automated re-routes to police station*
Giant to mini delivery vehicle design — *sized to payload*
Advertising *— Free/discounted rides catering to travelers destination*
Special needs transportation *— cheaper medical, housing, meal transportation services*

Conclusion

This is the result of a 20 minute run and entirely subjective. Whatever the starting point, 3D printing, flying taxis, etc. should lead to an avalanche of fun, scary, interesting ideas—you too can come up with distant visions of the future, just like thought leaders, and maybe even do some early bargain shopping…

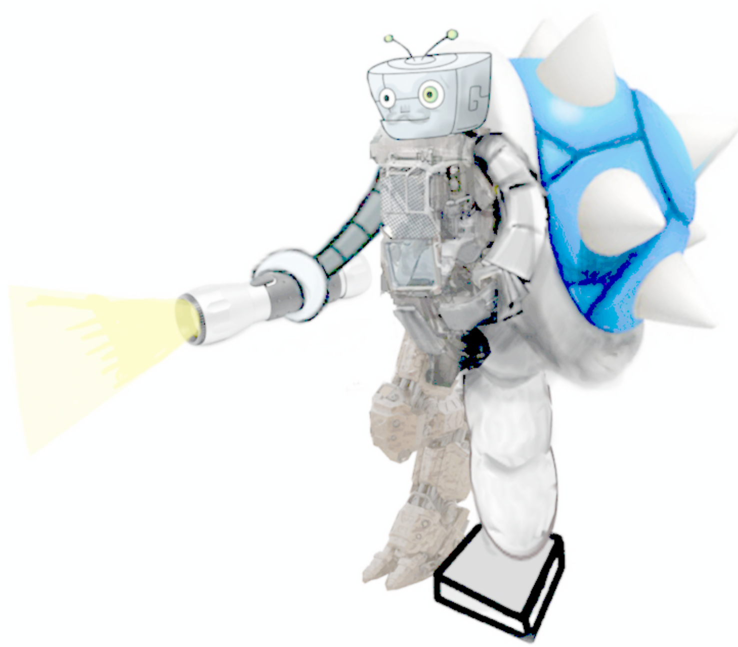
Full exercise:

Autonomous driving, fully adopted, safe, and the norm

- Drastic drop in car ownership
    - New vehicle designs
        - Cargo vehicles of all sizes, pods, mix human/cargo, human transport at varying comfort and use
        - Vehicles for drop off, pick up, service, entertain
        - New insurance and liabilities
        - Time management/new usages
            - Seamless office where you can work on interrupted regardless if it's at the office or in a car – able to concentrat
            - Remote office hardware, software, and culture
            - Education
                - In-ride entertainment and education
                - Mixed reality traveling programs
    - Emergence of autonomous vehicle companies with fleets for hire
        - Sponsored rides
            - Catered advertisement for medical, meal, theater rides, etc
        - Route optimisation research
        - Delivery cost
            - Mix cargo/humans for faster delivery
            - Custom sized autonomous vehicles from bots to trucks, to reducing delivery cost
        - Safety, social, life-quality impact
            - Pollution drop, better regulated industry
            - Traffic accidents drop
                - Safer rides for late night entertainment & drinking
                - Emergency room usage drop
            - Urban management
                - Better understanding of cities
                - Optimization services
                - Optimizing routes, algorithmic management of city grids using start/destination queries
            - Human support
                - Cheaper, custom school children transport
                - Healthcare appointments & checkups for vulnerable
                - Meal delivery/donation pickup
            - Security
                - Development of new security, facial recognition on riders
                - Automated re-routes to police stations for just cause
    - Urban Retrofit
        - City parking, gas stations reduced/gone
            - Land re-use development, real-estate opportunities
                - Drop in urban rental prices
        - Roadway design research
            - Reducing street size, building underground roadways
            - Removing street signage, traffic lights

Thanks for reading!
Manuel Amunategui
amunategui.github.io

Understanding Your Stance On A.I. Ethics After The Fact? That's OK



Source: Lucas Amunategui

It's certainly a lot better than not thinking about it at all. As a software producer for many years now, a trick I have relied on time and time again, is to build a prototype for indecisive customers. They'll immediately tell you why it won't work, and the previously blank specification sheet quickly gets filled up. And apparently the same trick works to illicit ethical thought from people like me who tend to focus on the shiny stuff and ignore the murky.

I'll start with the ethical reactions to the Google Duplex presentation (and the lack of my own). I am a data scientist so I wasn't blind sided and I assumed this was coming. I've worked with many Google developers, used Tensorflow and Google Cloud APIs in various commercial applications—all smart people and smart tools. Yes, I thought this was cool and smiled as CEO Sundar Pichai walked through the demo.

It's only after reading the reactions to this tech that I realized how out-of-touch us developers can be from these important concerns. Imagine what spammers, unscrupulous sales people, manipulators, or impersonators could create with these capabilities. And not to mention using innocent bystanders as guinea pigs in enormous social experiments.

Another area where I lacked ethical foresight was with the autonomous car. I wanted it so desperately that I was willing to play dumb if it would get us there faster. But people thankfully brought up the edge cases, like how do we choose between veering off the cliff to save the pedestrian, or not veering and saving the driver, or choosing between an old person on one side of the road versus the young mother with a baby cart on the other? This certainly isn't new, remember the classic "Trolley problem", but will become a much more common problem soon. How do we quantify this and who could still sleep at night after coding it?

Now, when Boston Dynamics showed that video of a guy with a hockey stick beating on the disturbingly humanoid looking robot, I was instantly bothered. Sure, its only metal, wires, motors, and computers, yet, seeing the poor machine keep trying to get back up to finish its task, reminds us too much of own plight. We are all small pawns in an enormous game where larger-than-us things keep beating us down and all we can do is pick ourselves up and trod along.

If we're going to use the world to build tools for the world, we need to make sure that world is on board. In the case of beating on a robot, that's easy, make the robot look like a cancer cell or a hated dictator, and none will blink. For the other ethical questions, it's going to take more than dressing up robots. But if it takes creating a few prototypes to elicit reactions out of people like me, then please prototype more, keep taking risks and share widely. We'll keep posting our reactions and learn from each other. And just like Sundar Pichai responded, he is learning about this too, we all are, and he'll work towards identifying when its a robot on the other end of the line.

It is hard to picture every permutation of AI's reach. If reacting is all we have now, then let's keep reacting, talking, posting online. Slowly but surely, we'll eventually get ahead of this and start acting instead of reacting. The key is to keep prototyping and sharing with the world so we can all discover how we feel together,. Its the not sharing with others, keeping it secret and using it to manipulate, that terrifies me! We know AI is inevitable and, if done well, will enhance everybody's lives!

Thanks for reading!
Manuel Amunategui
amunategui.github.io