Introduction to Machine Learning Systems

Pooyan Jamshidi
UofSC

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

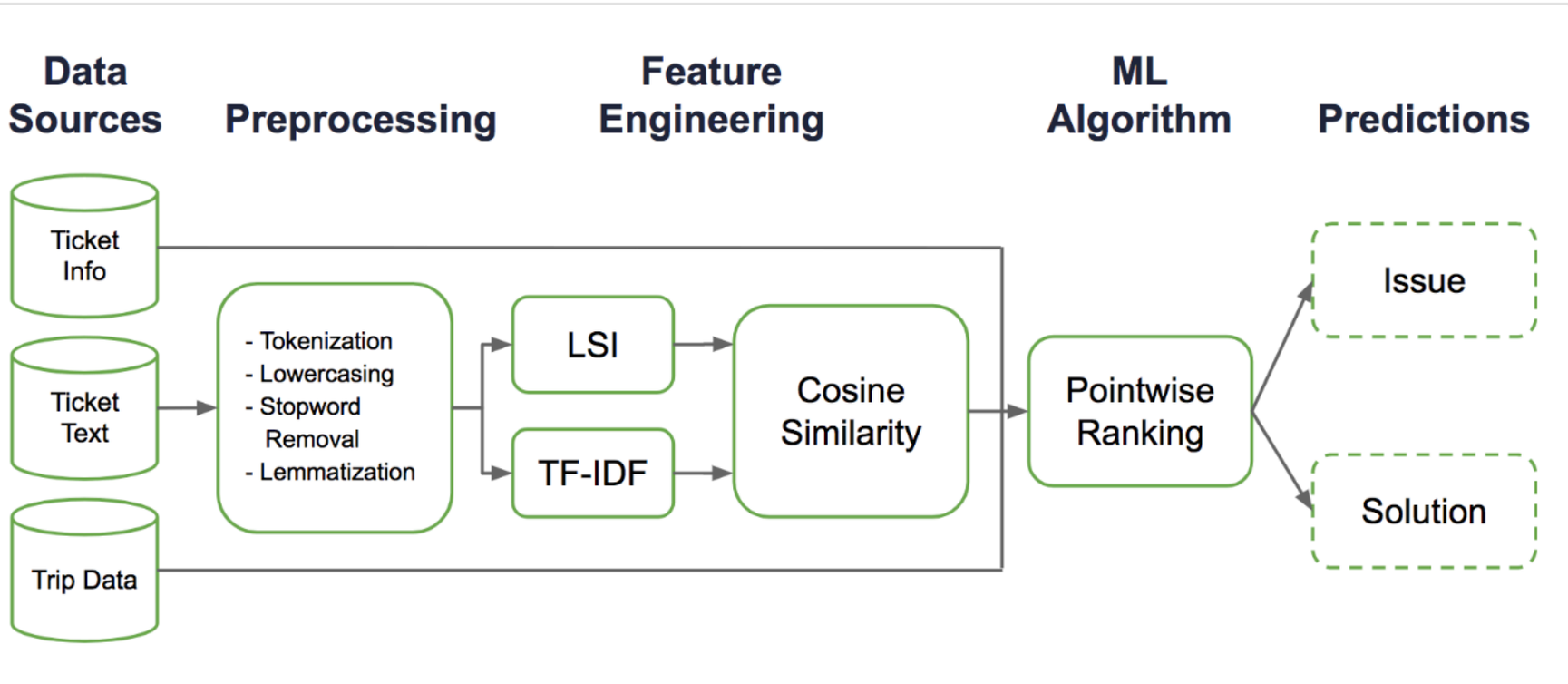
Outline

- Motivations behind creating an ML system
- Outline its backend architecture
- Showcase how such system has led to increased customer satisfaction in Uber

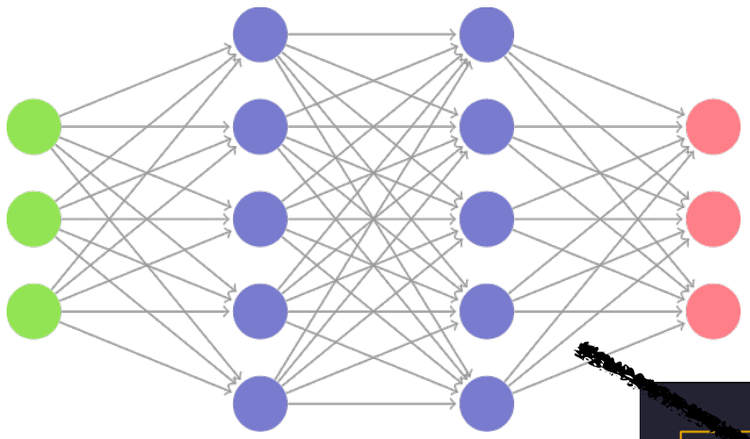
COTA: Improving Uber Customer Care with NLP & Machine Learning

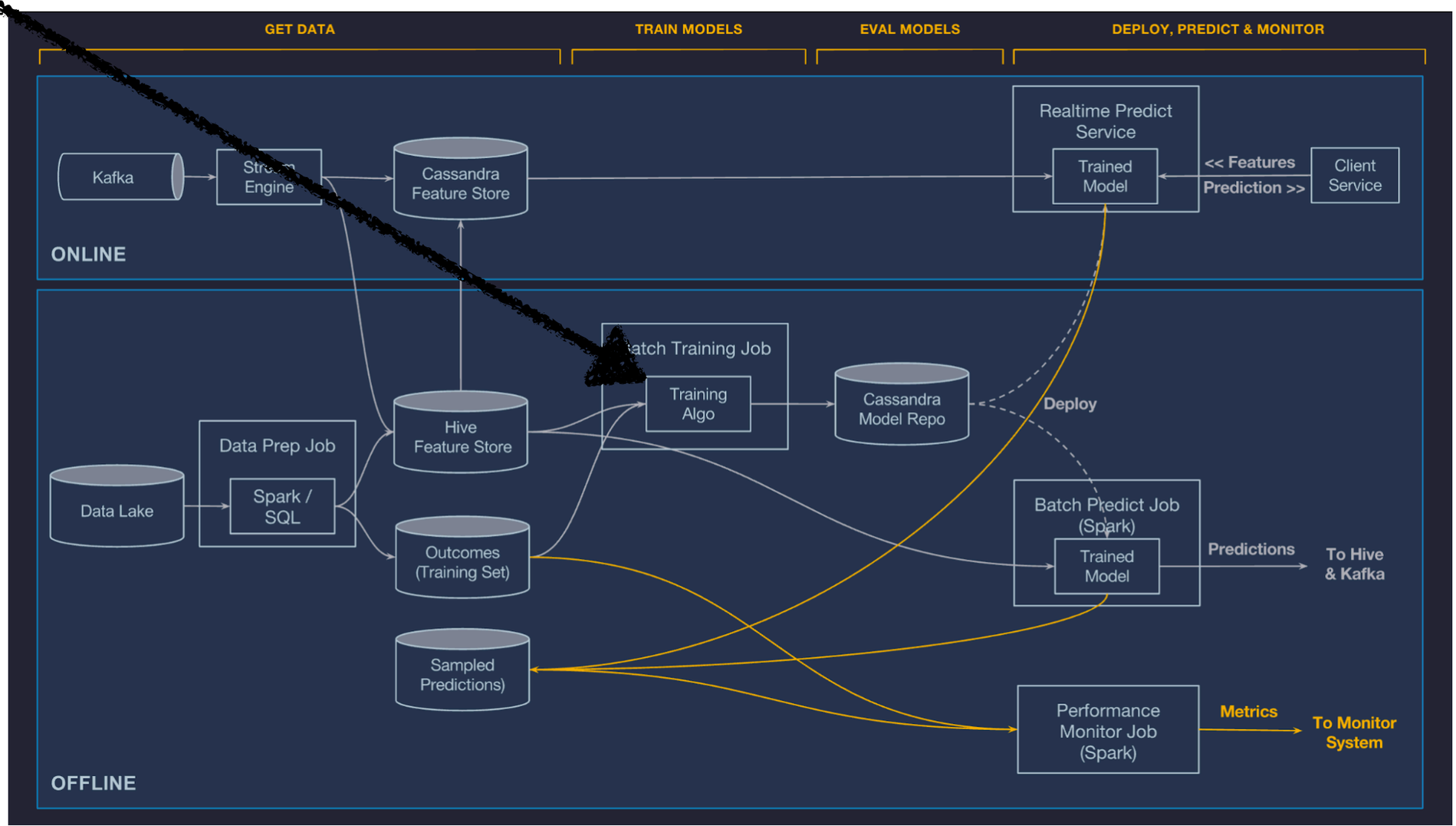
By Huaixiu Zheng, Yi-Chia Wang, & Piero Molino

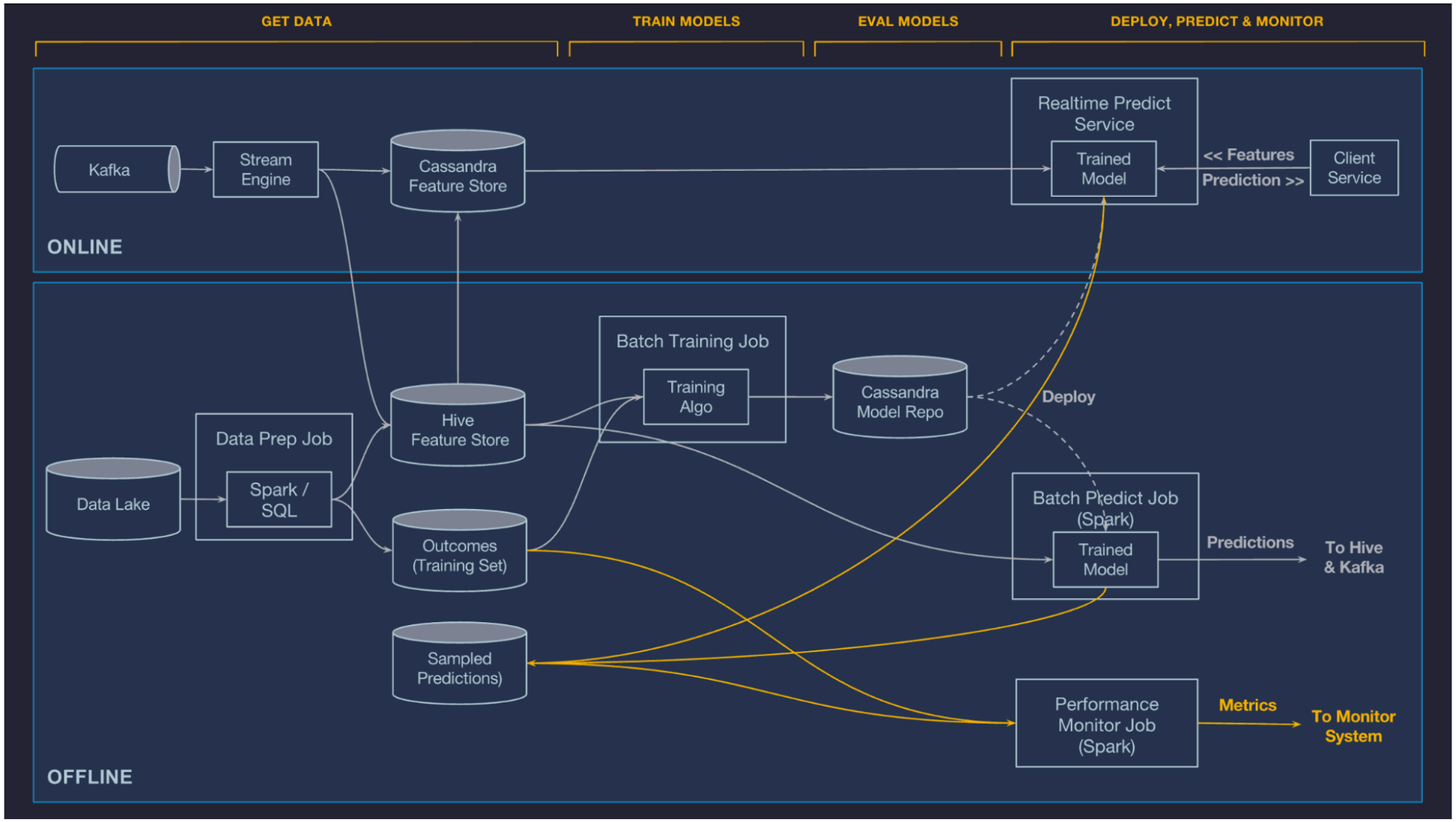
January 3, 2018



What is an ML system?





