CS 188

Guest Lecture

Andrew Mutz

November 24, 2020

Introduction

Who am I?

- CTO, Real Estate at Appfolio
- We make business software for vertical markets
- Currently working with multiple teams at Appfolio to build AI products
- Fun fact: I used to teach this class

For Today...

- Introduction
- Terminology
- Motivating example: hotdog vs not-hotdog
    - How should we make predictions when running?
    - How does training data flow through the system?
    - How hard is it to train the models?
    - How often do we need to retrain?
- Examples from industry
    - Lisa
    - Smart Bill Entry

Introduction

Why are we talking about AI & ML in a course on Scalable Internet Services?

Introduction

Why are we talking about AI & ML in a course on Scalable Internet Services?

- ML is a big deal: we can do much more than before
- Most of the work building ML systems in the real world is not ML

Why is ML such a big deal?

- ML systems are delivering breakthroughs in the fundamental capabilities of software systems:
    - The ability to understand the content images and video
    - The ability to understand human natural language (spoken and written)
    - The ability to play and win a game from only a set of rules
    - The ability to generate plausible images and language
- A naive reflection on these breakthroughs can see where they can be applied:
    - Organize your photos
    - Talk to Alexa
    - Better Dota adversaries
- But this is so much more...

Why is ML such a big deal?

- Today's products are the result of many constraints: what machines can and can't do
- These new capabilities **significantly** change the constraints that technologists work within
- If the form and function of today's products are the result of working within constraints, and many of these constraints fall away, the form and function of these products may change radically
    - Each one of these changes is an opportunity for each of you!

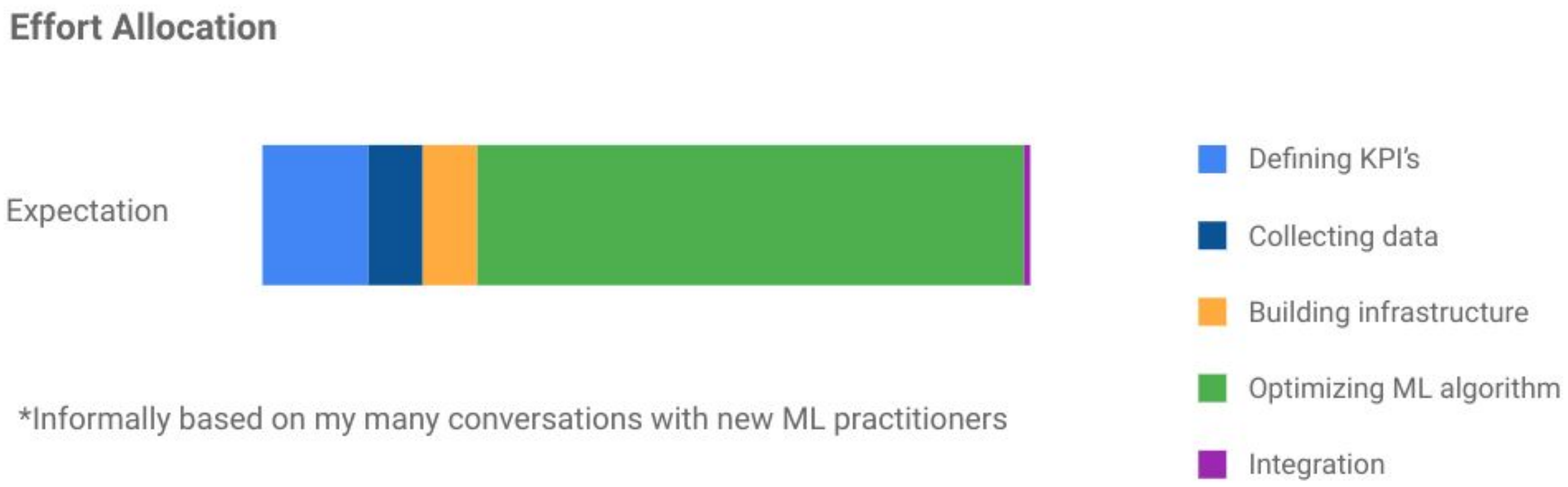
The Machine Learning Surprise

Most of the work in building real ML systems is not ML...

