# Introduction to Machine Learning Systems

Pooyan Jamshidi
USC

# Target Audience

- First, it serves students who are interested in machine learning but haven't built many real-world machine learning systems.

- The course is different from other ML courses you may have picked up. In this course, we will study techniques applicable to building whole production-ready systems, not just naive scripts.

- We'll explore the entire range of possible components you might need to implement in a machine learning system, with lots of hard-won tips about common design pitfalls.

- Along the way, you'll learn about the various jobs of a machine learning system, in the context of implementing systems that fulfill those needs. So, if you don't have a lot of background in machine learning, don't worry that you'll have to wade through pages of math before you get to build things.

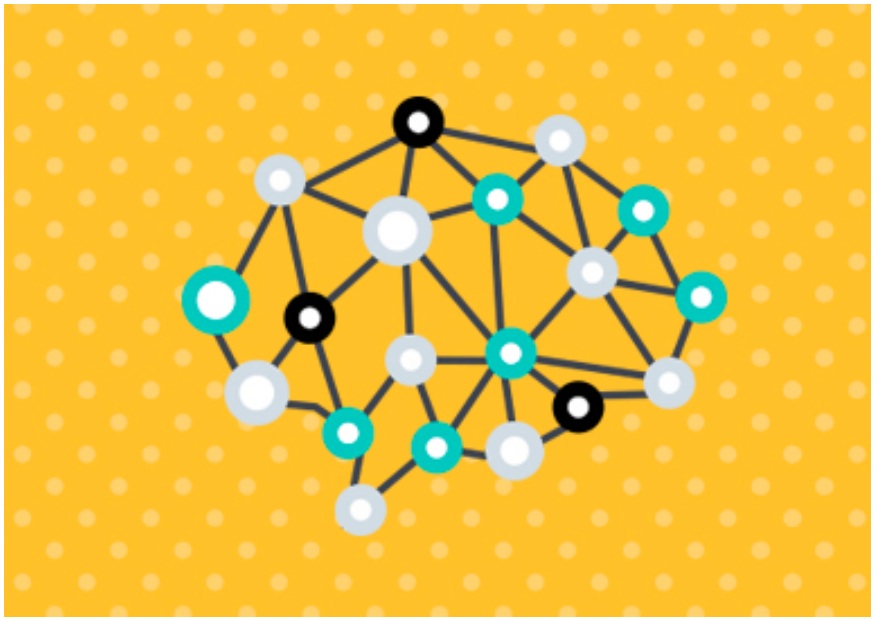
# Target Audience

- Second, this course serves students who are interested in the bigger picture of machine learning systems.
- I presume they know the concepts of machine learning but may only have implemented simple machine learning functionality (for example, scripts over files on a laptop).
- For such students, the course may introduce you to a range of concerns that you've never before considered part of the work of machine learning.
- I'll introduce components of a system that are often neglected in academic machine learning discussions, and then I'll show you how to implement them.

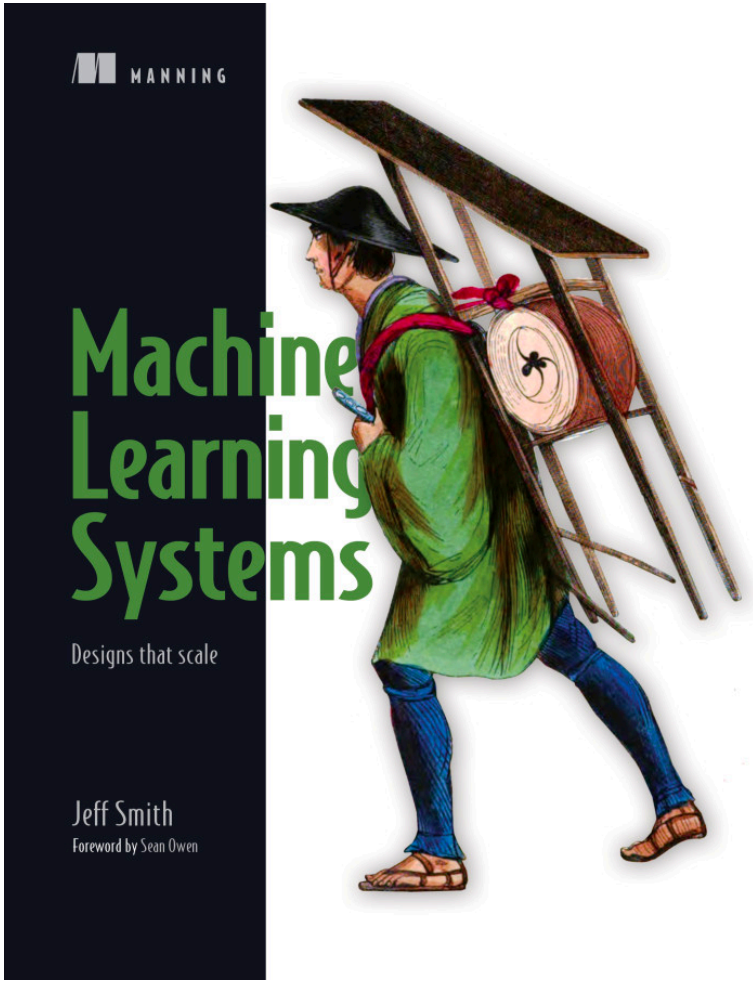
# Learning goals

- Understand how to build a system that can put the power of machine learning to use.
- Understand how to incorporate ML-based components into a larger system.
- Understand the principles that govern these systems, both as software and as predictive systems.

In this course, we study real-world challenges of adopting ML and solutions that scale