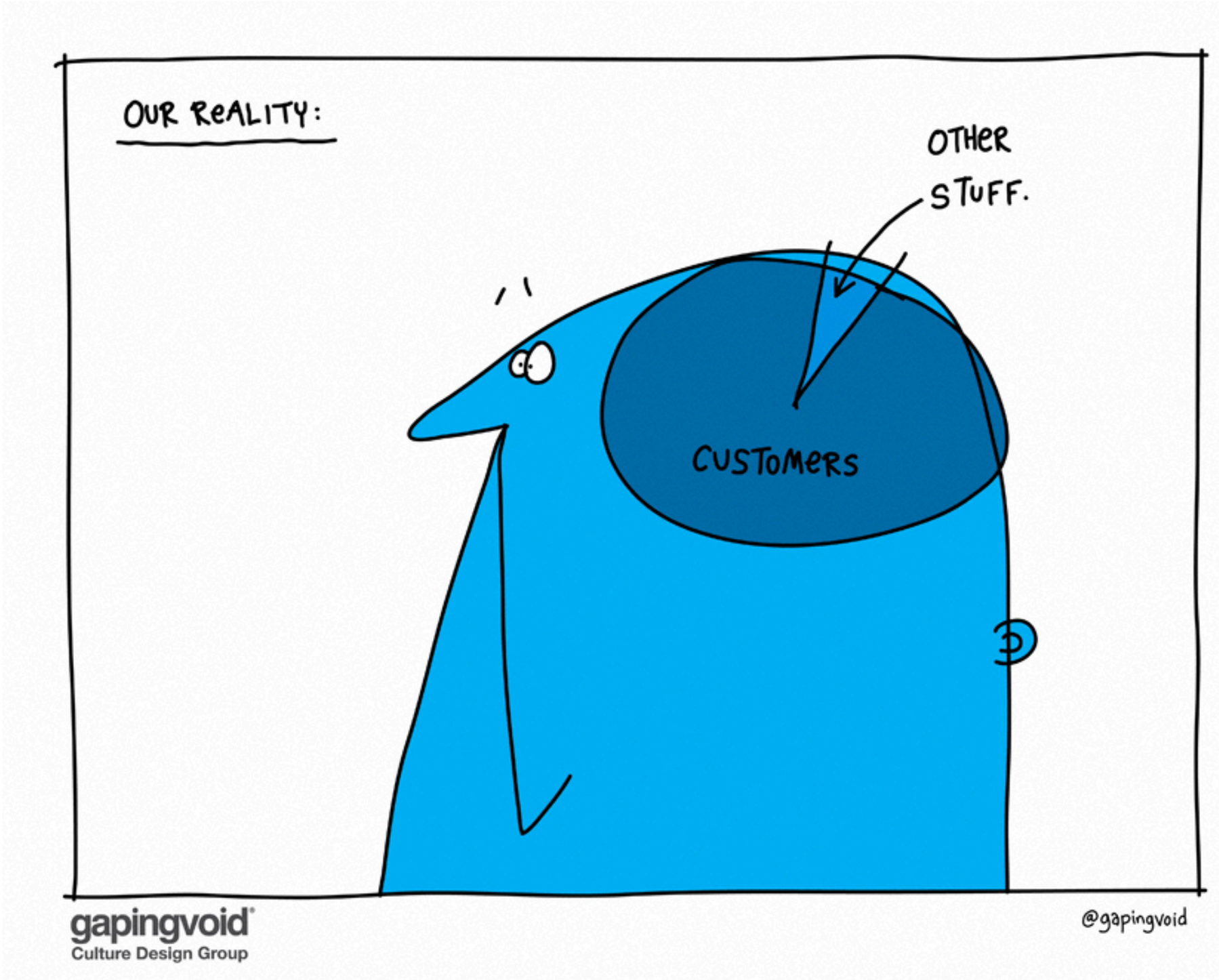


Machine learning systems comprise software/hardware components capable of learning from data and making predictions about the future.

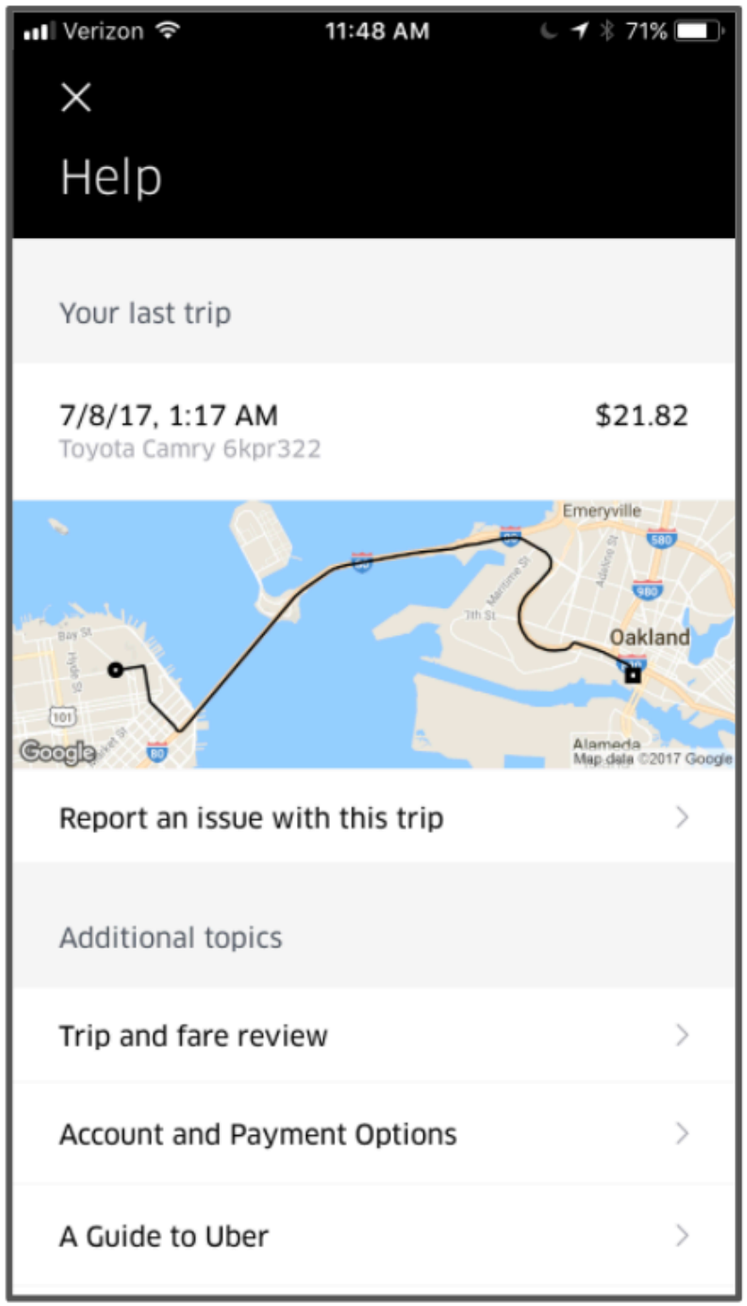
Customer Obsession Ticket Assistant (COTA) is an ML system

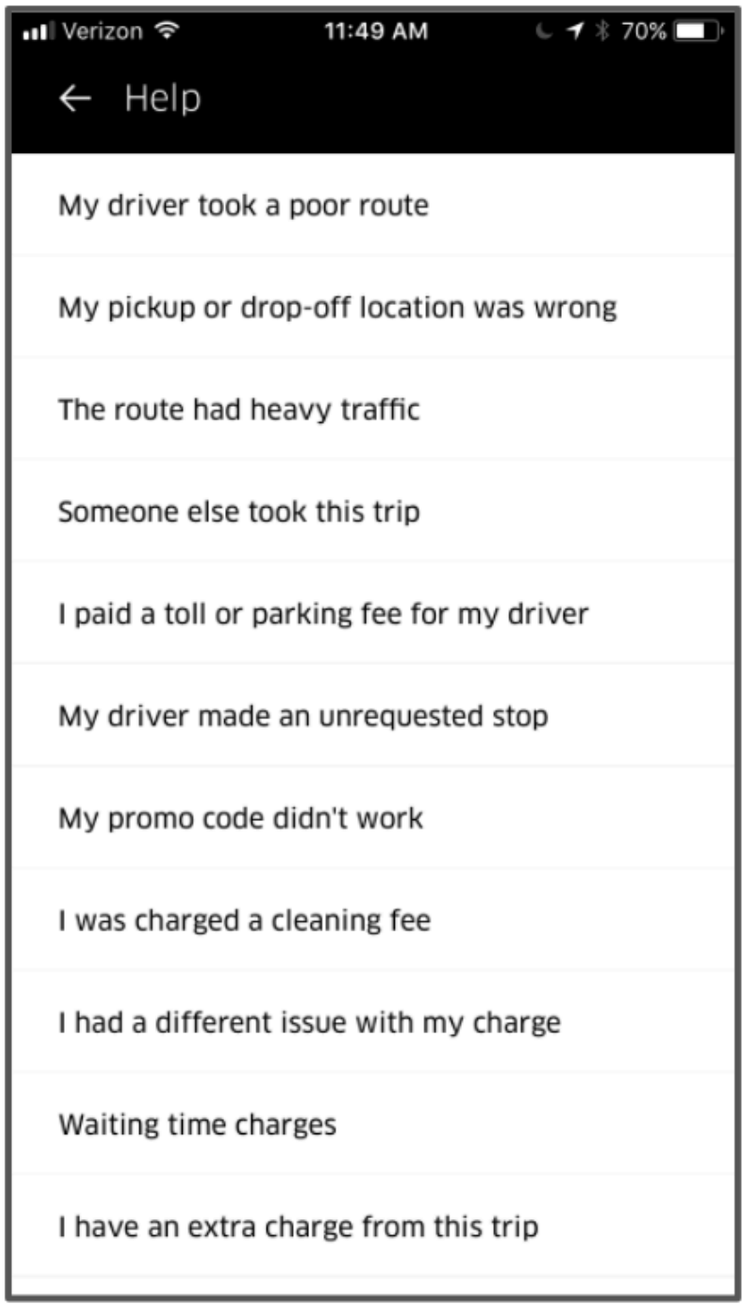
- Assists Uber agents to resolve issues
- Hundreds of thousands issues daily
- 400+ cities worldwide

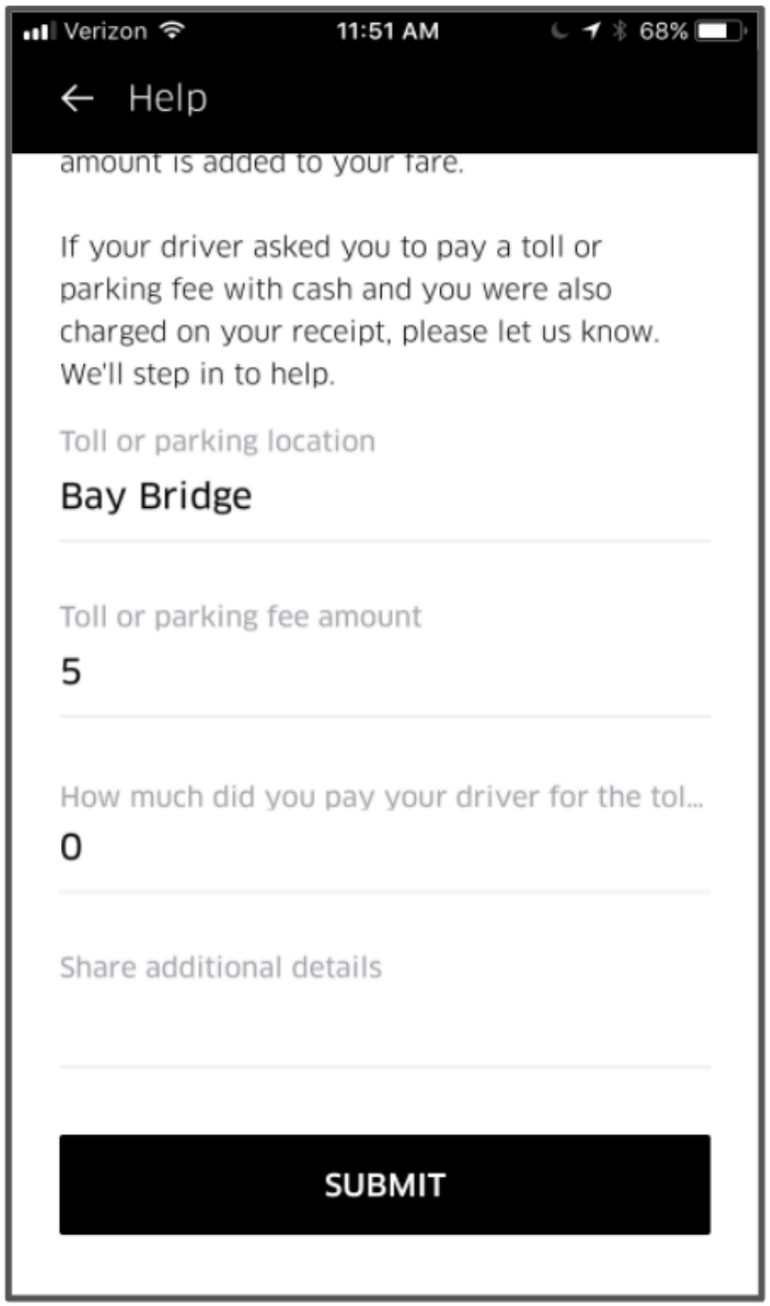
Have you ever wonder how Uber quickly resolve your issues?



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.