- "The Surprising Truth About What it Takes to Build a Machine Learning Product" by Josh Cogan*

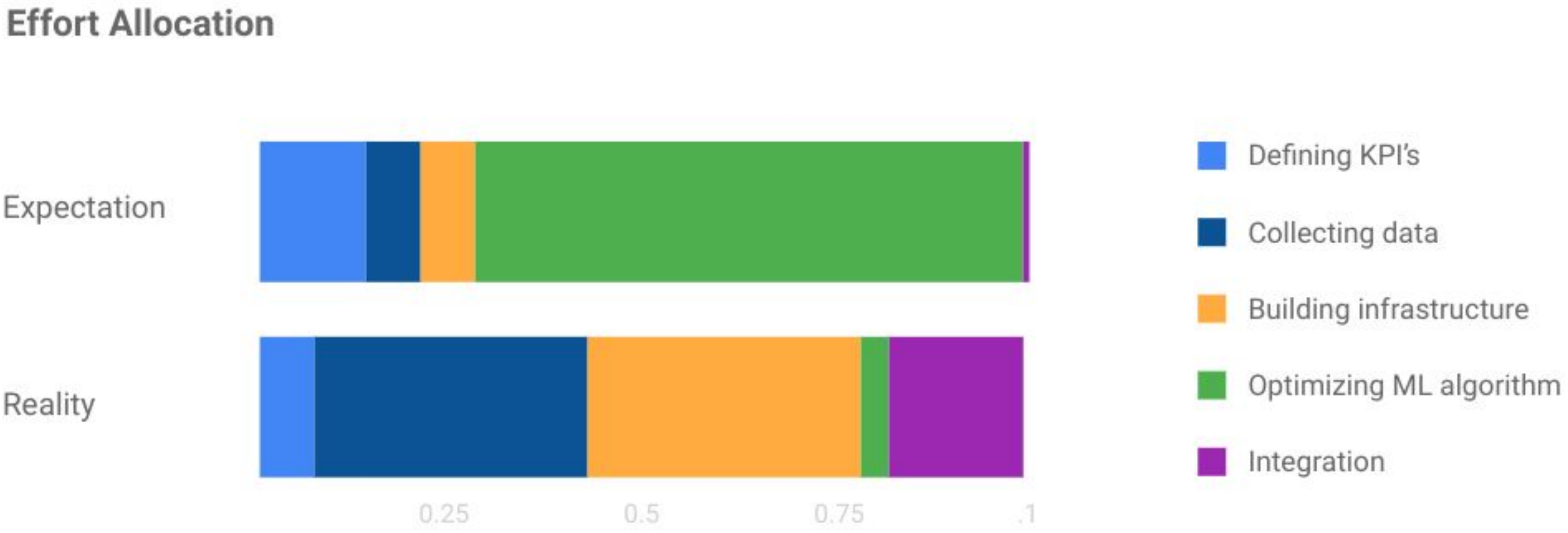
*https://medium.com/thelaunchpad/the-ml-surprise-f54706361a6c