# Main Sources




UBER Engineering

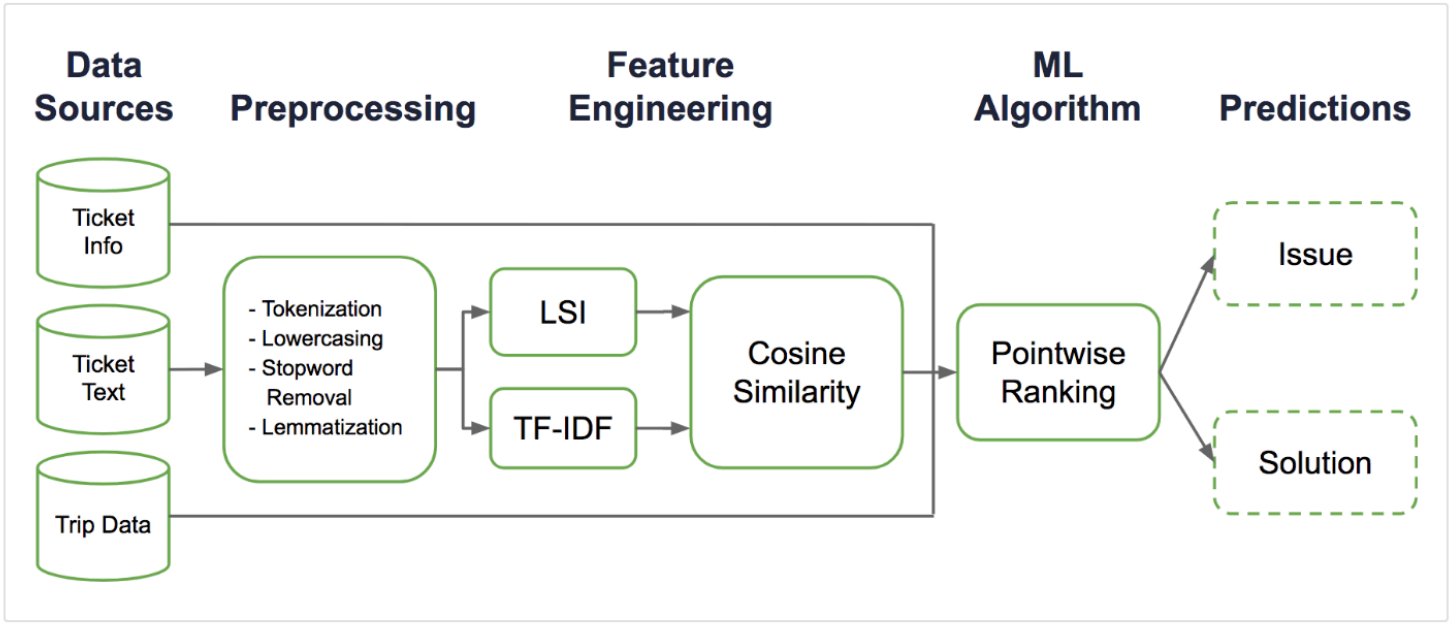
THE NETFLIX TECH BLOG

Spotify Labs

# COTA: Improving Uber Customer Care with NLP & Machine Learning

By Huaixiu Zheng, Yi-Chia Wang, & Piero Molino

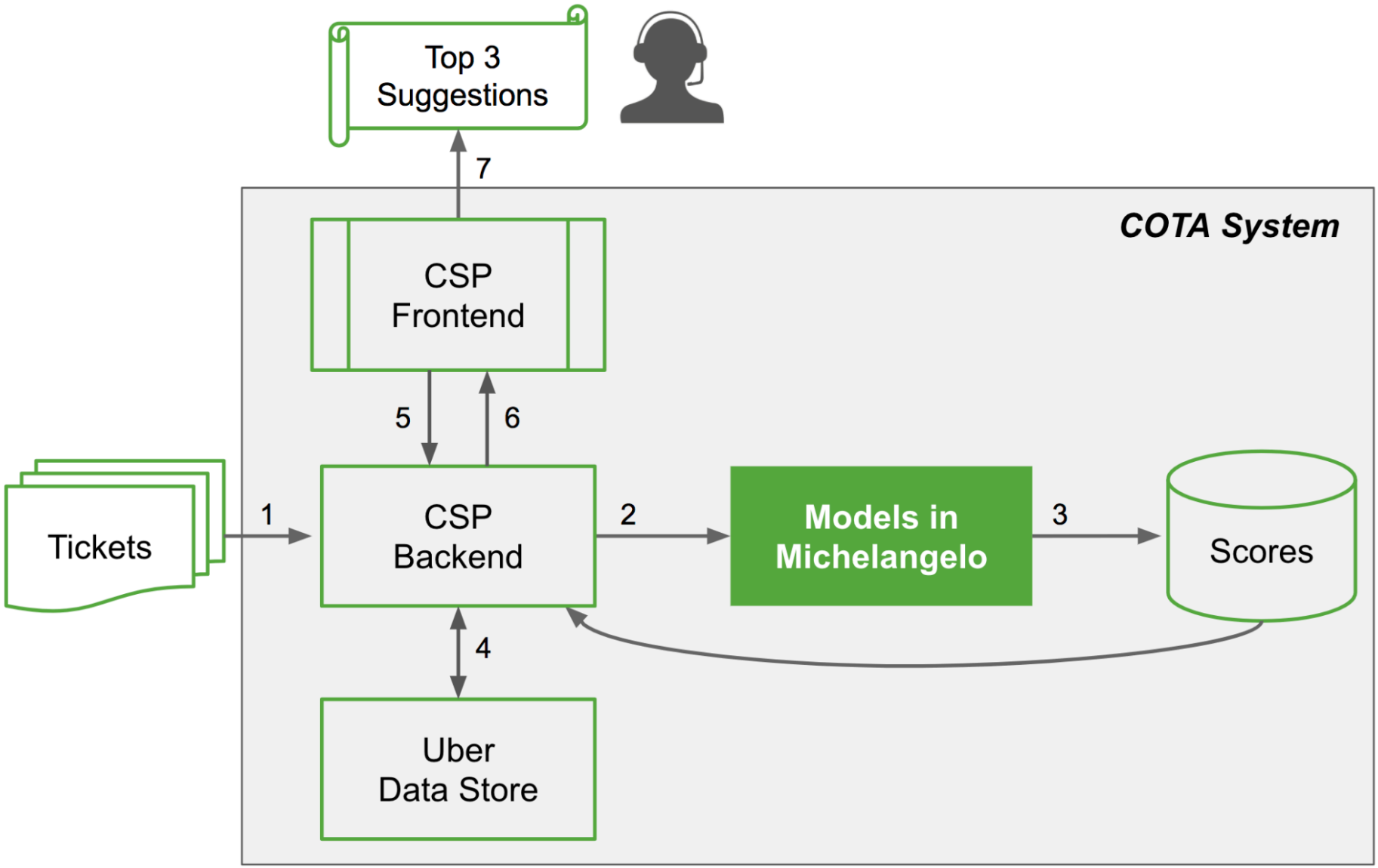
January 3, 2018



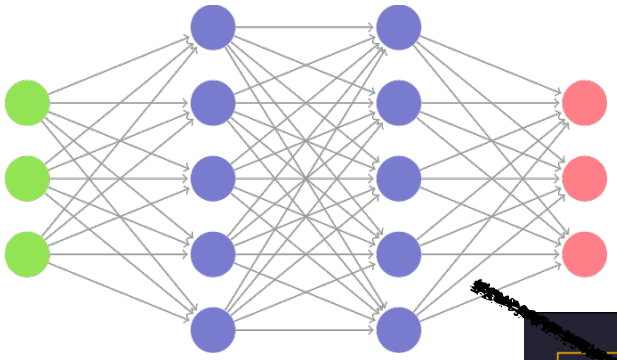
# How each lecture looks like?

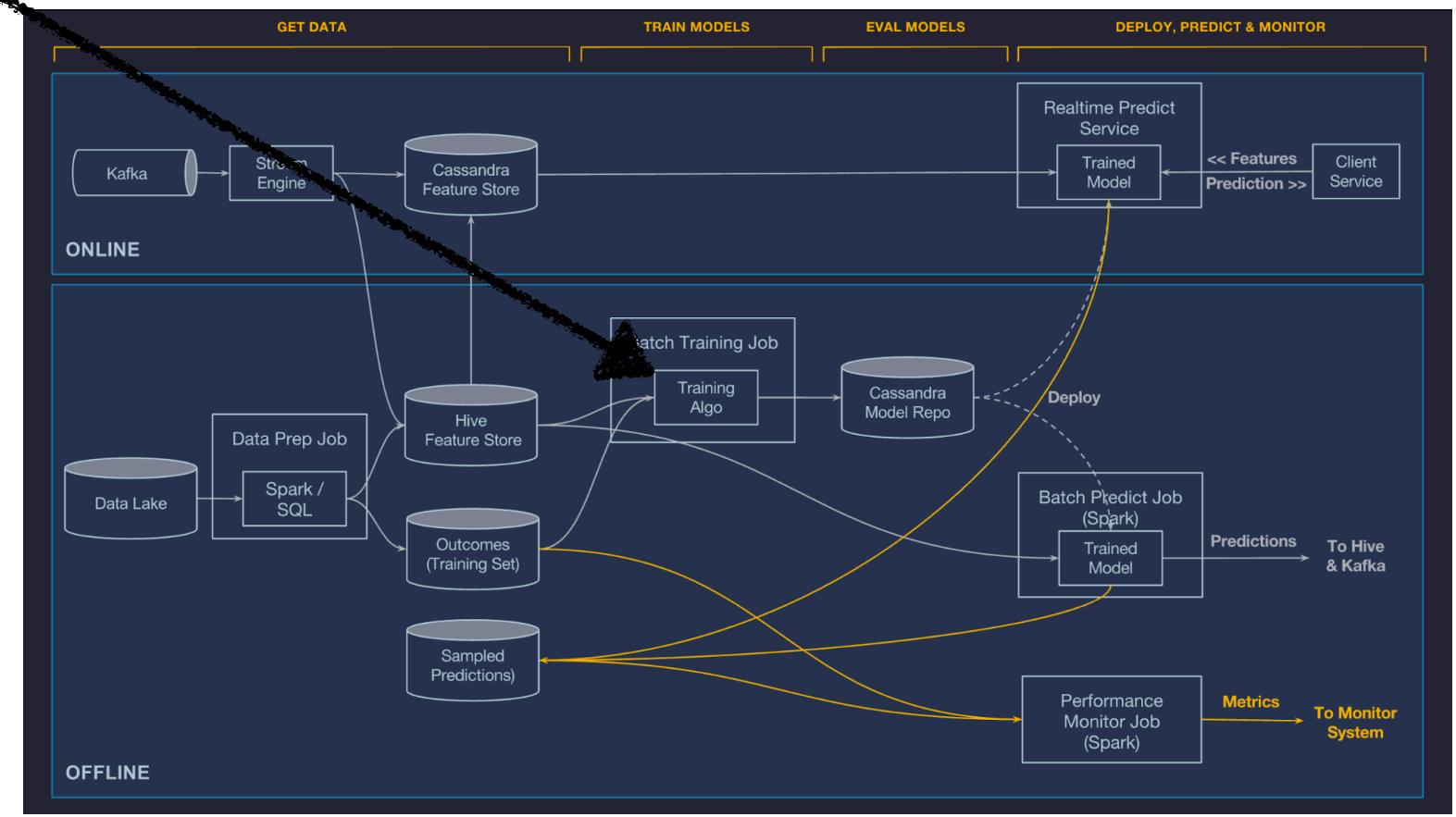
- We study ML systems in real world
- We discuss challenges for ML systems in real world
- We review solutions thats scale