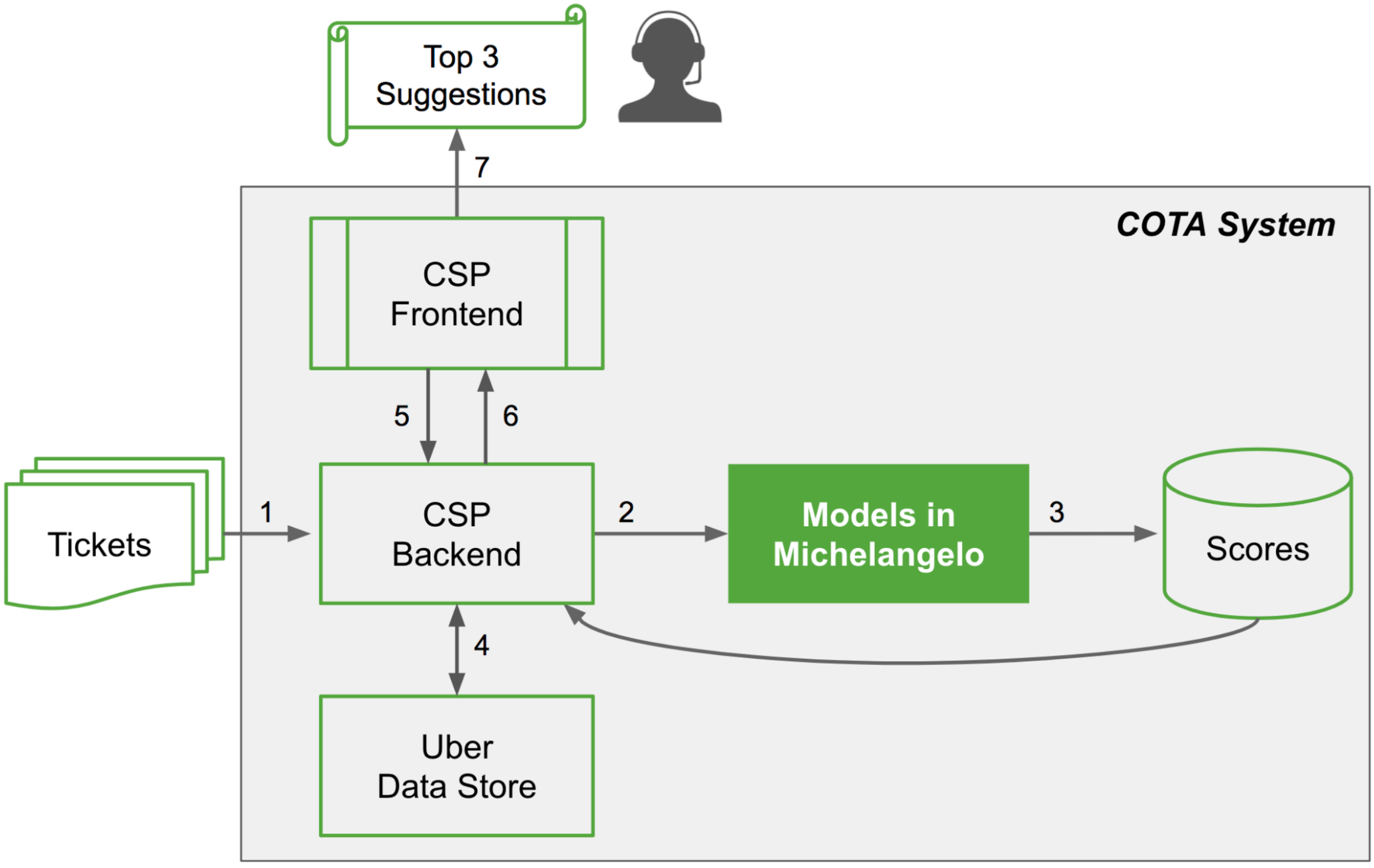
- In fact, when an agent opens a ticket, the first thing they need to do is determine the issue type out of thousands of possibilities—no easy task!

- 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.

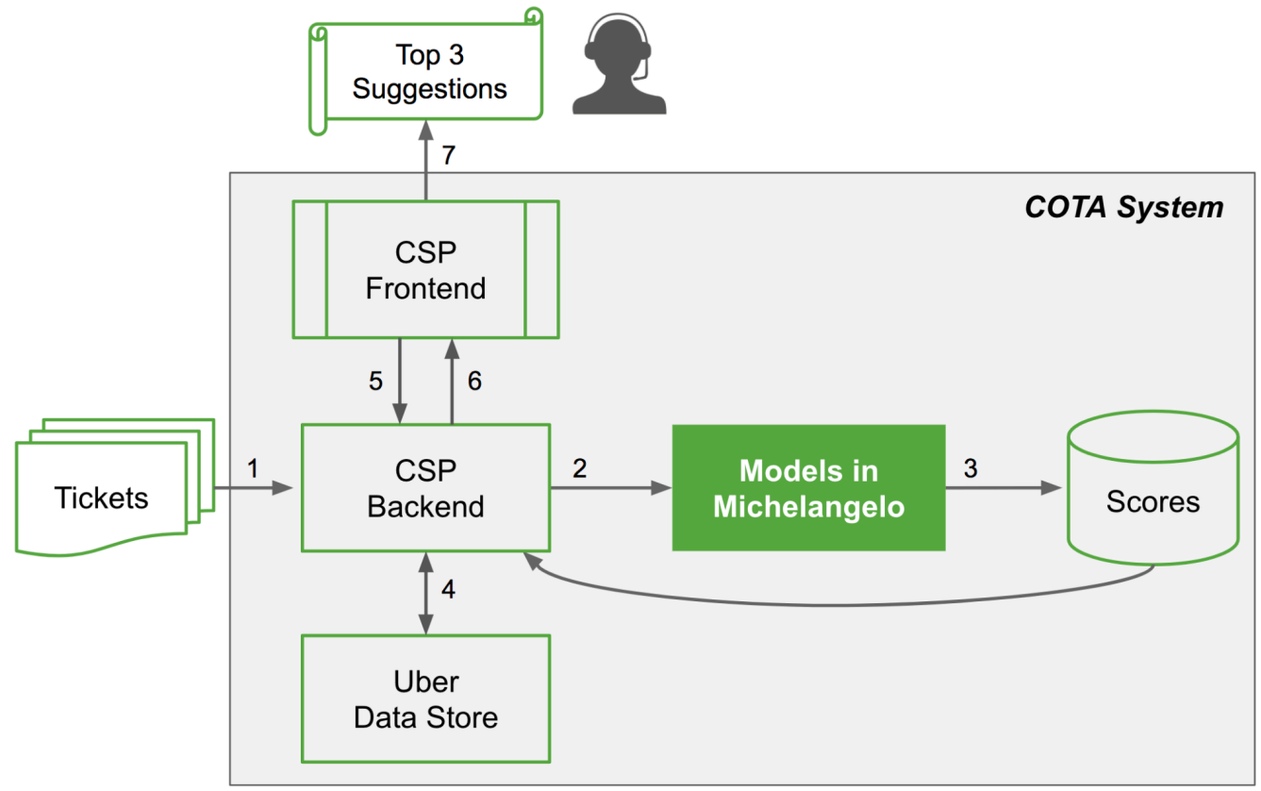
COTA: Customer Obsession Ticket Assistant



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.

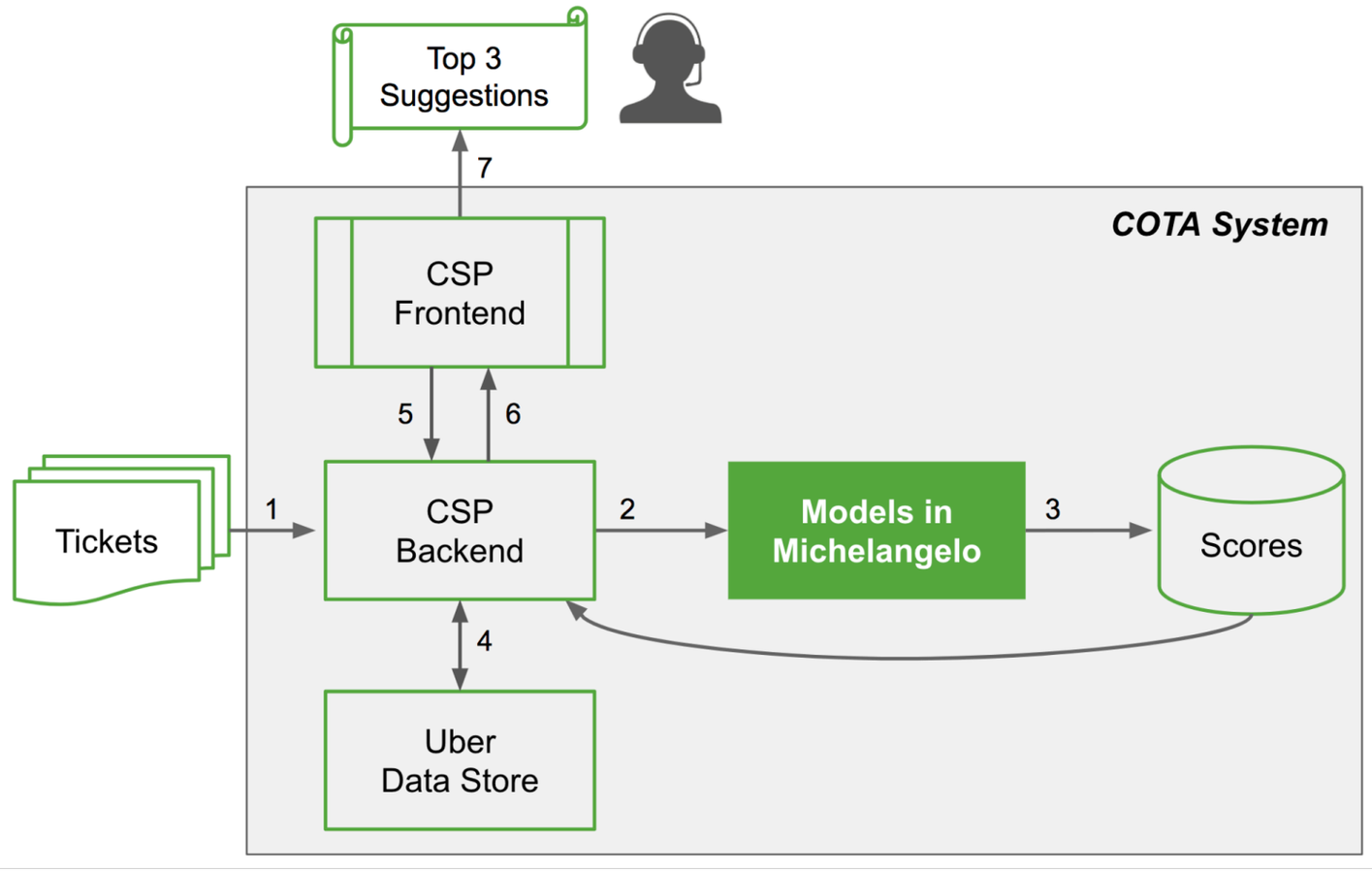
Is COTA an ML system?



COTA Architecture

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

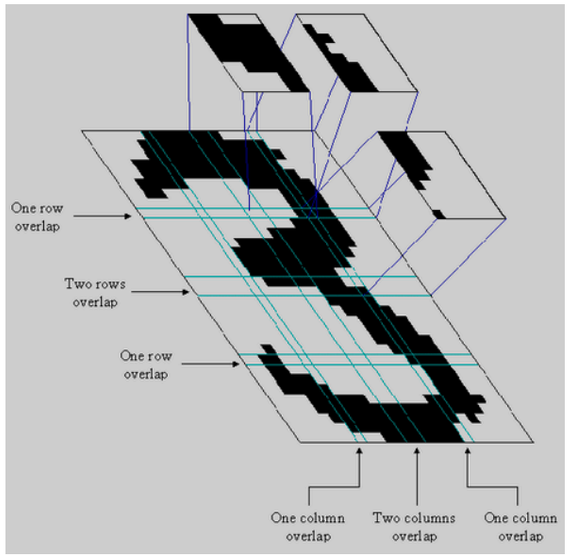
2. The back-end service then sends these features to the machine learning model in Michelangelo.

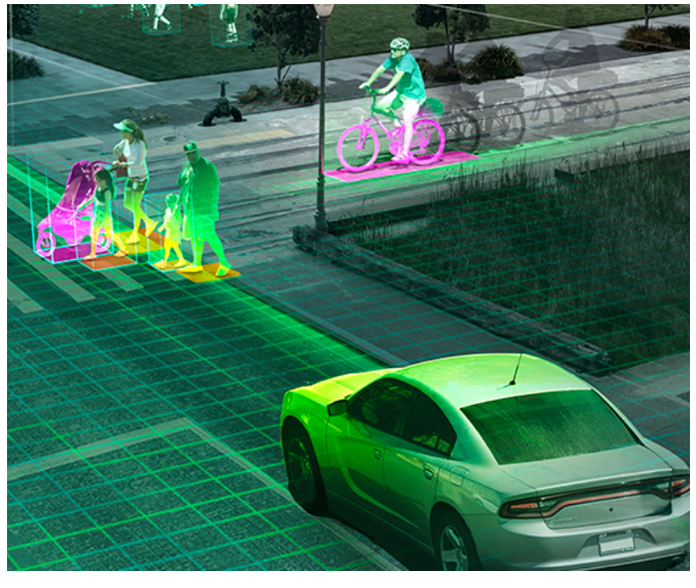


Wait, what is a feature?