The Machine Learning Surprise



https://medium.com/thelaunchpad/the-ml-surprise-f54706361a6c

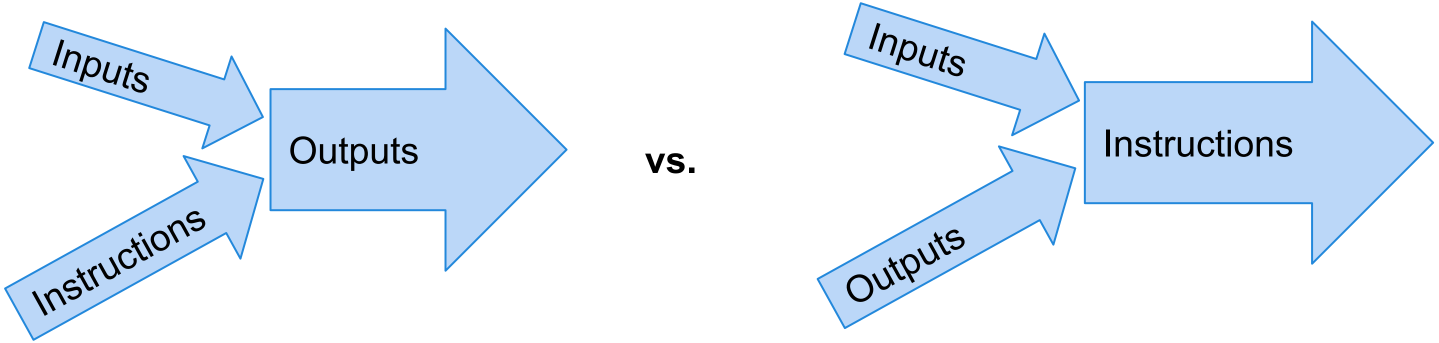
The Machine Learning Surprise



https://medium.com/thelaunchpad/the-ml-surprise-f54706361a6c

Terminology

- How does building software with machine learning compare to traditional software?
- Traditional software: Programmers told computers how to solve problems
- **Machine Learning**: Programmers give millions of examples of correct behavior, and the computer figures out the best rules