# What is an ML system?



# What is an ML system?

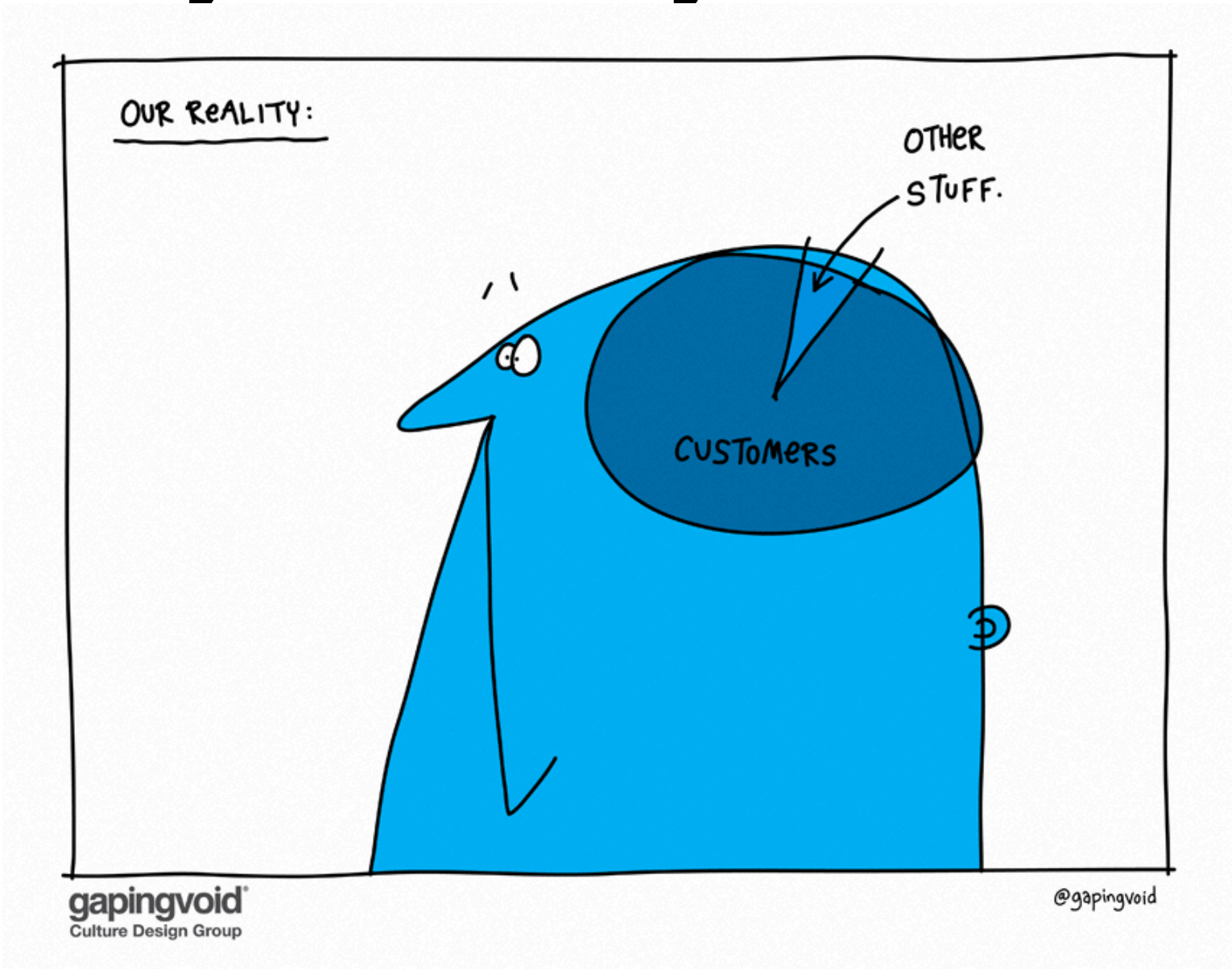




# Customer Obsession Ticket Assistant (COTA) is an ML system

- Assists Uber agents to resolve issues
- Hundreds of thousands issues daily
- 400+ cities worldwide

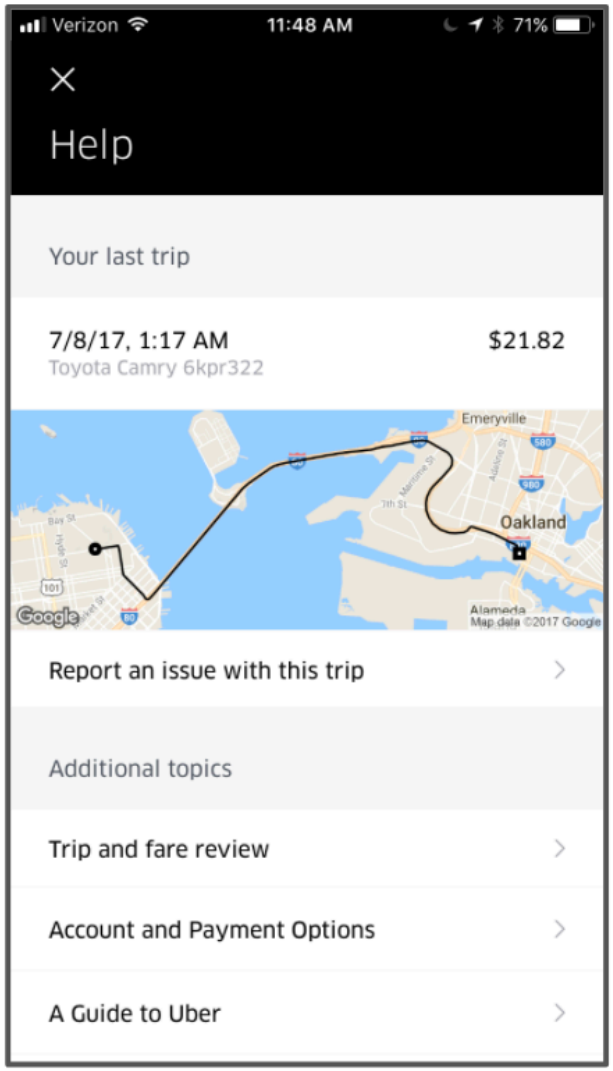
# Have you though how Uber quickly resolve your issues?