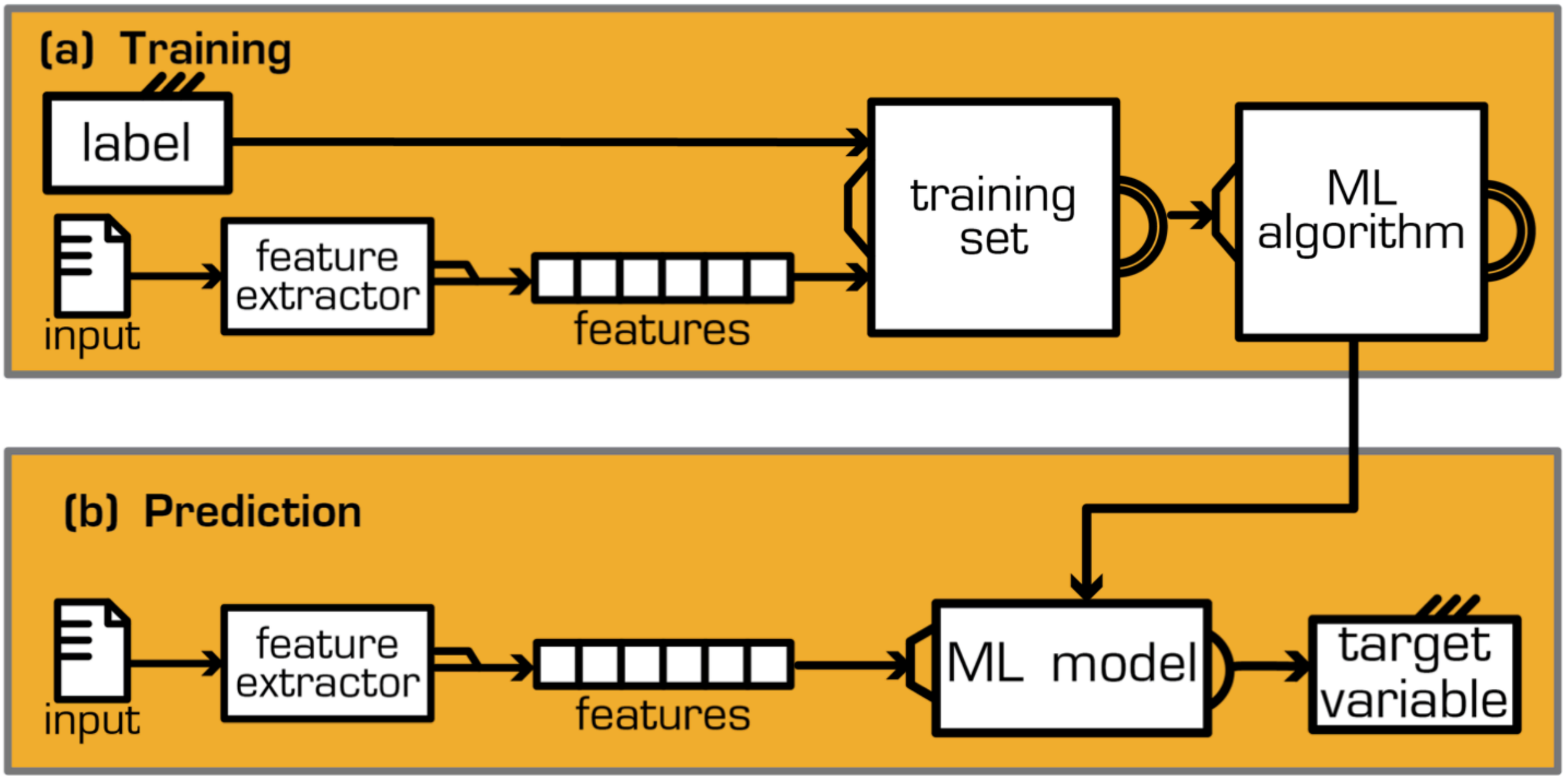
- A feature is an individual measurable property or characteristic of a phenomenon being observed.

What are the important features in each of these domains?





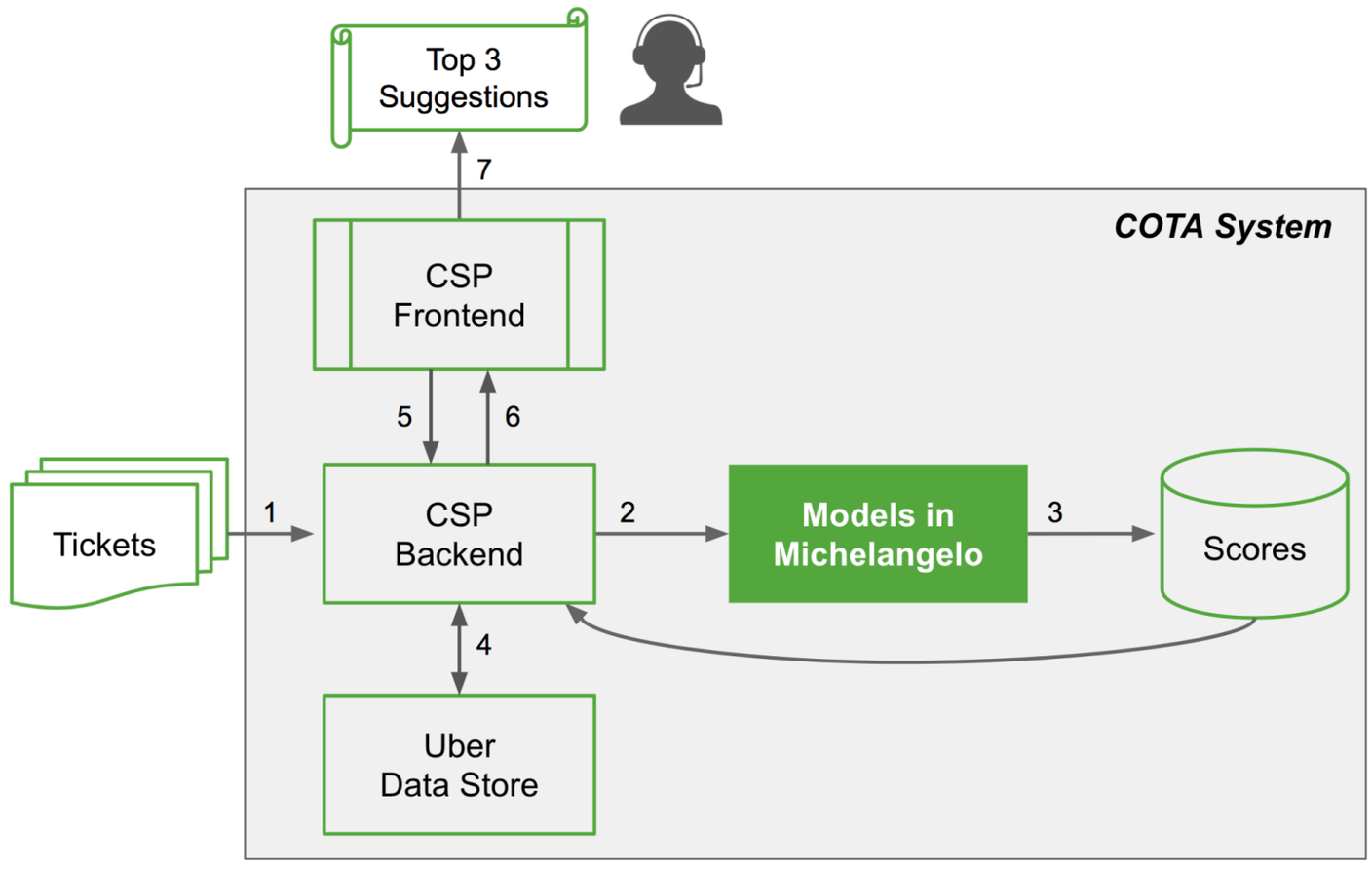
How features will be used in training and testing?



COTA Architecture

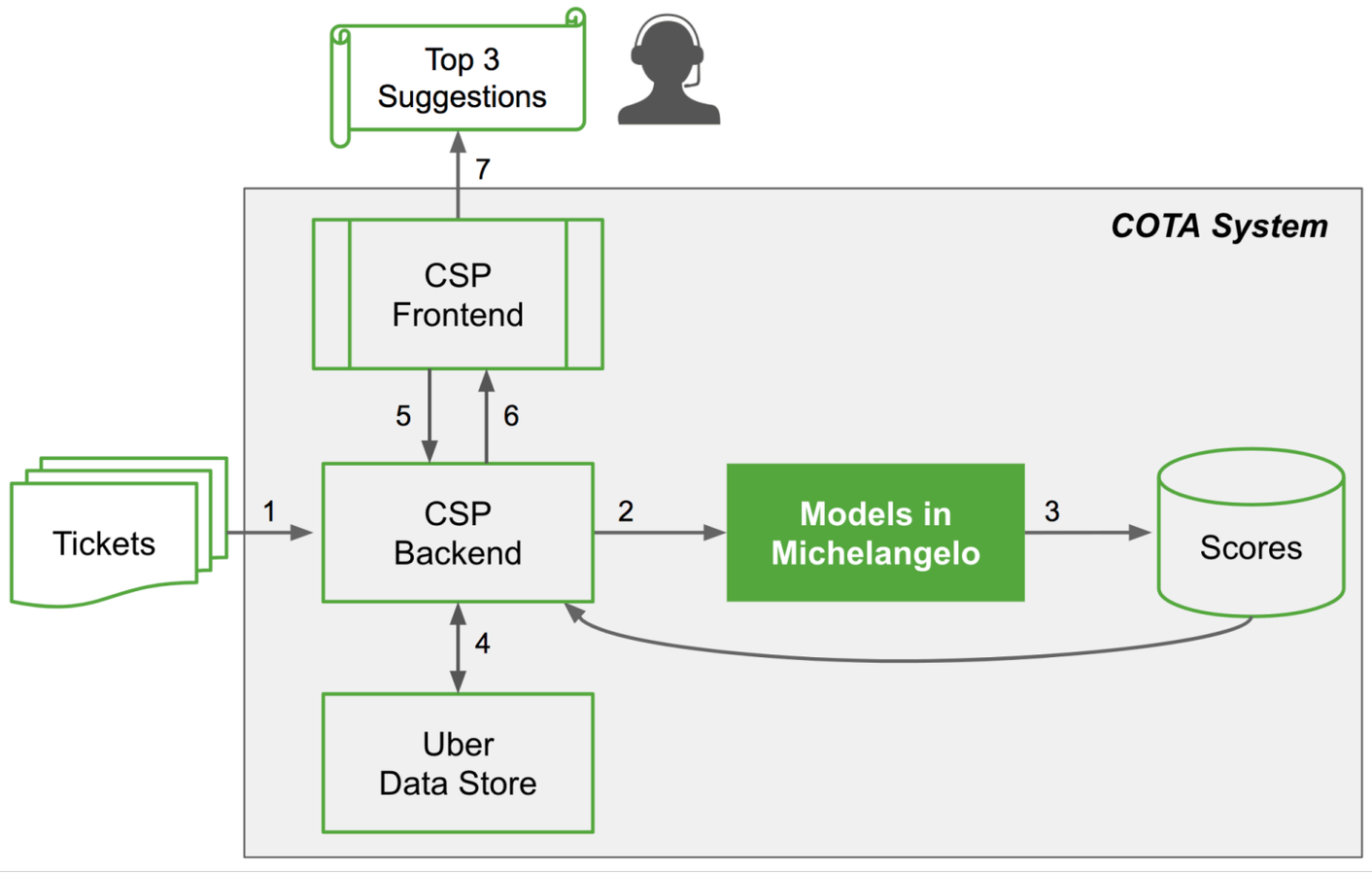
3. The model **predicts scores** for each possible solution.

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



COTA Architecture

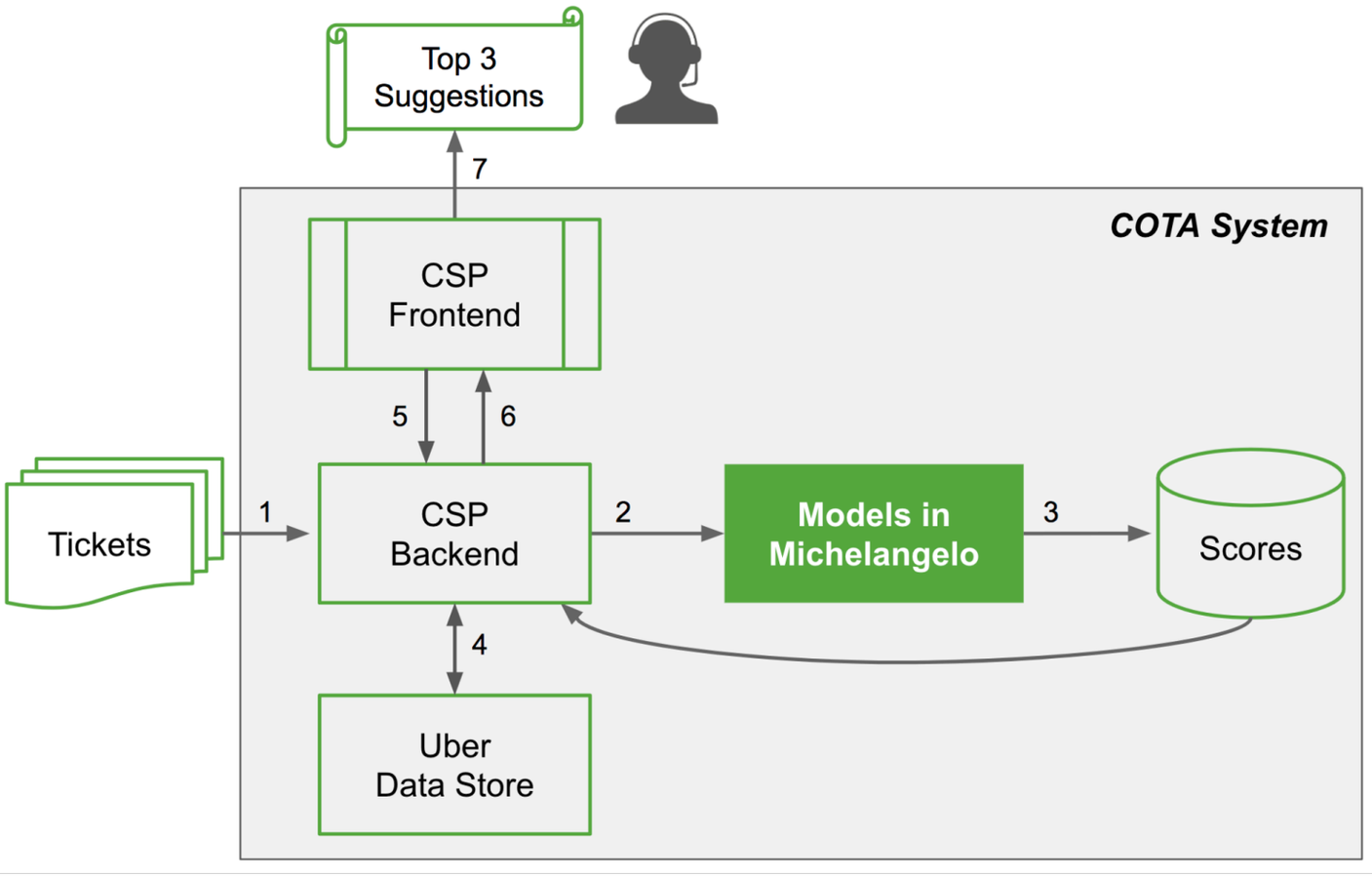
5. Once an agent opens a given ticket, the front-end service triggers the back-end service to check if there are any updates to the ticket. If there are no updates, the back-end service will retrieve the saved predictions; if there are updates, it will fetch the **updated features** and go through steps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