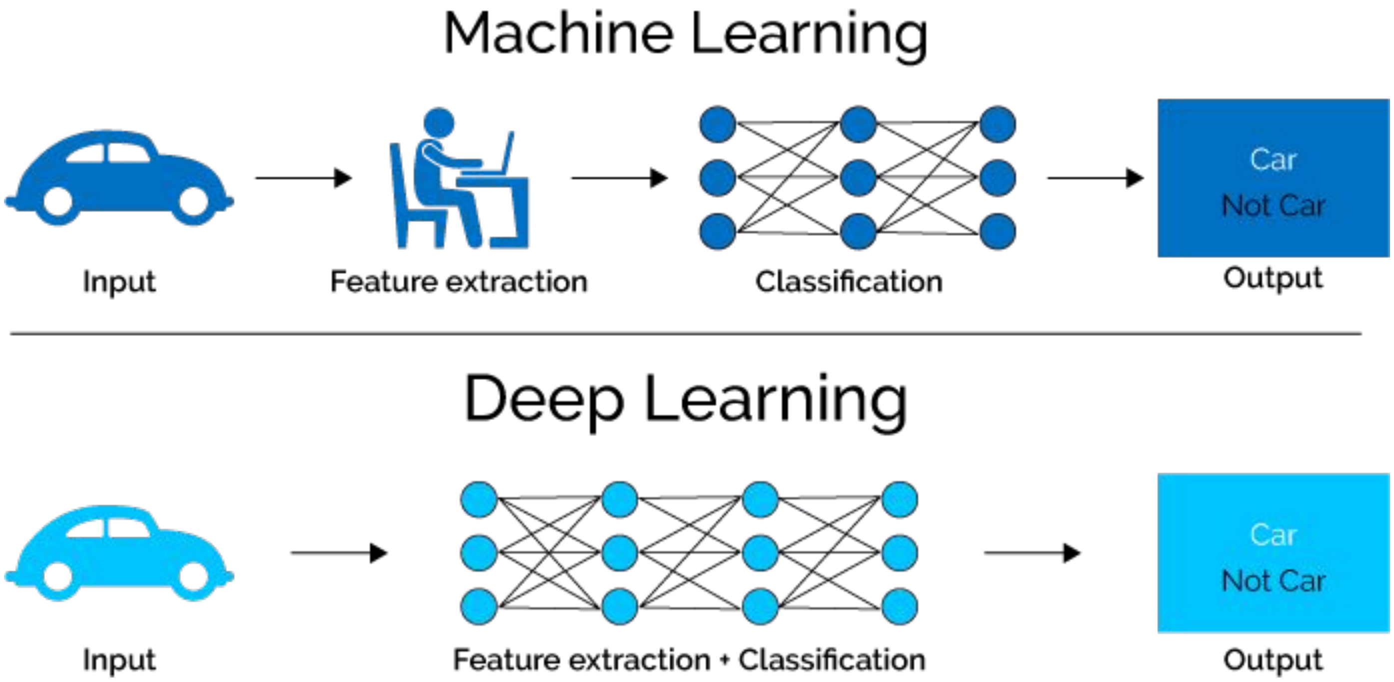
Terminology

- Training vs evaluation
  - Training is when you present a large number of examples to the machine, and it determines a set of weights that best map the inputs presented to the outputs (or "labels")
    - The output of training is referred to as a "model"
  - Evaluation is when you run a new set of inputs on a previously-generated set of weights in order to produce a new output.
    - The output of evaluation is referred to as a "prediction"

Terminology

- Deep Neural Networks
  - The source of the most impressive breakthroughs in recent years
  - Millions of connected artificial neurons, inspired by the human brain
  - Generally less emphasis on feature extraction
  - Accuracy scales well with size of the data set
  - Examples: CNNs, RNNs, LSTMs, GANs
- Traditional Machine Learning
  - Feature extraction and engineering done by the engineer
  - Generally easier to reason about and debug
  - Better performance on smaller data sets
  - Examples: Random Forest, Naive Bayes, Support Vector Machine

Terminology



From https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063

Where does ML fit in?

- So we have...
  - Training data examples (MB or GB)
  - Python code that, when given training data, generates a model (MB or GB) over minutes to days
  - Python code that, when given an input and a model, generates a prediction (usually small: bytes or kb), over milliseconds to a few seconds

- How should we fit this in to our existing web/mobile tech stack?

Where does ML fit in?

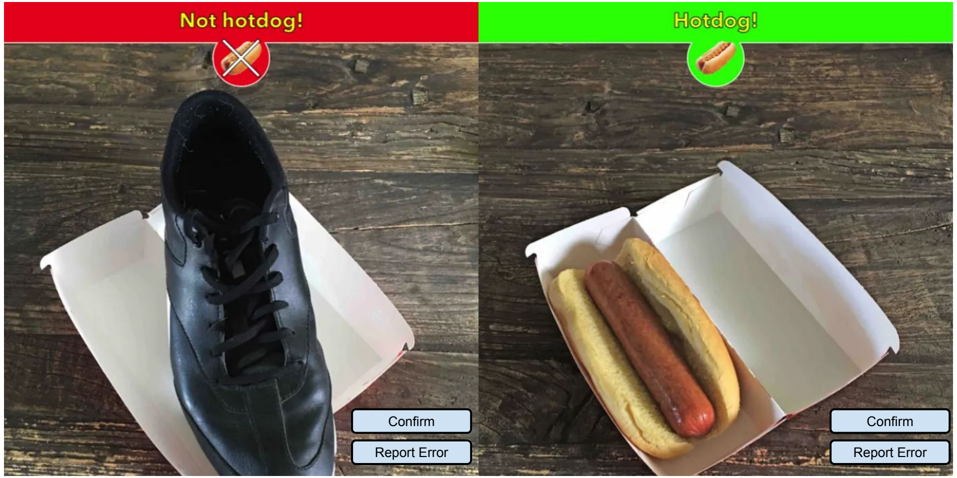
- In order to figure out where and how the ML fit in, we first need to answer some questions:
    - Inference: How should we make predictions when running?
    - Training: How does training data flow through the system?
    - Training: How hard is it to train the models?
    - Training: How often do we need to retrain?

- Let's look at a made-up example...

Where does ML fit in?

Hotdog detector

- User provides a photo
- The UI either says the photo is a hotdog or not a hotdog
- User interface allows the user to confirm or reject



How is hotdog detector using ML?

Let's talk about each of these questions as they relate to the hotdog app

How should we make predictions when running?