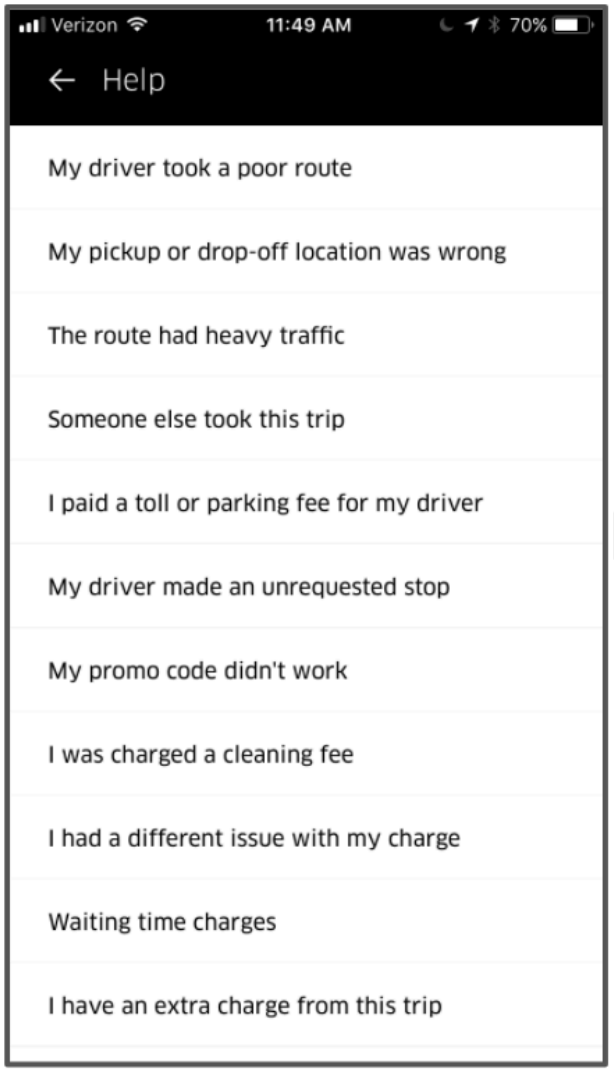


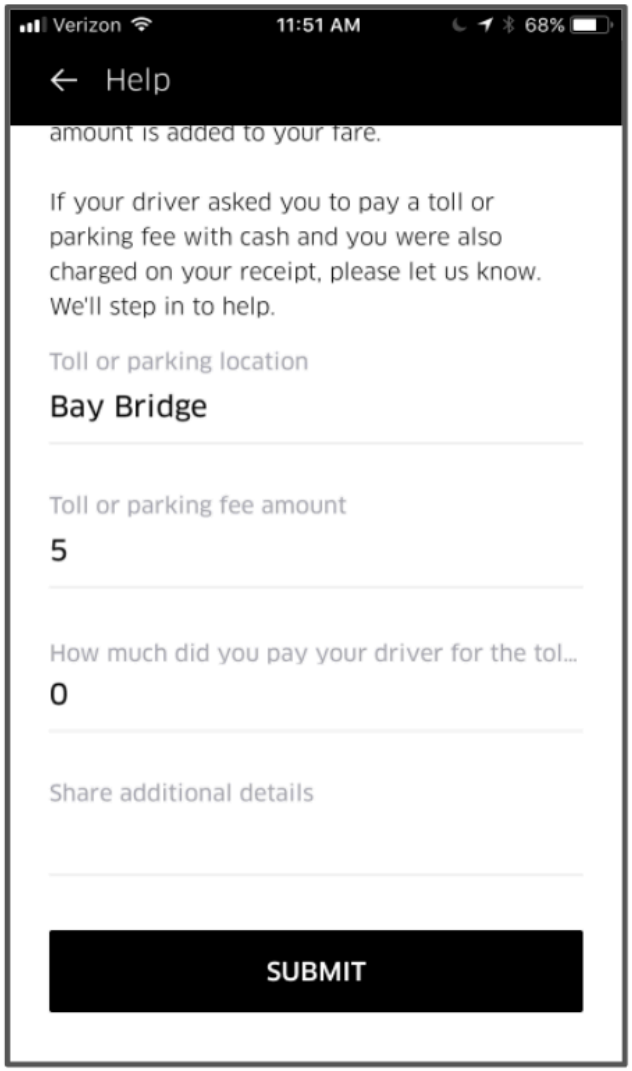
# Outline

- Motivations behind creating COTA
- Outline its backend architecture
- Showcase how the powerful tool has led to increased customer satisfaction

# Looks familiar?



# So what is the challenge?

- Although this provides important context,
- Not all of the information needed for solving an issue is obtainable through this process, particularly given the wide variety of possible solutions available.
- Moreover, the diversity of ways a customer can describe an issue associated with a ticket further complicates the ticket resolution process.

# The main challenge is showing up at scale

- As Uber continues to grow at scale, support agents must be able to handle an ever-increasing **volume** and **diversity** of support tickets, from technical errors to fare adjustments.

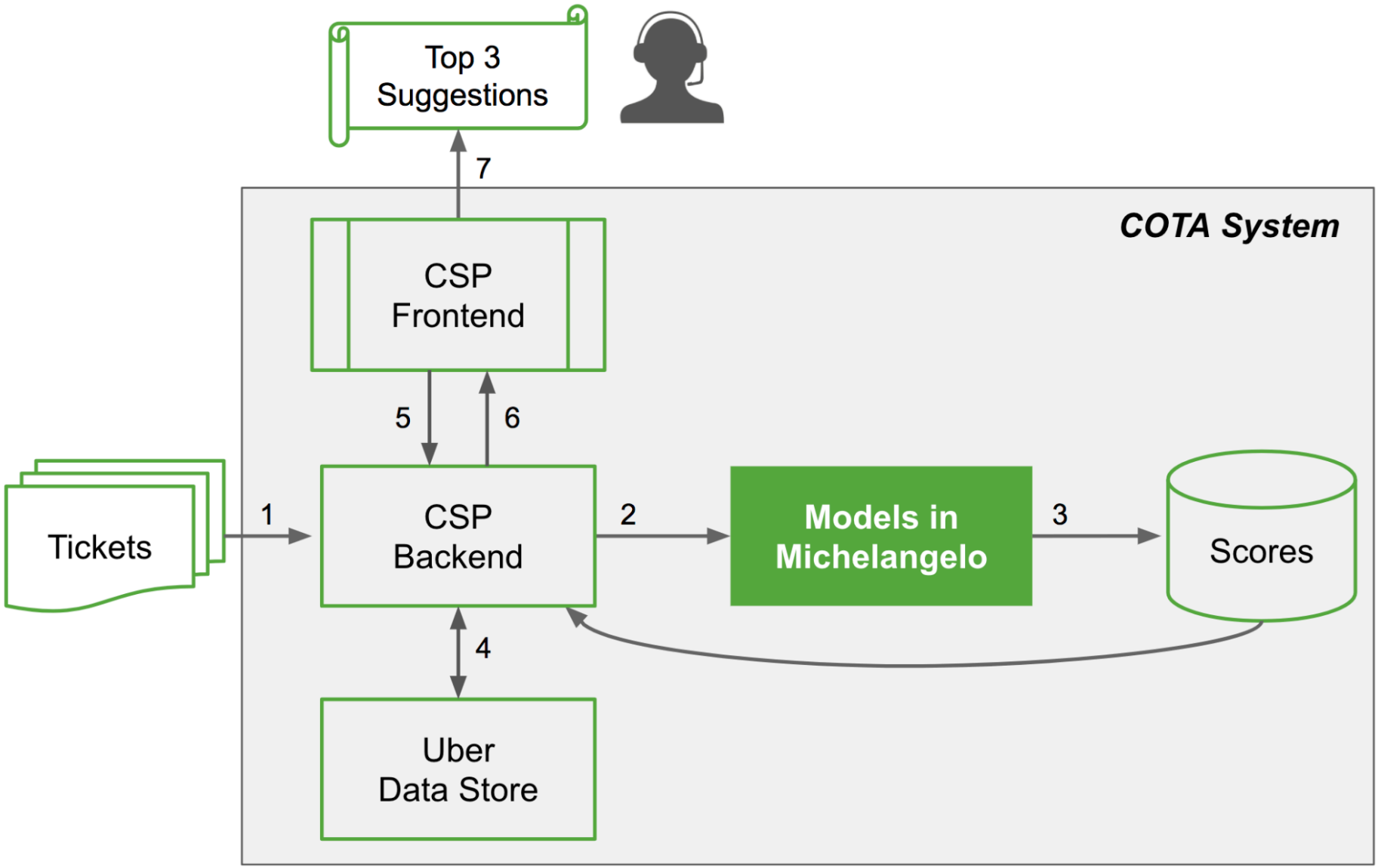
- In fact, when an agent opens a ticket, the first thing they need to do is determine the issue type out of thousands of possibilities—no easy task!

- Reducing the amount of time agents spend identifying tickets is important because it also decreases the time it takes to resolve issues for users.

# Choosing resolution at scale

- Once an issue type is chosen,
- The next step is to identify the right resolution,
  - with each ticket type possessing a different set of protocols and solutions.
  - With thousands of possible resolutions to choose from, identifying the proper fix to each issue is also a time-intensive process.