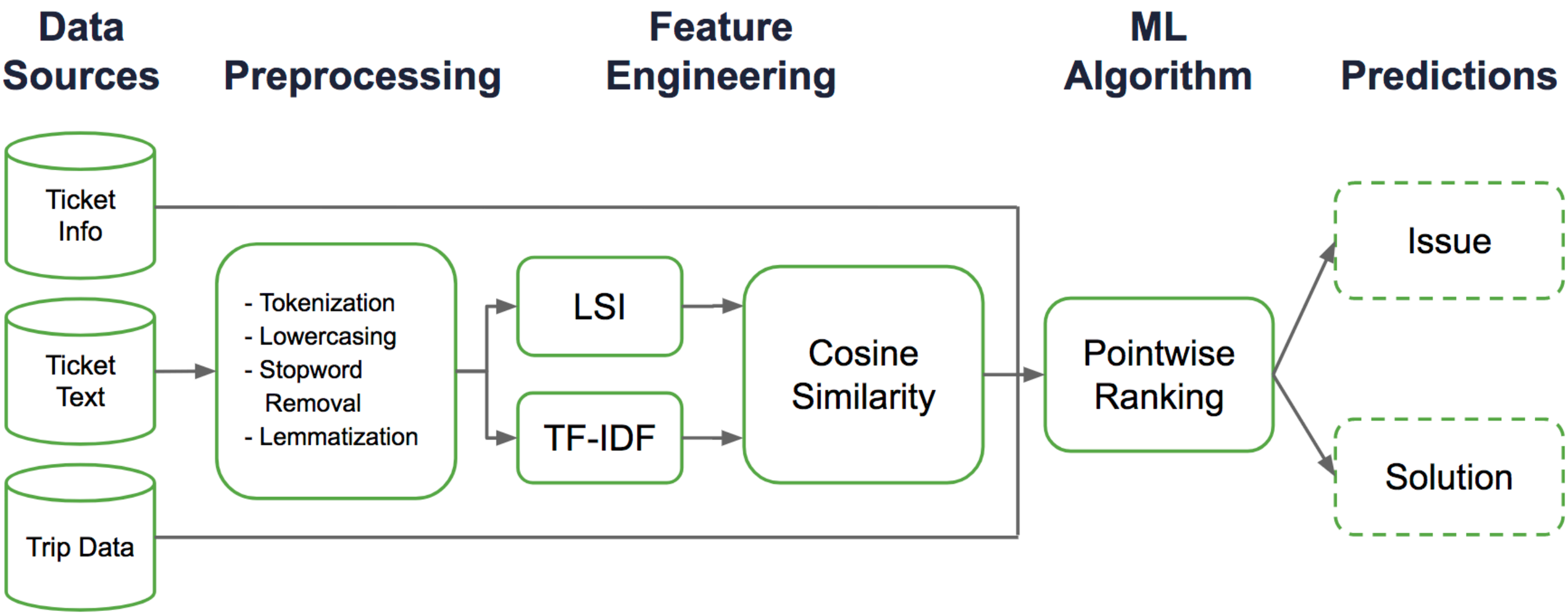
Any guess what is the most important feature of the ticket you submit to Uber?

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.

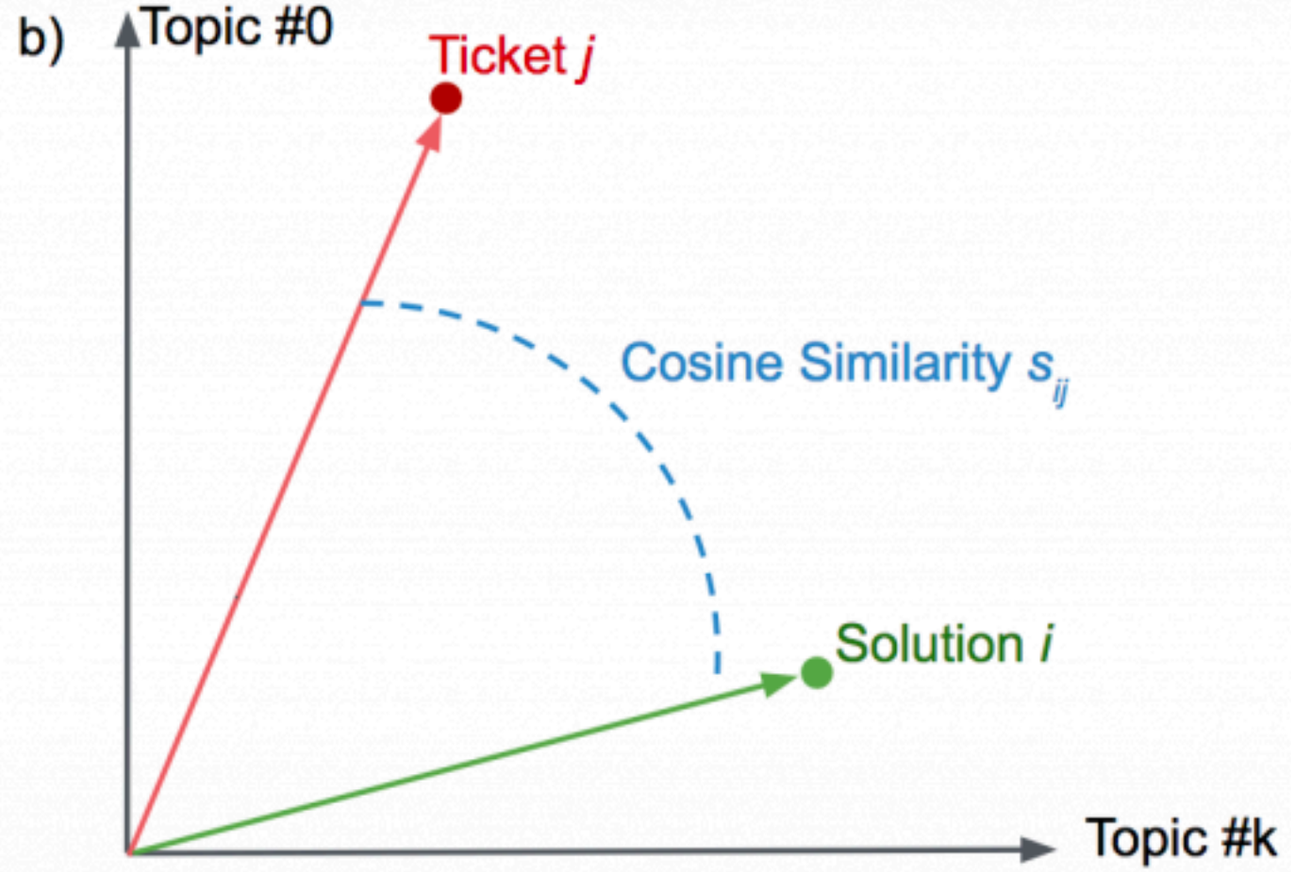
- Conducting **lemmatization** to convert words in different inflected forms into the same base form.

- Finally, **converting** the documents into a collection of words (a so-called bag of words) and build a dictionary of those words.

Topic modeling

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

Feature Engineering: All the solutions and tickets are mapped to the topic vector space, and cosine similarity between solution and ticket pairs are computed.



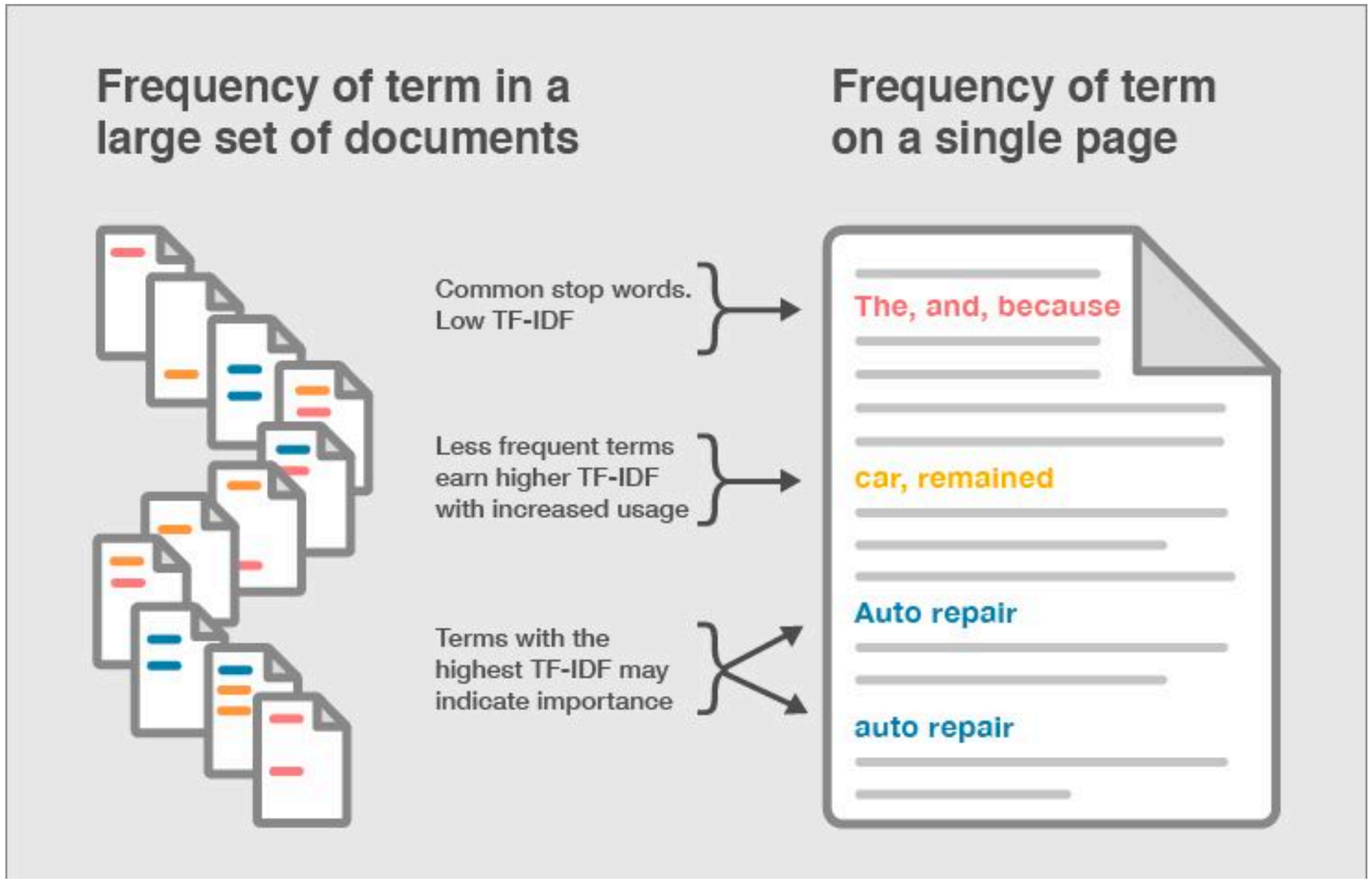
TF-IDF?

- Tf - Term Frequency
- Idf - Inverse document frequency

Quiz!

- Would you weight common words higher , or rare words?