- There is a convolutional neural network that infers from each photo whether or not it has a hot dog
- Where should we run the model? What are our options?

How is hotdog detector using ML?

Let's talk about each of these questions as they relate to the hotdog app

How should we make predictions when running?

- There is a convolutional neural network that infers from each photo whether or not it has a hot dog
- Where should we run the model? What are our options?
    - We can run on the client
    - We can run inside our app server process
    - We can run in a separate process
- What are the advantages and disadvantages of each?

How is hotdog detector using ML?

Option 1: We can run on the client

- Details:
  - Run in browser via Tensorflow.js/ONNX.js
  - Build a mobile app and run natively
- Pros:
  - You can support a broader array of product experiences
    - What type of user experience would require this for catvdog?
  - Might see cost savings at high scale
- Cons:
  - Restricts your technical choices
    - Model size is limited by smallest supported device
  - Harder to build, test, monitor

TensorFlow.js

How is hotdog detector using ML?

Option 2: We can run inside our app server process

- Details:
  - Models are just a large collection of weights in memory.
  - At training time, serialize model to "pickle" format
  - At inference time, deserialize model to app memory
- Pros:
  - No integration pains
  - Can work well for small, rarely changing models
- Cons:
  - Restricts your choice of tech stack (python)
  - Heavyweight inference could affect other web requests

How is hotdog detector using ML?

Option 3: We can run in a separate process somewhere

- This is the most common approach
- Provide a RESTful interface to model evaluation (and any feature extraction)
- Cons:
  - Extra complexity for deploying additional service
- Pros:
  - It decouples your technology choices between the internet and the ML
  - It can run wherever you want it to
  - It can scale independently

How is hotdog detector using ML?

Pro: "It decouples your technology choices between the internet and the ML"

- Platform choices are hard, and their consequences last many years
- In 2019 what is the best internet platform to start your project on?
- In 2019 what is the best ML language and framework for your project?
- What if these aren't the same?

How is hotdog detector using ML?

Pro: "It can run whereve you want it to"

- The best hardware to run your web app may not be the best hardware to run your models
- Models vary and your model may…
  - Require a lot of memory
  - Require a lot of CPU
  - Require a GPU (or TPU)
- You may want to outsource your model evaluation completely
  - Google Cloud AI Platform, AWS Sagemaker are both great tools for evaluating and monitoring your models
- Models are semi-portable
  - Pickle
  - PMML (predictive model markup language)

How is hotdog detector using ML?

Pro: "It can scale independently"

- This is true of any service, but why is this particularly true of a service that evaluates ML models?

How is hotdog detector using ML?

Pro: "It can scale independently"

- This is true of any service, but why is this particularly true of a service that evaluates ML models?
- Model evaluation is stateless!
  - In this class you have learned how easy it is to scale out stateless application servers and how hard it is to scale out stateful databases
  - Put a load balancer in front of as many servers as you want, and scale model inference easily

Where does training data come from?

You can't build a model without training data

- If you were going to build this, where would you get photos?
- And where would you get labels?

Where does training data come from?

You can't build a model without training data

- If you were going to build this, where would you get photos?
- And where would you get labels?

You can do a lot of things

- Take photos yourself
- Find images online by hand
- Label images yourself
- Buy data labeling services online

These are great, but some issues arise:

- None of these deliver massive data sets
- Are these photos and labels representative of what we will see in production?

Where does training data come from?

The ideal is when you can get your training data directly from your users

- There is no real limit to the amount of data you can gather
- The data is by definition representative of what your users will upload

But we have a "chicken and the egg" problem:

- We need a model to get users
- We need user data to get a model

How do we solve this?

Where does training data come from?

How do we solve this?