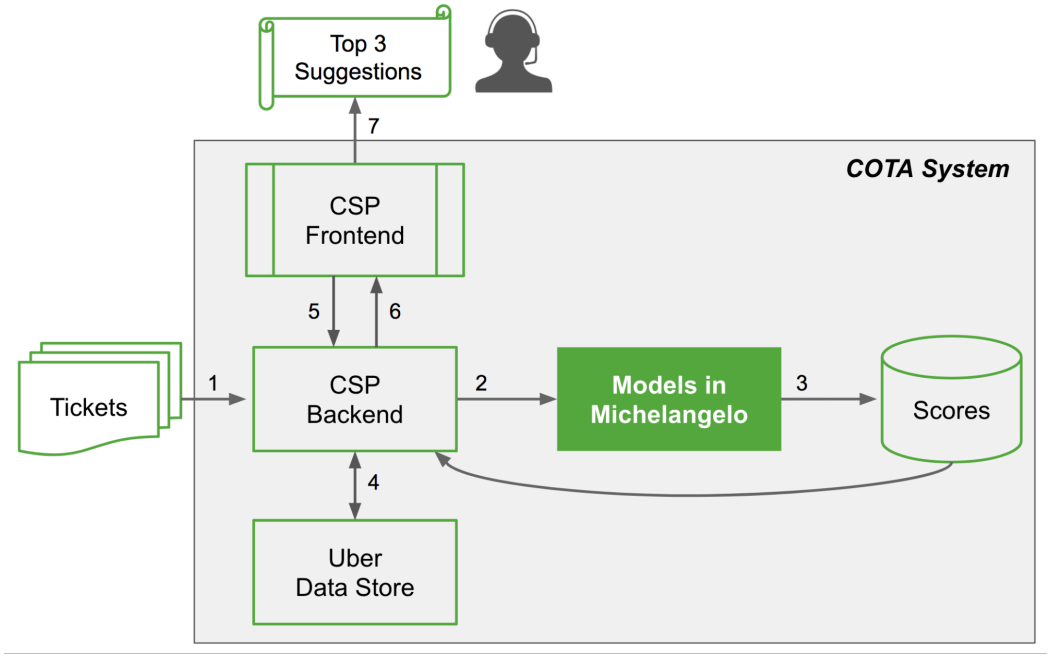
# COTA: Customer Obsession Ticket Assistant



# COTA composes/built on top of two subsystems

- Built on top of customer support platform,
- Michelangelo-powered models suggest the three most likely issue types and solutions based on ticket content and trip context.

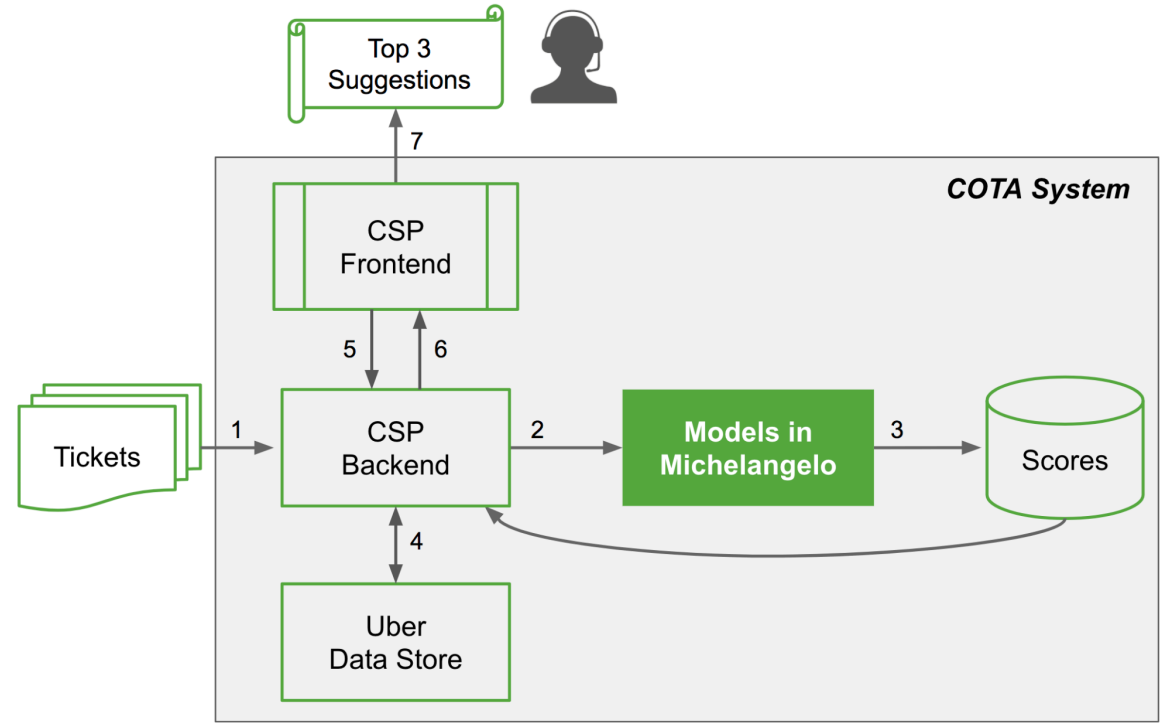
# Is COTA an ML system?



# COTA Architecture

1. Once a new ticket enters the customer support platform (CSP), the back-end service **collects all relevant features** of the ticket.

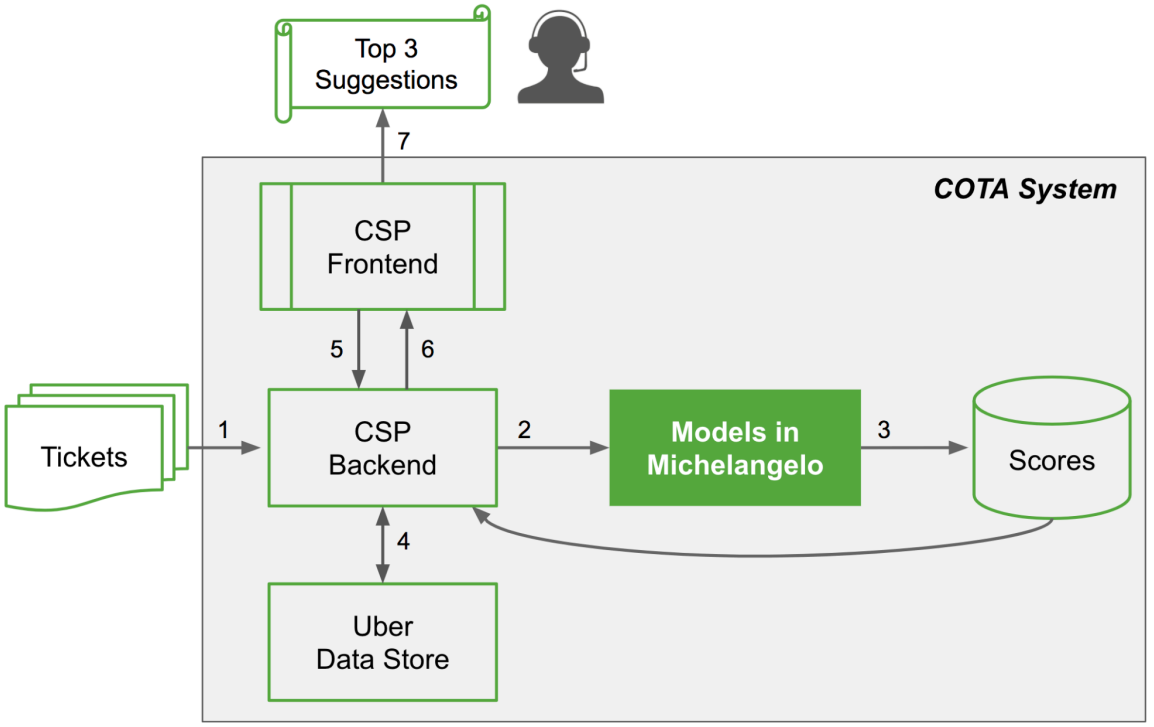
2. The back-end service then sends these features to the machine learning model in Michelangelo.