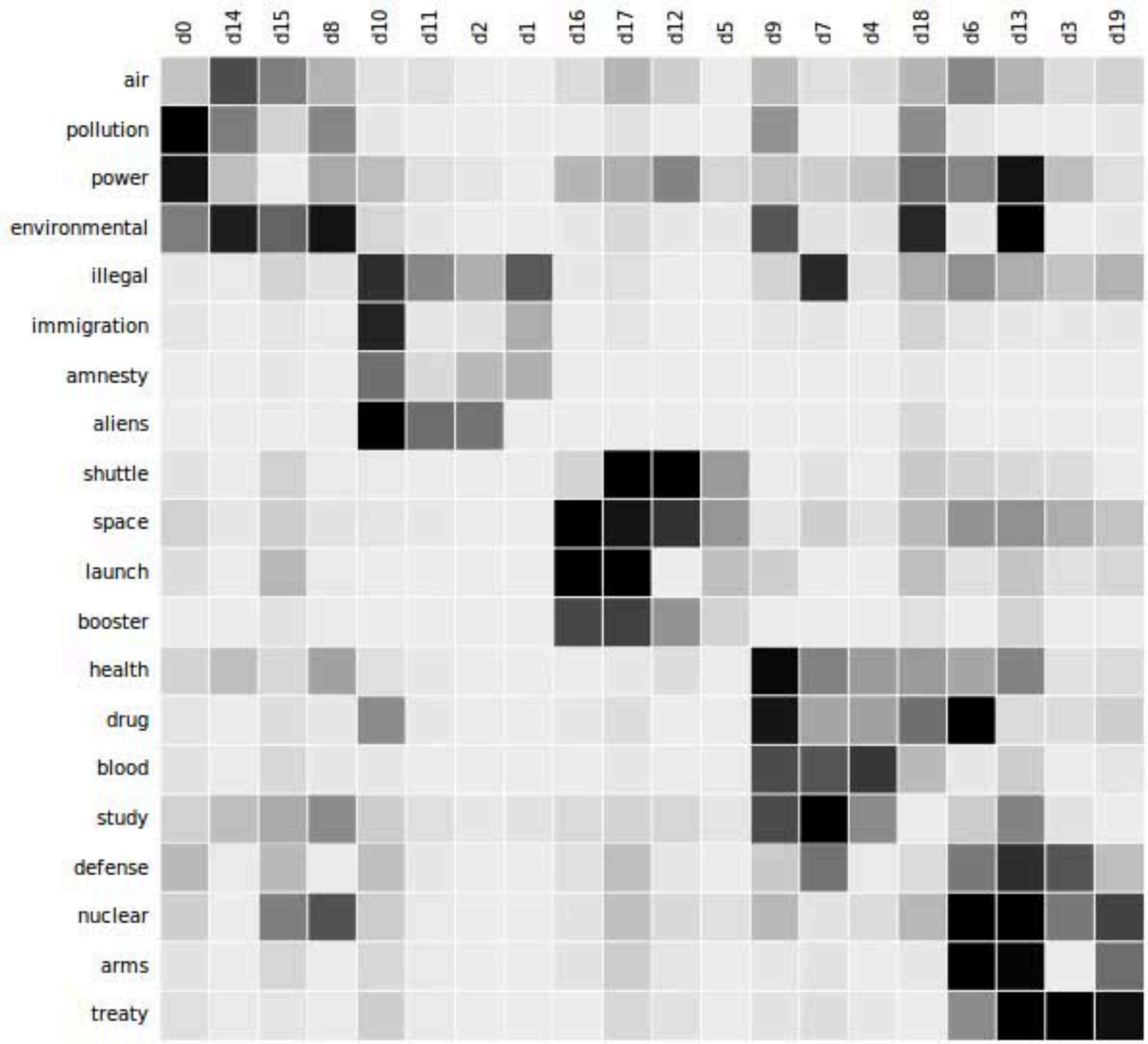
TF-IDF, ha?


Frequency of term in a large set of documents
Frequency of term on a single page
Common stop words. Low TF-IDF
Less frequent terms earn higher TF-IDF with increased usage
Terms with the highest TF-IDF may indicate importance
The, and, because
car, remained
Auto repair
auto repair

LSA?

- Latent Semantic Analysis

LSA, ha?


d0
d14
d15
d8
d10
d11
d2
d1
d16
d17
d12
d5
d9
d7
d4
d18
d6
d13
d3
d19
air
pollution
power
environmental
illegal
immigration
amnesty
aliens
shuttle
space
launch
booster
health
drug
blood
study
defense
nuclear
arms
treaty

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

- 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.

Cosine similarity, ha?

$$\mathbf{A} \cdot \mathbf{B} = \| \mathbf{A} \| \ \| \mathbf{B} \| \cos\theta$$

Cosine similarity, ha?

$$\textbf{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Pointwise ranking

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

Do you guess what is the challenge now?

Solution space becomes large

- With over **1,000** possible solutions
- For **100s** of ticket types,

- Solution space becomes a challenge for the ranking algorithm of distinguishing the fine differences between these solutions.

How Uber solved the challenge?

- Learning to rank!

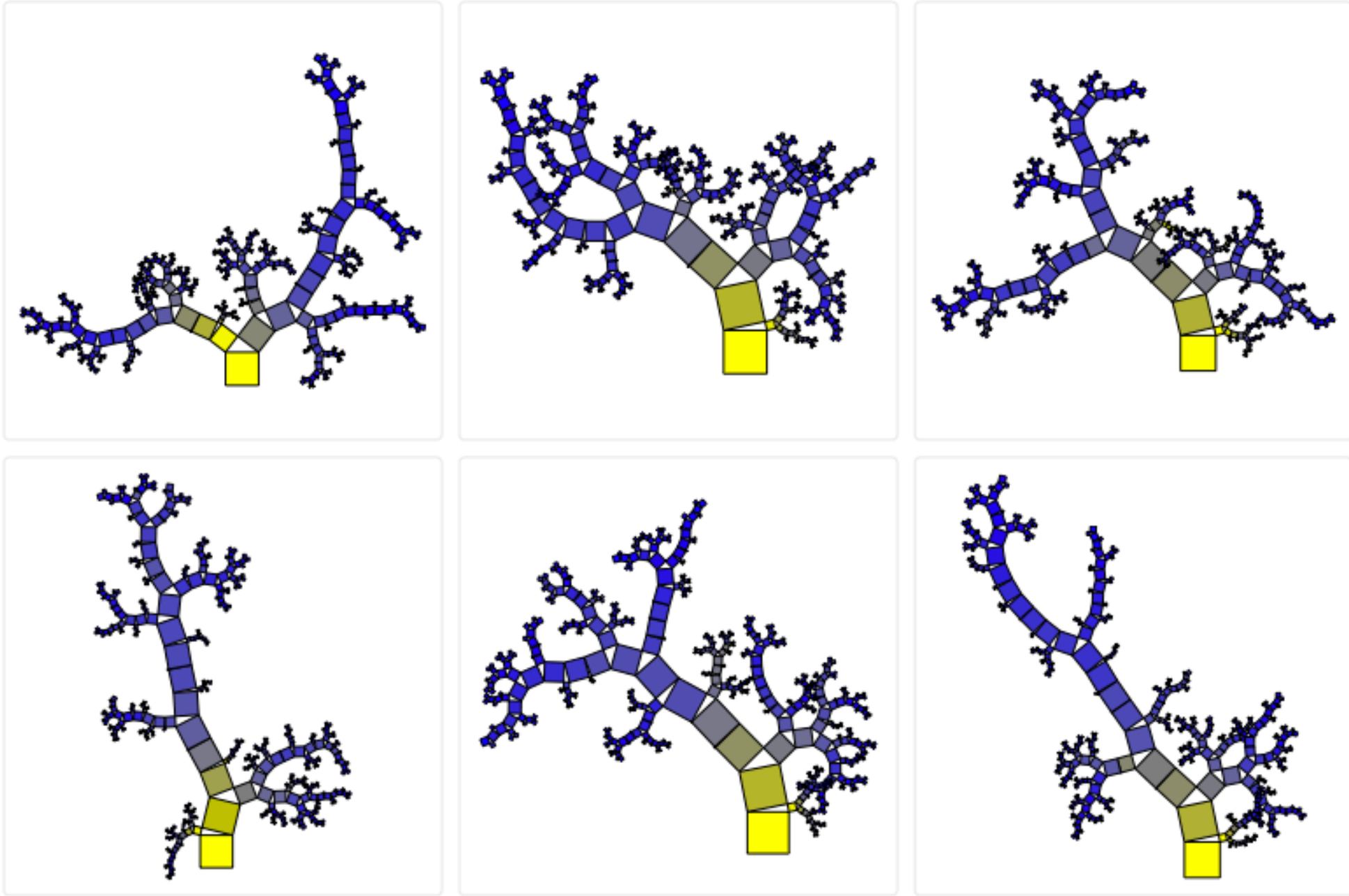
How Uber solved the challenge?

- 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**

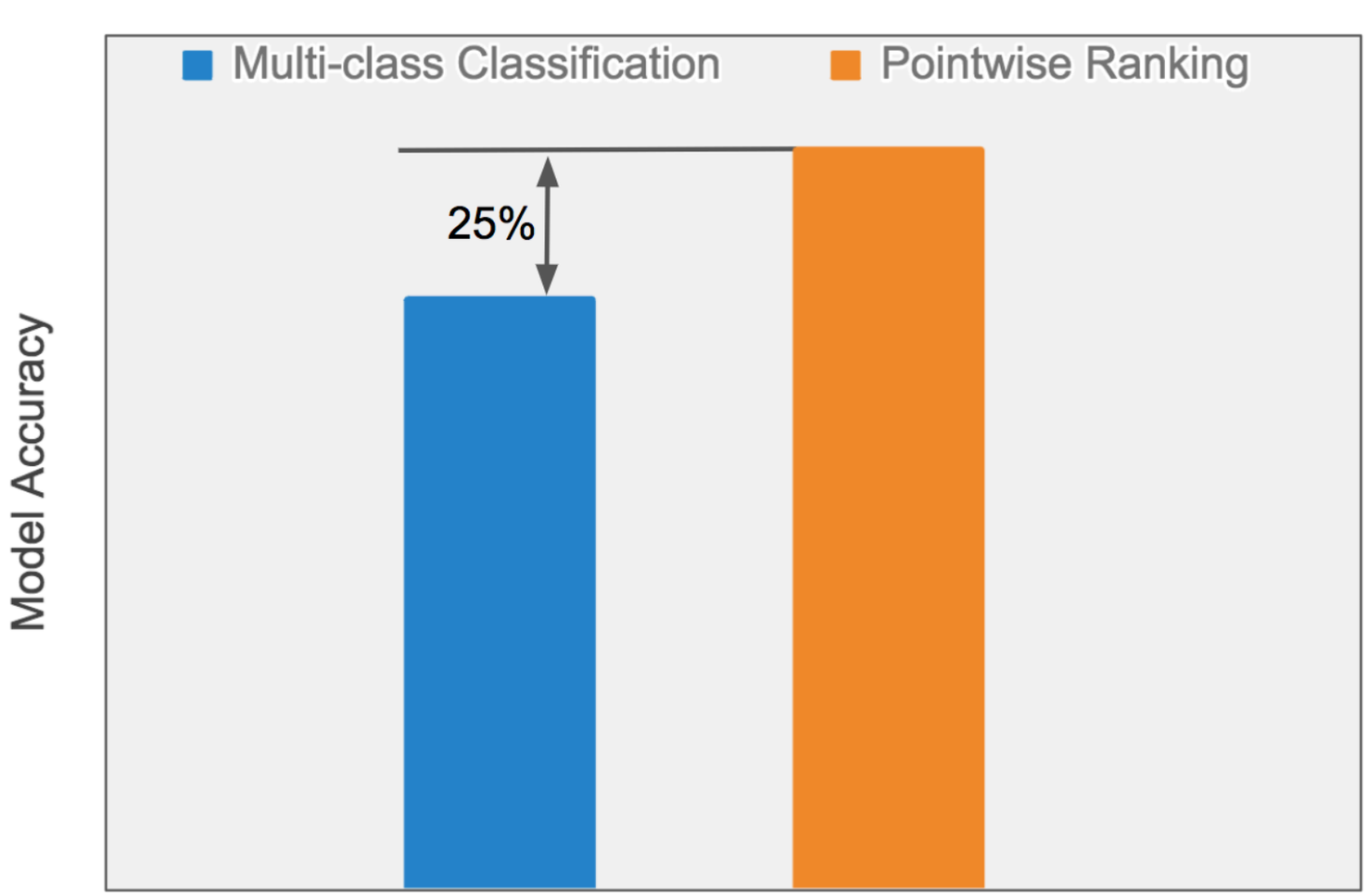
- Using the cosine similarity as well as ticket and trip features, they built a **binary classification** algorithm that leverages the **random forest** technique to classify whether or not each solution-ticket combination matches.

Random forest, ha?