- Start with a poor model built with non-ideal data
- Get enough usage to improve the data set
- Improve your model with the new data
- A better model will mean a better product
- A better product will mean a more usage

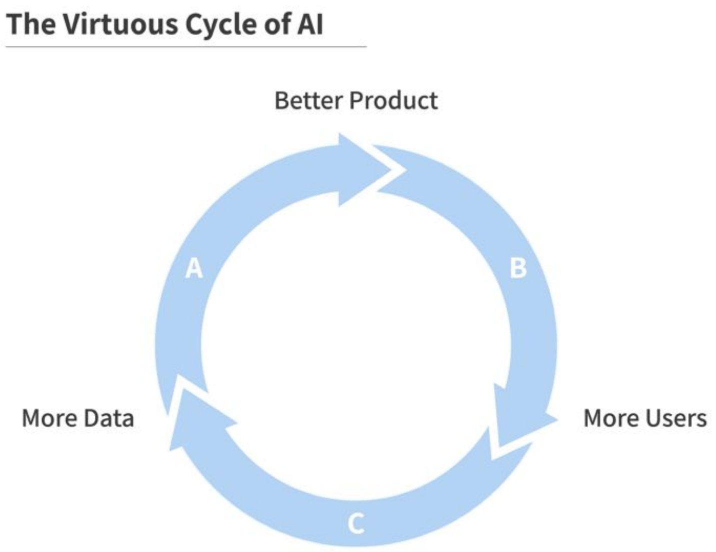


Image source: Landing AI's transformation playbook

Where does training data come from?

So how would we do this for hotdog app?

- Take lots of photos of hotdogs and not hotdogs
- Label them yourself
- Build a so-so model on this small data set
- Get some people to try using your product
- When you have enough user data, switch to building the model with user data

Depending on your product, "bootstrapping" your model may be really difficult.

Where does training data come from?

Technical options of getting training data

- Developer does a SQL query on the live DB
- ETL process to a data warehouse
- Unstructured data lake

How hard is it to train the model?

Some questions to ask in order to understand the training requirements of your system

- Can your **data preprocessing** be completed by a single machine within a reasonable amount of time?
- Can your **training** be completed by a single machine within a reasonable amount of time?
- Can your **training** fit within memory on a single machine at all?

How hard is it to train the model?

- Can your data preprocessing be completed by a single machine within a reasonable amount of time?
    - If not, you need to use a cluster computing framework
    - Hadoop: Open source map reduce implementation
    - Spark: JVM based cluster computing based around distributed data
    - Apache Beam (dataflow): Data processing pipelines
        - Unified batch and streaming architecture

- For hotdog: in the short term, we could preprocess on a single workstation, long term we would use something like Spark

How hard is it to train the model?

Can your training be completed by a single machine within a reasonable amount of time?

- If so, you can use your workstation. NNs can be significantly sped up by using GPUs (anywhere from 5 to 100x speedup)
- If not, you would use something like TF distributed
  - Data Parallelism: Many nodes train the same model on different subsets of the training data, periodically communicate about weights

Can your model fit within memory on a single machine/GPU at all?

- If not, you would use model parallelism
- Different GPUs train different parts of the network on the same examples

How hard is it to train the model?

For hotdog...

- In the short term we could use a single workstation
- In the long term we likely would eventually need a scalable way to train a reasonably sized NN on many examples, so would eventually use data parallelism
- Likely never need model parallelism

How often do we need to retrain?

How often do we retrain?

- The model is learning a function, and that function can degrade over time. Would you expect these to degrade?
    - Learning to recognize that a photo has a face?
    - Learning to identify a person from a face in a photo?

- Do you think hotdog would need to be retrained often? What might change that answer?

How often do we need to retrain?

Why is frequency of retraining important?

How often do we need to retrain?