# COTA Architecture

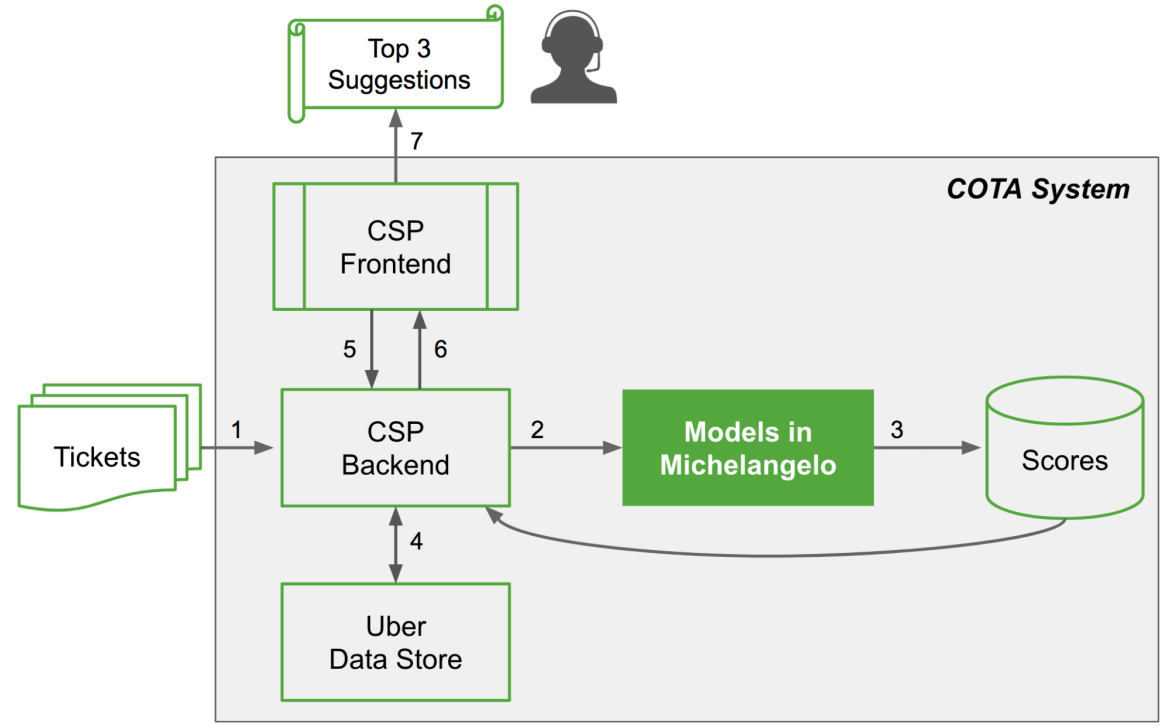
3. The model **predicts scores** for each possible solution.

4. The back-end service receives the predictions and scores, and **saves them to our Schema-less data store**.



# COTA Architecture

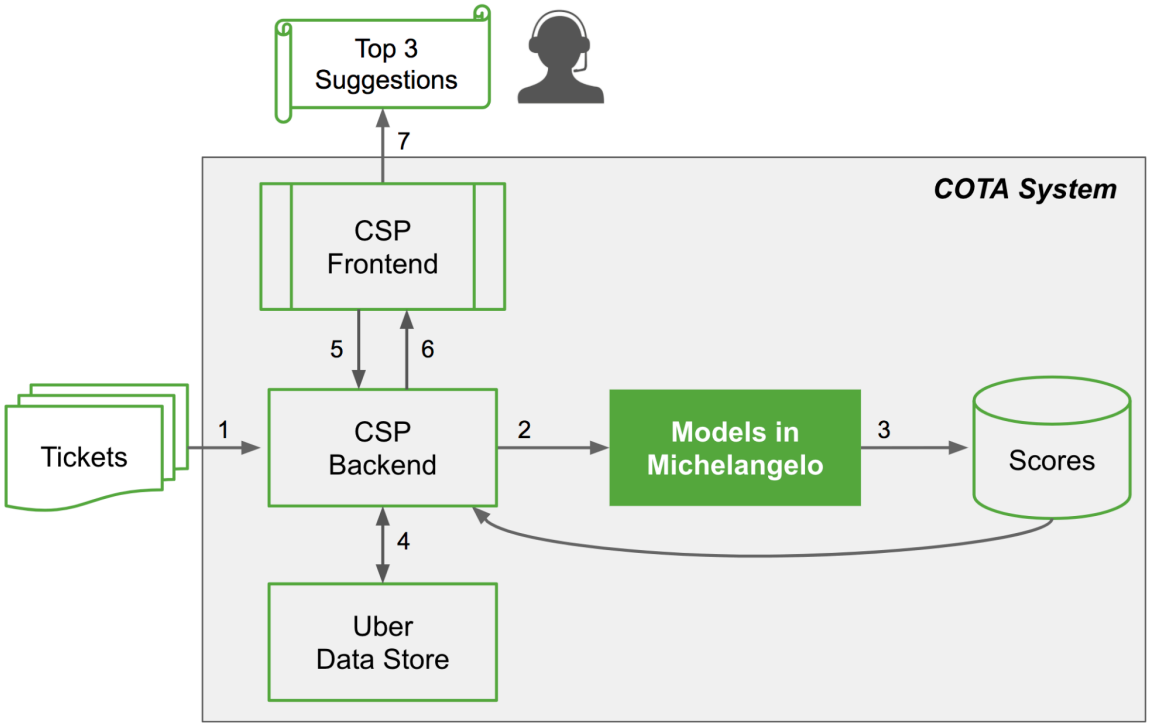
5. Once an agent opens a given ticket, the front-end service triggers the back-end service to check if there are any updates to the ticket. If there are no updates, the back-end service will retrieve the saved predictions; if there are updates, it will fetch the updated features and go through steps 2-4 again.



# COTA Architecture

6. The back-end service **returns the list of solutions** ranked by the predicted score to the frontend.

7. The top three ranked solutions are suggested to agents; from there, agents make a selection and resolve the support ticket.



# So what?

- Results are promising;
- COTA can reduce ticket resolution time by over 10 percent
- While delivering service with similar or higher levels of customer satisfaction

# Let’s have a look at the backend

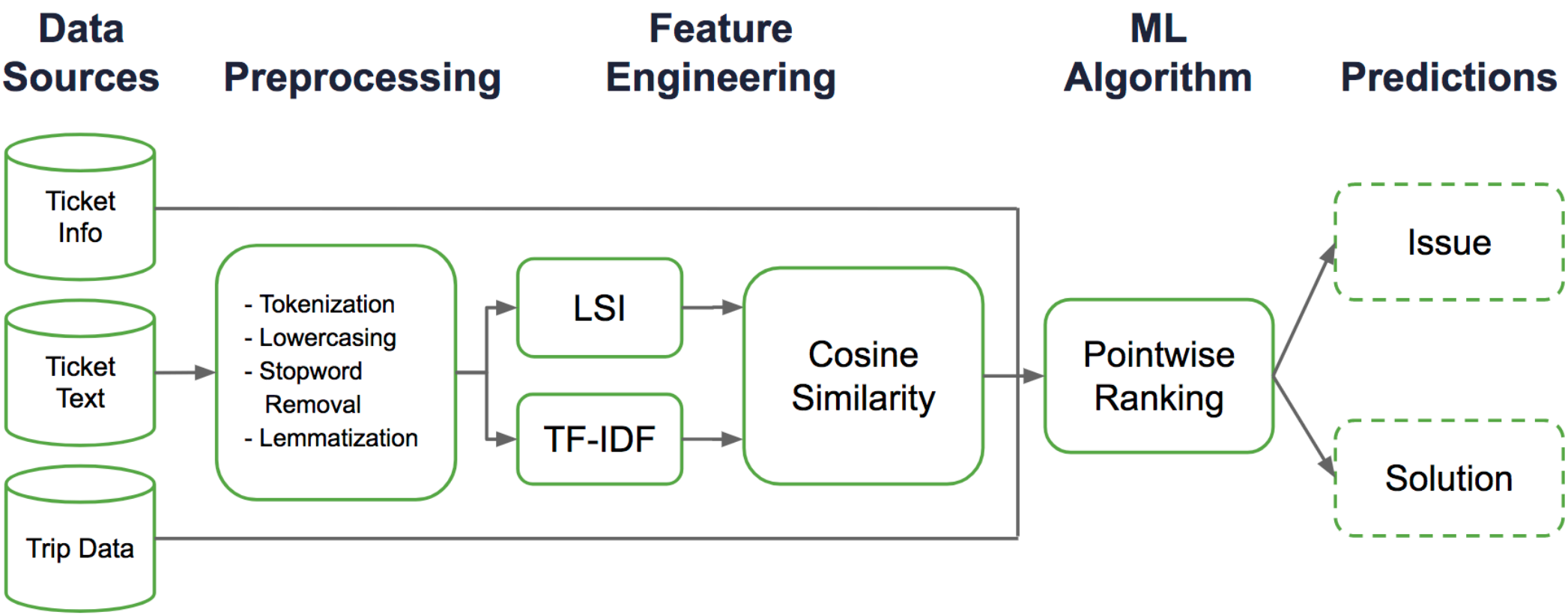
- To accomplish the goal, machine learning model leverages features extracted from
  - customer support messages,
  - trip information,
  - and customer selections in the ticket issue submission hierarchy outlined earlier.