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 customer support experience, Uber 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

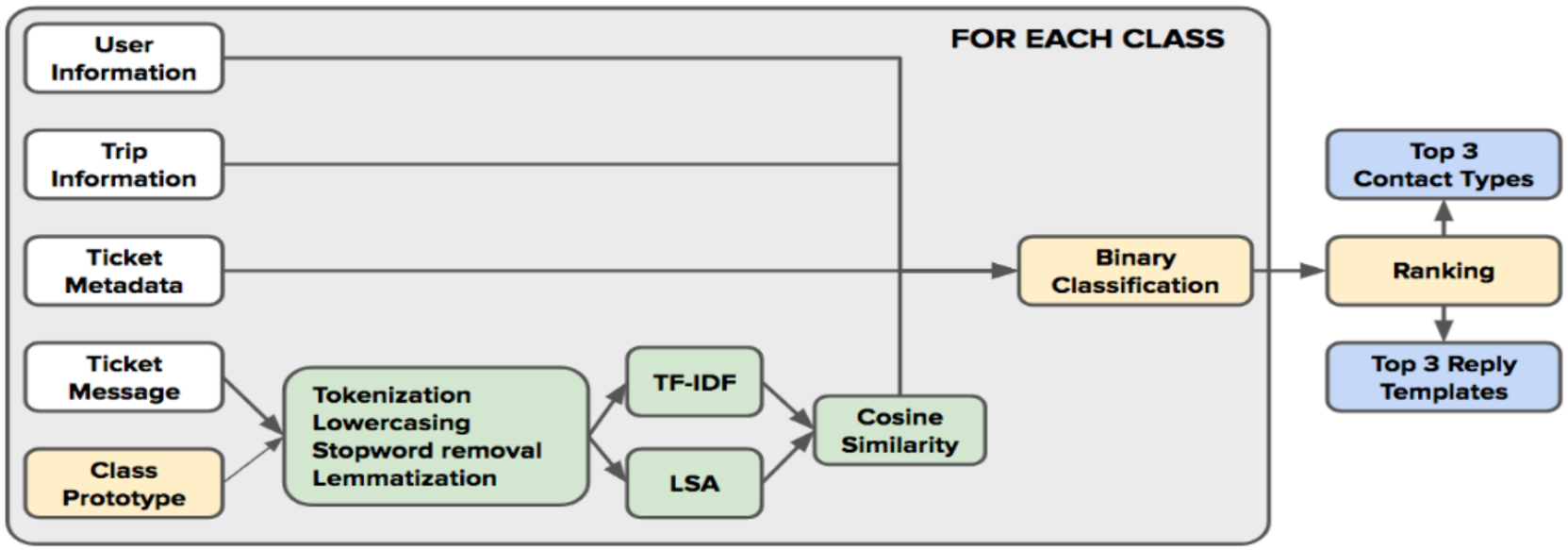
- Measured the online model performance for both groups and compared them with offline performance. The model **performance is consistent** from offline to online.

- 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**.

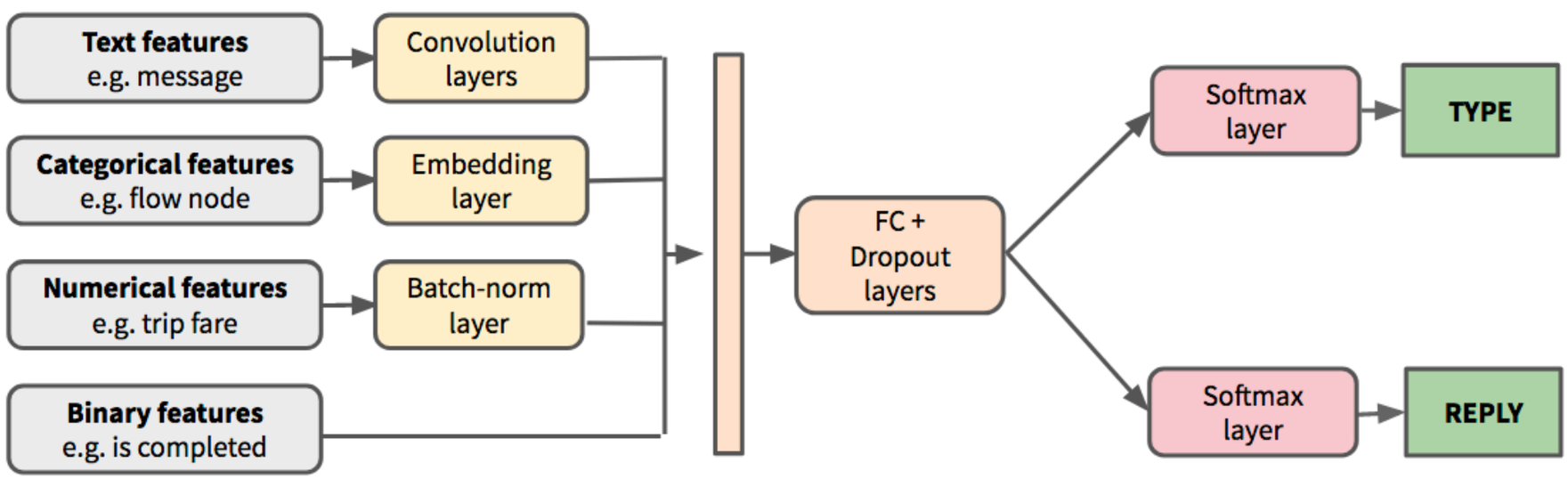
- To determine how much COTA affected ticket resolution speed, we compared the average ticket handling time between the control and treatment groups. On average, this new feature reduced ticket **handling time by about 10 percent**.

Next?
Moving to COTA v2 with deep learning

a)



b)



Moving to COTA V2

- Building a **Spark-based deep learning Pipeline** to productize the second generation of COTA (COTA v2)

- Given that model performance decays over time, we also built a **model management pipeline** to automatically retrain and retire models to keep them up-to-date.

Challenges of COTA V1

- COTA v1 conducted **negative sampling** in an overly complex way that made it difficult to train our models.

- Original implementation was not **extensible** enough to be used by future NLP models by other teams.

Why Deep learning?

- Serving more than 600 cities worldwide,
- supporting multiple languages,
- facilitating over five communication channels,
- Uber’s customer support reaches customers across businesses including ridesharing, Uber Eats, bikesharing, and Uber Freight.

Why Deep Learning?

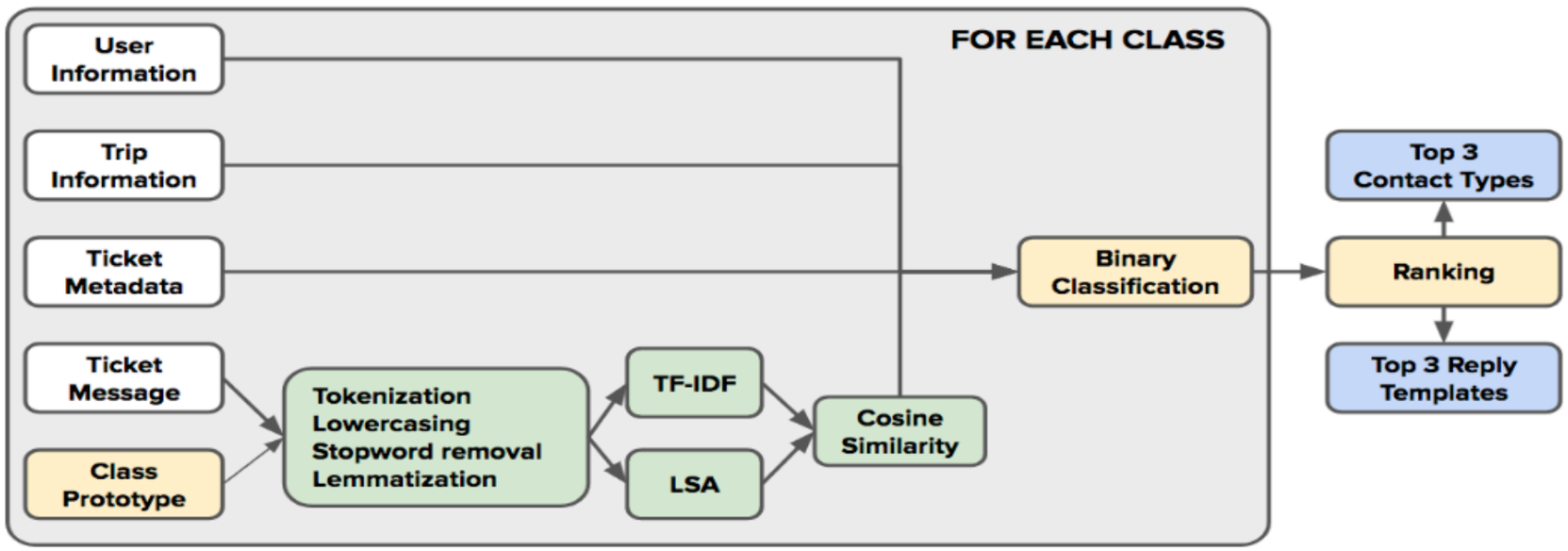
- Deep learning models have already outperformed humans on some **image classification and recognition** tasks.

- For the task of using retina photographs to detect **diabetic eye disease**, Google has shown that deep learning algorithms perform on-par with ophthalmologists.

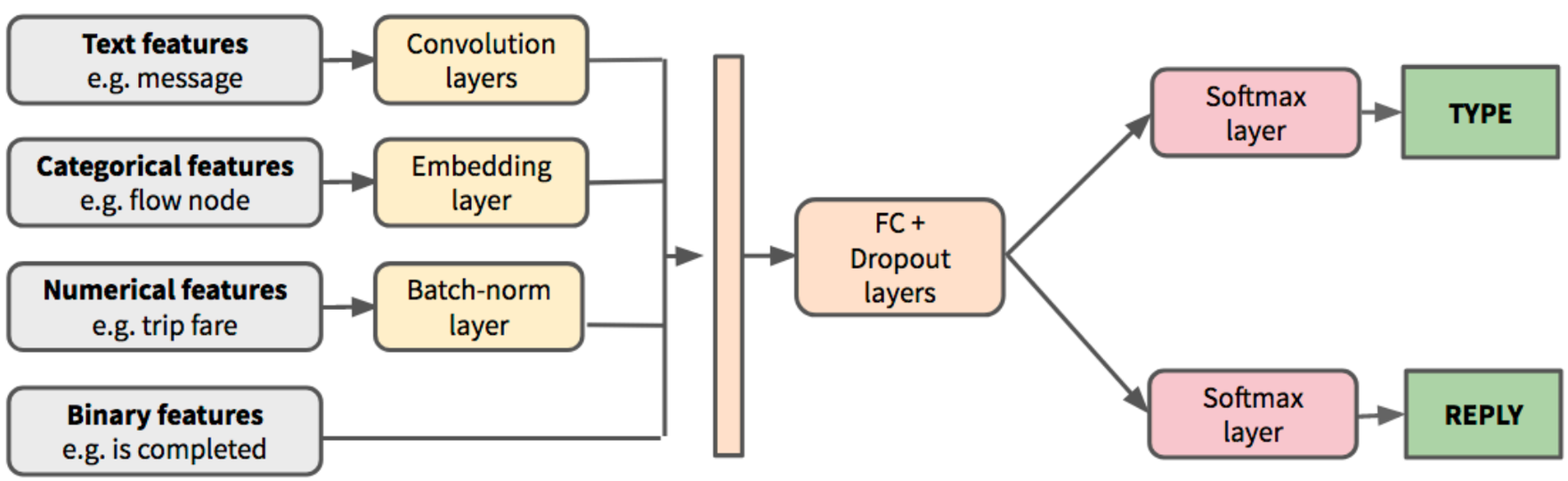
- The recent success of **AlphaGo** demonstrates that deep learning algorithms combined with reinforcement learning can even beat the world's best human Go players in what is often considered humankind's most complicated board game.

COTA V1 vs V2

a)



b)



Moving to COTA v2 with deep learning

- NLP Pipeline was built to process incoming ticket messages.

- Topic modeling was used to extract feature representation from the text feature.

- Additional feature engineering was used to generate cosine-similarity.

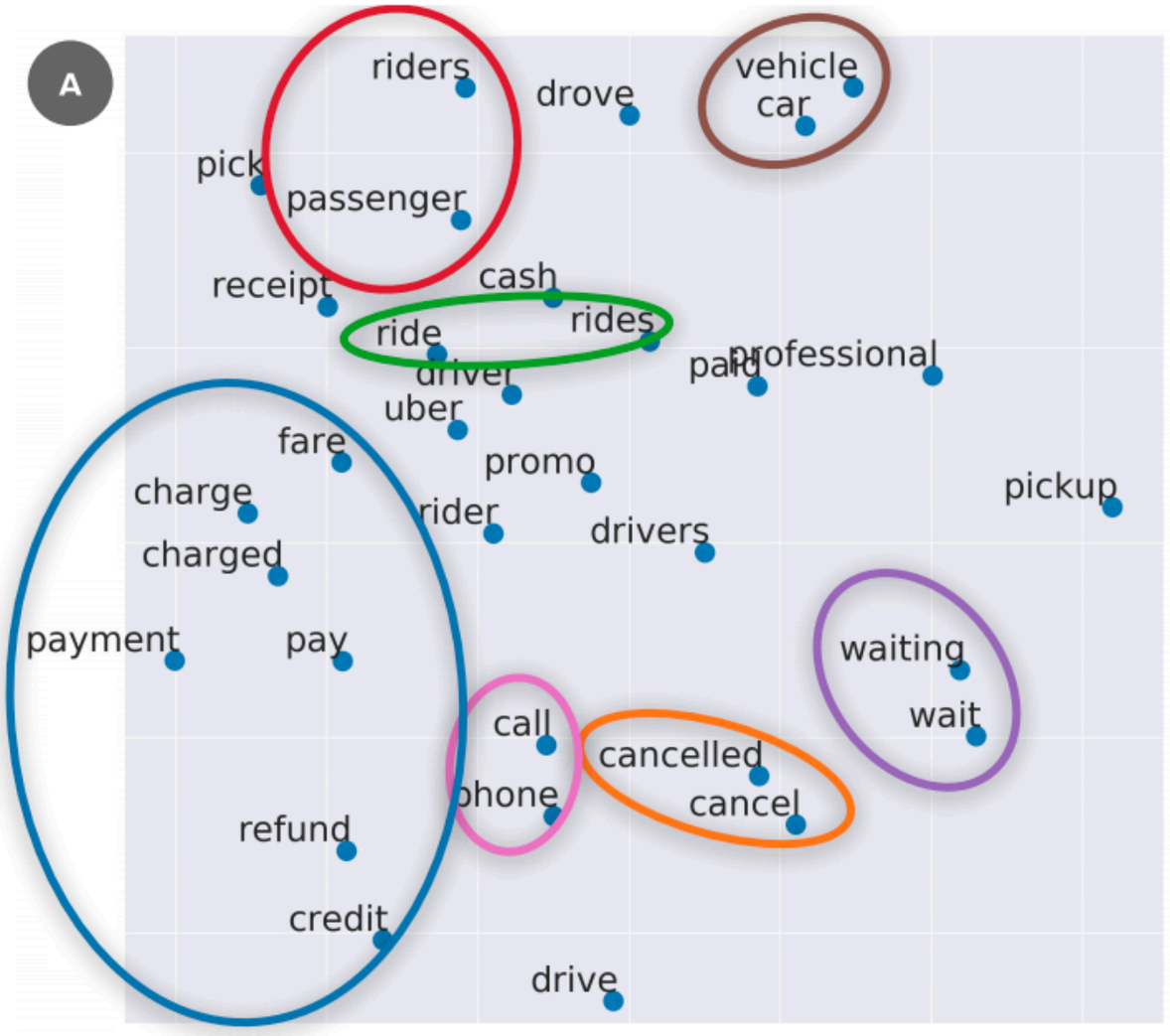
- Once each feature was engineered, all the features were fed into a binary point-wise ranking algorithm to predict the Contact Type and Reply responses.