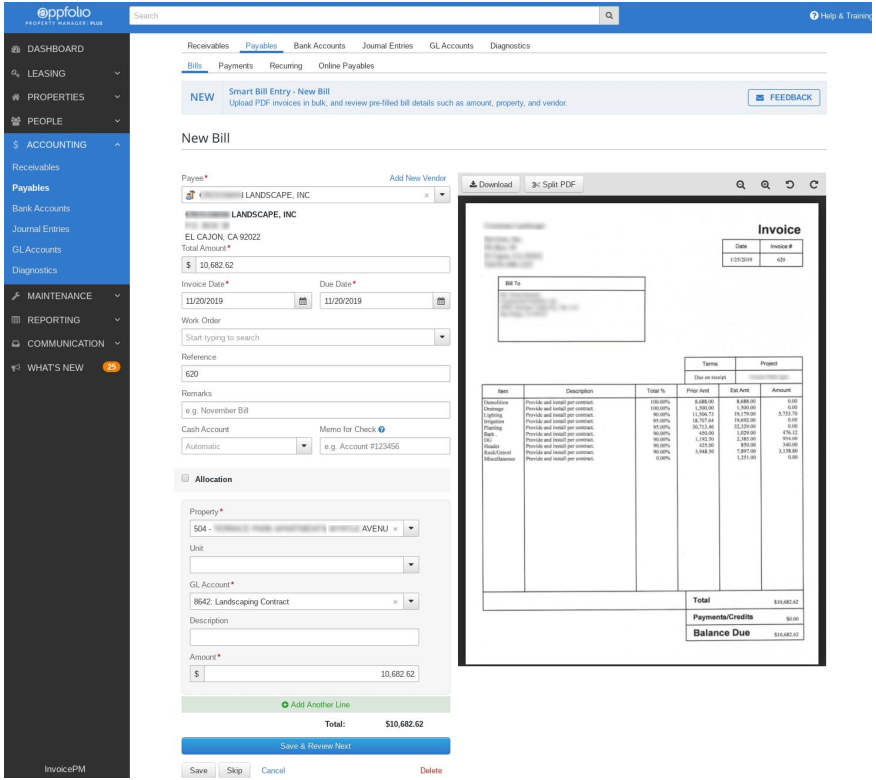
Why is frequency of retraining important?

- It affects how developers interact with training
- If you train a model infrequently, a model can be thought of as the output of development work
    - The developer is creating a model
- If you are frequently training models, they start to resemble recipes
    - The developer is creating code to create and manage models
    - Natural fit for a workflow or CI pipeline

Where does ML fit in?

- Lets see some examples of how ML fits in to real products:
  - Smart Bill Entry
  - Lisa AI

Where does ML fit in?



- User uploads invoices via PDF
- Machine interprets the invoice
- User is presented with the invoice, and the (hopefully) correct accounting data to be entered

Where does ML fit in to SBE?

How is ML being used in the product?

- Uploaded invoice fed to multiple models in order to extract information

Where does training data come from?

- Users reviewing the invoices

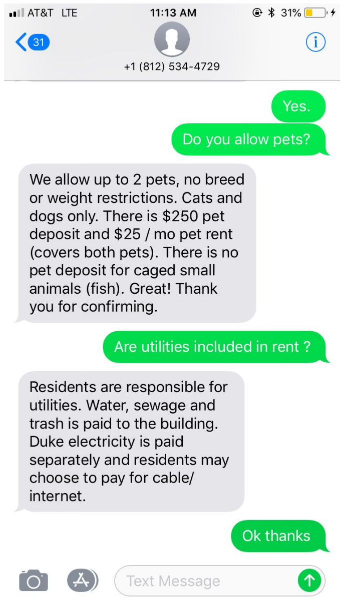
How hard is it to train the model?

- Any particular model is small, but we tune models per-customer and have >10K customers

How often do we retrain?

- Models degrade quickly, so we retrain weekly

Where does ML fit in?



- Prospect has a text-based conversation with an AI system
- Prospects ask questions about apartments
- Conversationally schedule a time to come see the apartment in person
- Light pre-qualification
- Follow up after the showing

Where does ML fit in to Lisa?

How is ML being used in the product?

- All human communication runs through multiple models to understand it

Where does training data come from?

- Behind the scenes operators

How hard is it to train the model?

- Few models, take 2-4 hours each to train

How often do we retrain?

- Models degrade slowly, so every few months

Conclusion

ML is having a big impact: not just on our user experiences, but on the software systems that power them

Thank you for your time!

Any questions?