Any guess what is the most important feature of the ticket you submit to Uber?

# Any guess what is the most important feature of the ticket you submit to Uber?

- The most valuable feature for identifying issue type is the message customers send to agents about their issue.
- Uber built a NLP pipeline to transform text across several different languages into useful features.

# The NLP pipeline



# What NLP pipeline does?

- NLP models can be built to translate and interpret different elements of text, including phonology, morphology, grammar, syntax, and semantics.

- Depending on the building units, NLP can also register character-level, word-level, phrase-level, or sentence/document-level language modeling.

- Traditional NLP models are built by **leveraging human expertise** in linguistics to engineer handcrafted features. With the recent upsurge in end-to-end training for **deep learning models**, researchers have even begun to develop models that can decipher full chunks of text without having to explicitly parse out relationships between different words within a sentence, instead using raw text directly.

# What NLP pipeline does?

- Uber analyzes text at the word-level to better understand the semantics of text data.
- One popular approach to NLP is topic modeling, which aims to understand the meaning of sentences using the counting statistics of the words.
- Although topic modeling does not take into account word ordering, it has been proven very powerful for tasks such as information retrieval and document classification.

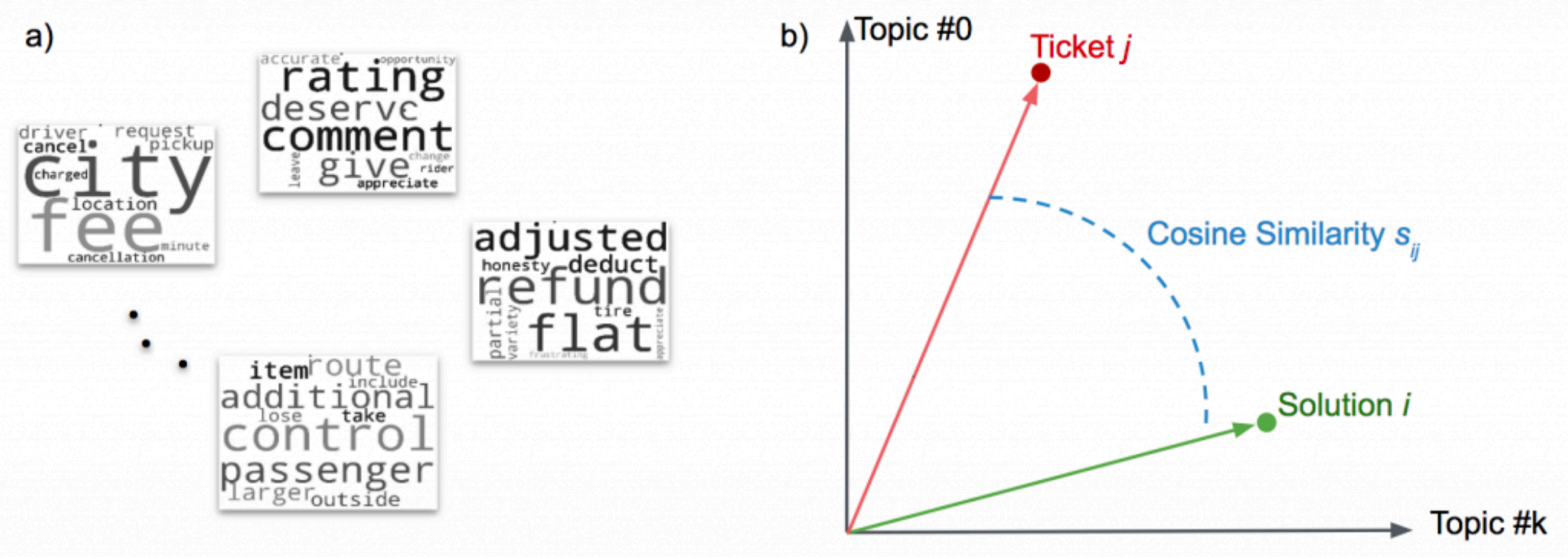
# Preprocessing

- **Cleaning** the text by removing HTML tags.
- **Tokenizing** the message's sentences and remove stopwords.
- Conducting **lemmatization** to convert words in different inflected forms into the same base form.
- Finally, **converting** the documents into a collection of words (a so-called bag of words) and build a dictionary of those words.

# Topic modeling

**TF-IDF and LSA to extract topics from rich text data in customer support tickets processed by our customer support platform.**

**All the solutions and tickets are mapped to the topic vector space, and cosine similarity between solution and ticket pairs are computed. (Next)**



# Feature engineering

- Topic modeling enables us to directly use the topic vectors as features to perform downstream classifications for issue type identification and solution selection.

Do you guess what could possibly go wrong?

# Training becomes challenging at scale

- However, this direct approach suffers from a **sparsity of topic vectors**;
- In order to form a meaningful representation of these topics, we typically need to keep hundreds or even **thousands of dimensions of topic vectors** with many dimensions having values close to zero.
- With a very **high-dimensional feature space and large amount of data to process**, training these models becomes quite challenging.

# How Uber solved the challenge?

- Performing further feature engineering by computing **cosine similarity** features.
- Using solution selection as an example, we collect the **historical tickets** of each solution and form the bag-of-word representation of such a solution.

# Pointwise ranking

- Combined cosine similarity features together with other ticket and trip features that matches tickets to solutions

Do you guess what could possibly go wrong?

# Solution space becomes large

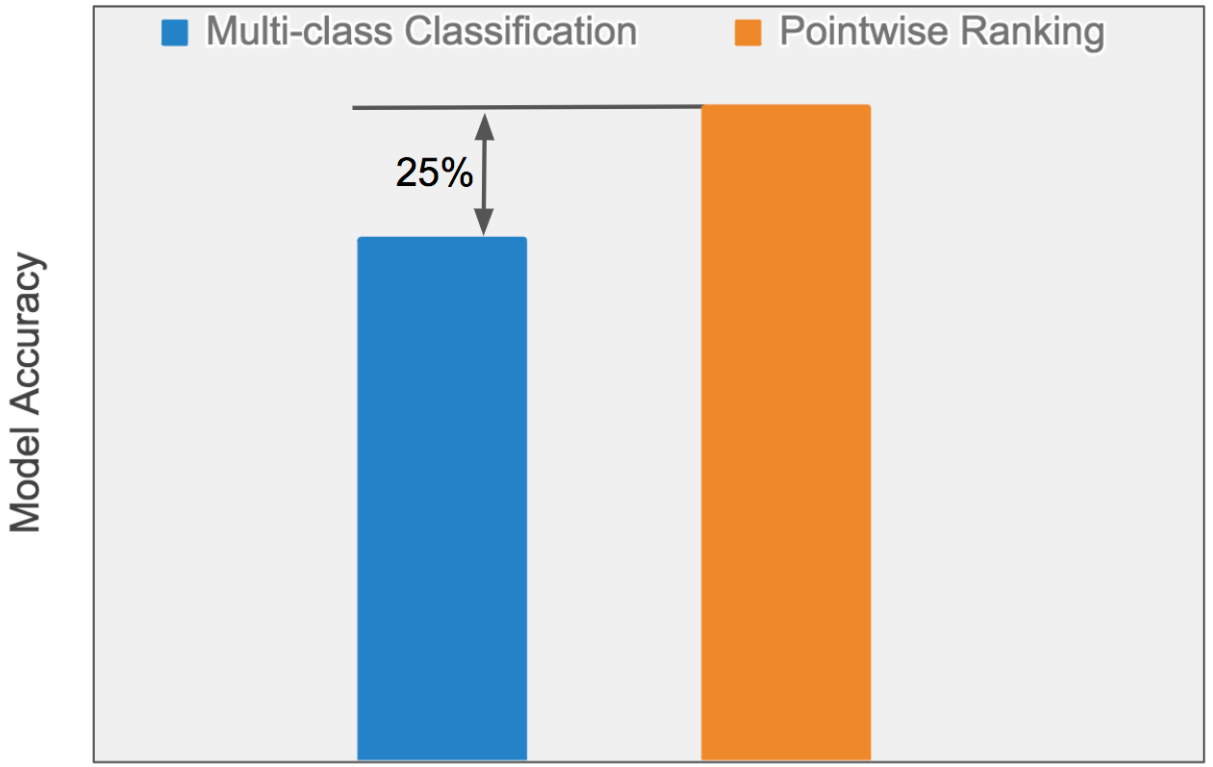
- With over 1,000 possible solutions
- For hundreds of ticket types,
- COTA’s large solution space becomes a challenge for the ranking algorithm of distinguishing the fine differences between these solutions.

# How Uber solved the challenge?

- Label the correct match between solution and ticket pair as positive
- Sample a random subset of solutions that do not match with the ticket and label the pairs negative
- Using the cosine similarity as well as ticket and trip features, we can build a binary classification algorithm that leverages the random forest technique to classify whether or not each solution-ticket combination matches.

# Results

- 25 percent relative improvement in accuracy
- speeds up the training process by 70 percent



# Easier and faster ticket solving = better customer support

But wait, how they actually measured they were successful for handling customer issues?

# Uber performed A/B tests

- To measure COTA's impact on our customer support experience, we conducted several controlled A/B test experiments online on English language tickets.
- Included thousands of agents and randomly assigned them into either control or treatment groups.
- Agents in the control group were exposed to the original workflow, while agents in the treatment group were shown a modified user interface containing suggestions on issue types and solutions.
- We collected tickets solved solely by either agents in the control or treatment group, and measured a few key metrics, including model accuracy, average handle time, and customer satisfaction score.

# Summary of results

- Measured the online model performance for both groups and compared them with offline performance. The model performance is consistent from offline to online.

- Measured customer satisfaction scores and compared them across control and treatment groups. Customer satisfaction often increased by a few percentage points. This finding indicates that COTA delivers the same or slightly higher quality of customer service.

- To determine how much COTA affected ticket resolution speed, we compared the average ticket handling time between the control and treatment groups. On average, this new feature reduced ticket handling time by about 10 percent.

# Deep learning experiments with various architectures

- Convolutional neural networks (CNNs),
- Recurrent neural networks (RNNs),
- And several different combinations of the two, including hierarchical and attention-based architectures.

# Training process

- Trained models in a multi-task learning fashion, with a single model capable of both identifying the issue type and suggest the best possible solution.

- Since issue types are organized into a hierarchy, we determined that we could train the model to predict the path in the hierarchy with a recurrent decoder using **beam search,** similar to the decoding component of a sequence to sequence model, and allowed for even more accurate predictions.

# Hyperparameter optimization to select the best model

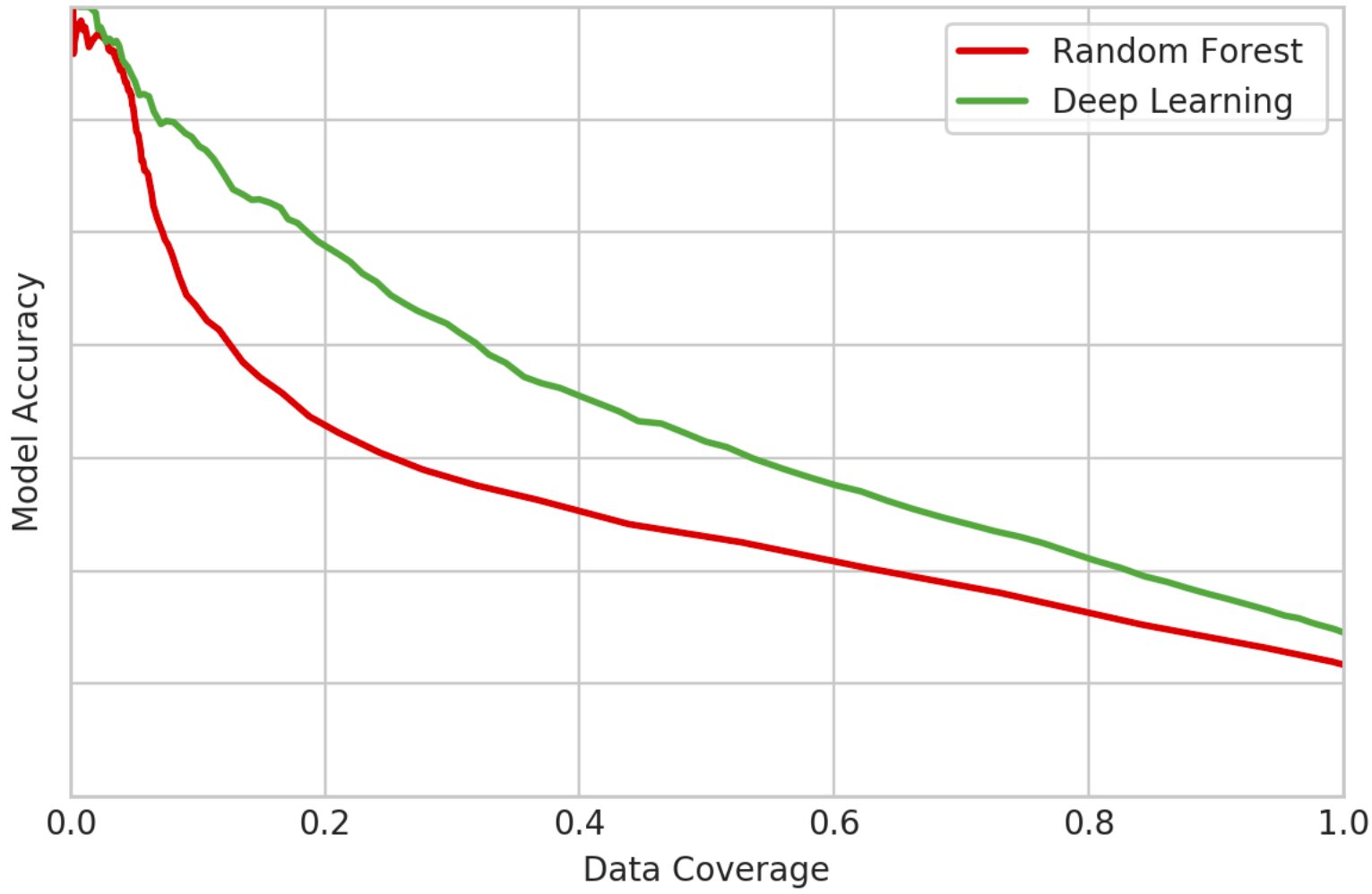
- Performed large scale hyperparameter optimization for all types of architectures, training them in parallel on our GPU cluster.

# Performance tradeoffs

- The final results suggest that the most accurate architecture is one that applies both CNNs and RNNs, Uber chose a simpler CNN architecture that was slightly less accurate but had more advanced computational properties in terms of training and inference time.

- In the end, the model Uber settled on provides about 10 percent greater accuracy with respect to the original random forest model.

# Tradeoff between accuracy and data coverage



# Summary

- We studied what an ML system looks like via a real-world system
- We reviewed the architecture of the ML system and challenges of building such system
- We reviewed some solution that helped Uber to scale and answer to customer issues more quickly