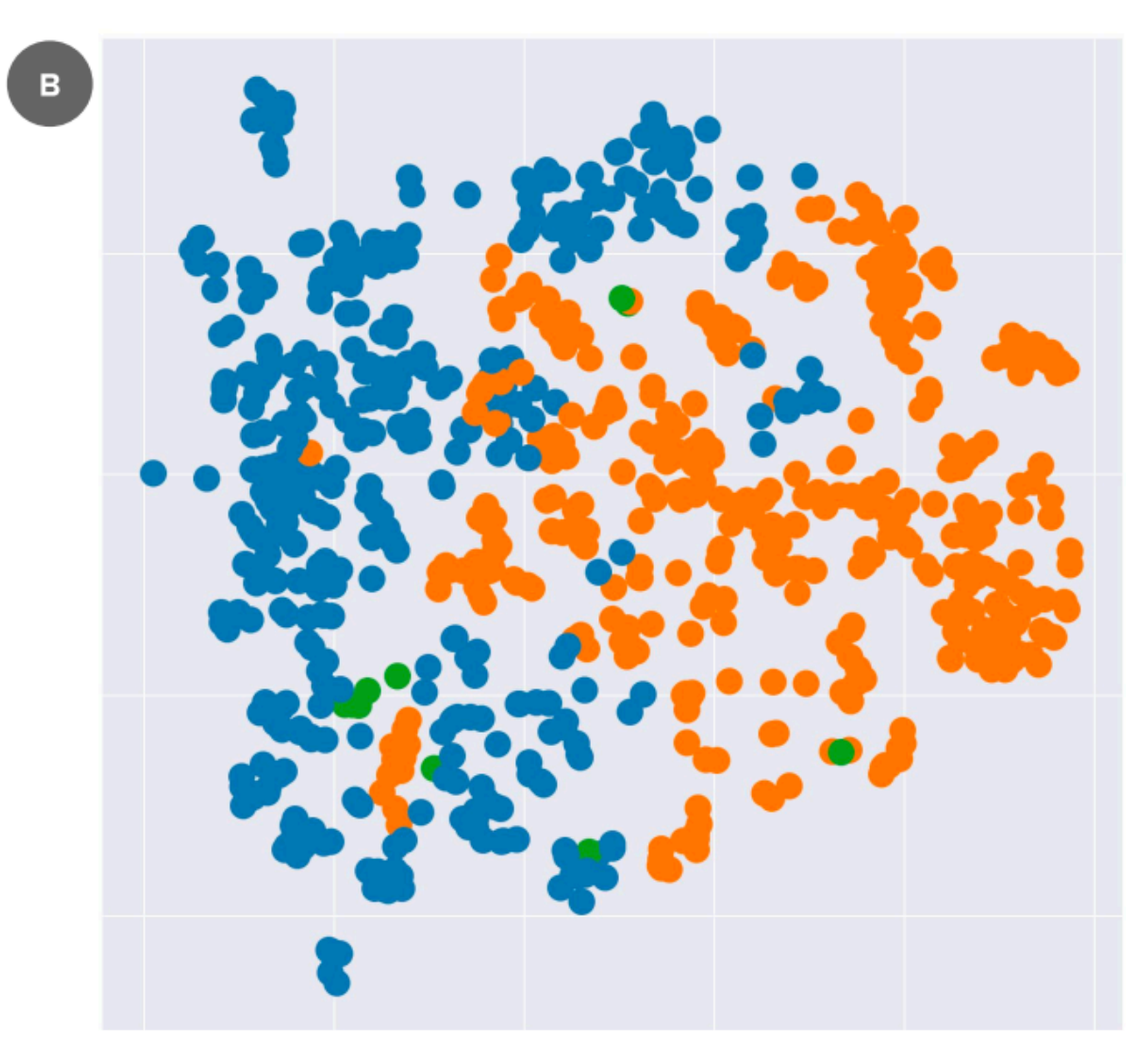
Moving to COTA v2 with deep learning

- The text feature goes through typical NLP preprocessing such as text cleaning and tokenization, and each word in the ticket is encoded using an embedding layer (not shown) to convert the word to a dense representation that further runs through convolution layers to encode the entire text corpus.

- Categorical features are encoded using an embedding layer to capture the closeness between different categories. Numerical features are batch normalized to stabilize the training process.

Embeddings learned by deep learning models





Results

- Deep learning can improve the solution's top-1 prediction accuracy by **16 percent** (from 49 percent to 65 percent) for the Contact Type model, and **8 percent** (from 47 percent to 55 percent) for the Reply model compared to COTA v1.

- This shows improvement of the customer support experience.

Challenges with COTA V2

- To leverage Spark for the NLP transformations in a distributed fashion.

- Spark computations are typically done using CPU clusters.

- On the other hand, deep learning training runs more efficiently on a GPU-based infrastructure.

- To address this duality, use both Spark transformations and GPU training, as well as build a unified Pipeline for training and serving the deep learning model.

Challenges with COTA V2

- Determining how to maintain **model freshness** given the dynamic nature of Uber's business.

- In light of this, a pipeline was needed to **frequently retrain and redeploy models**.

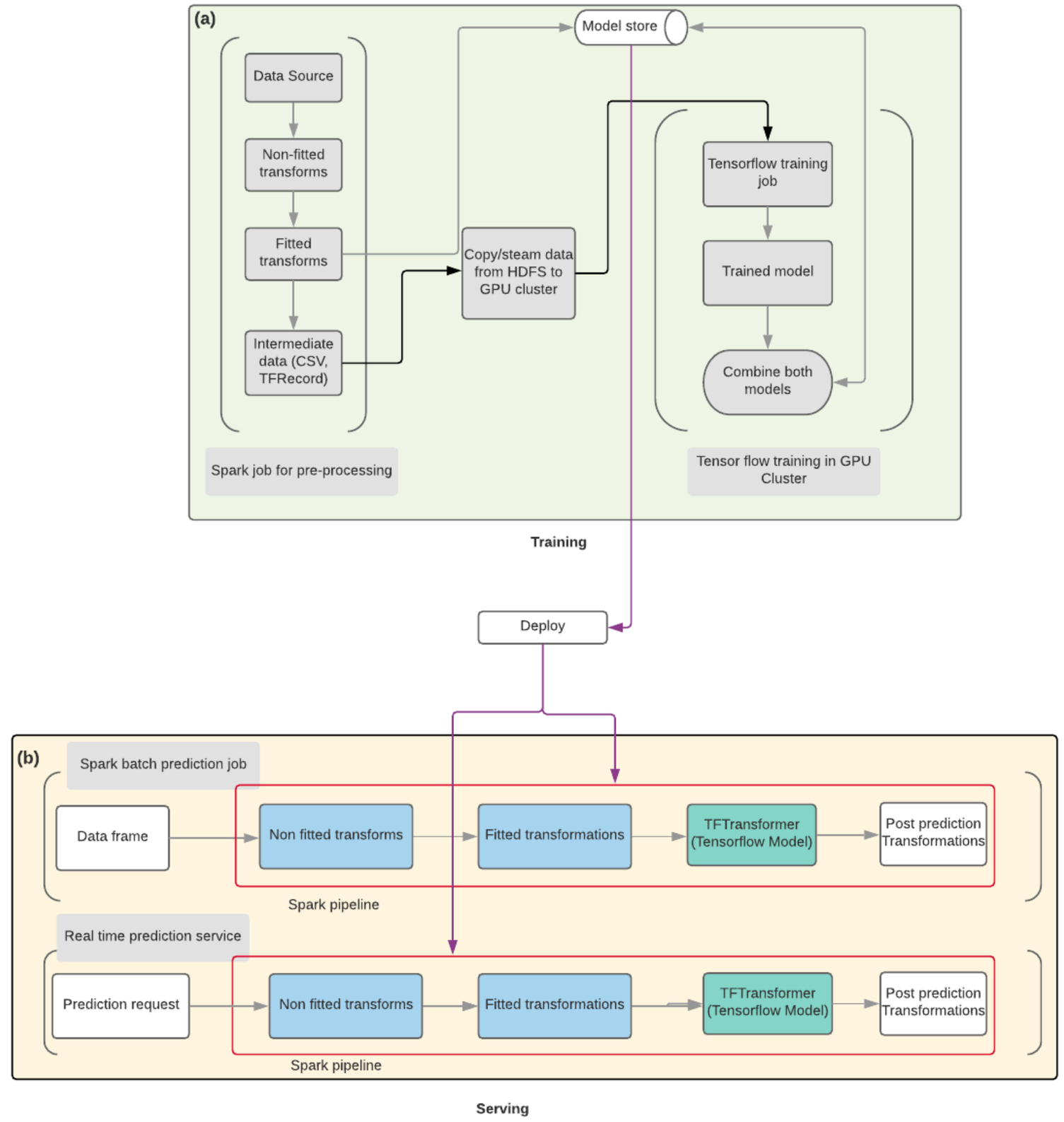
Solution to Challenges

- Deep learning Spark Pipeline (DLSP) to leverage both **Spark for NLP transformations** and GPUs for deep learning training.

- Integrated an internal job scheduling tool and built a model life-cycle management Pipeline (MLMP) on top of the DLSP, allowing us to schedule and run each job **at the frequency required**.

- These two pipelines enabled us not only to train and deploy deep learning models into Uber's production system, but also retrain and refresh the models to keep them at peak performance.

Deep Learning Spark Pipeline

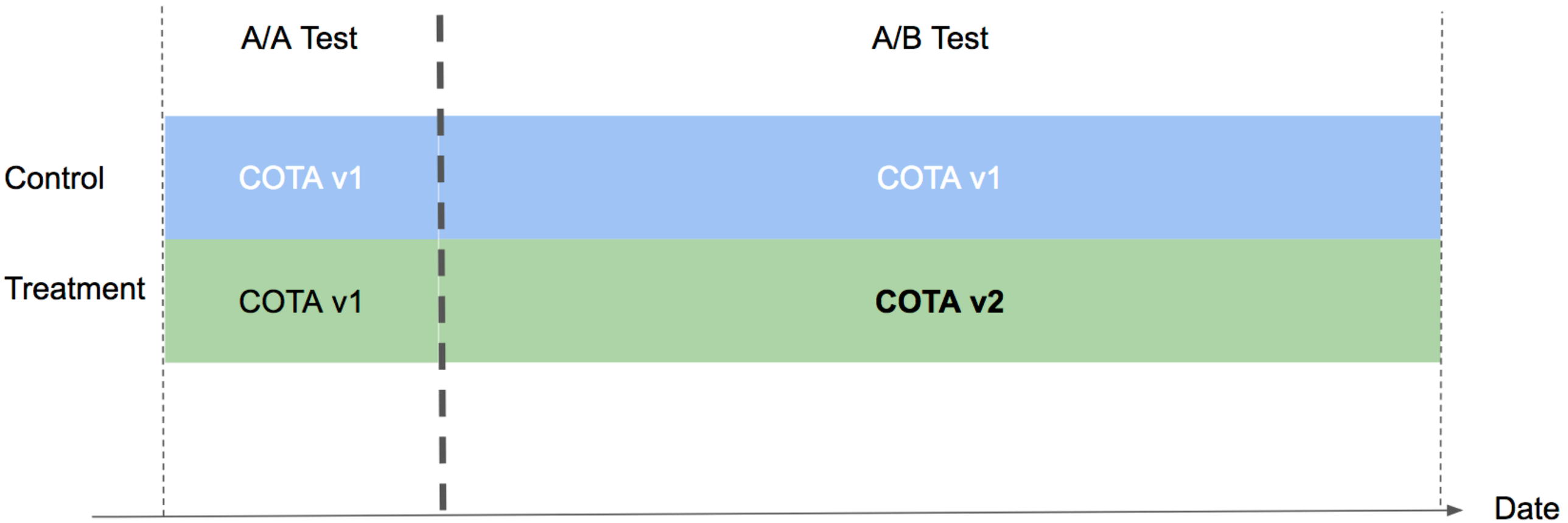


COTA v2's deep learning Spark Pipeline

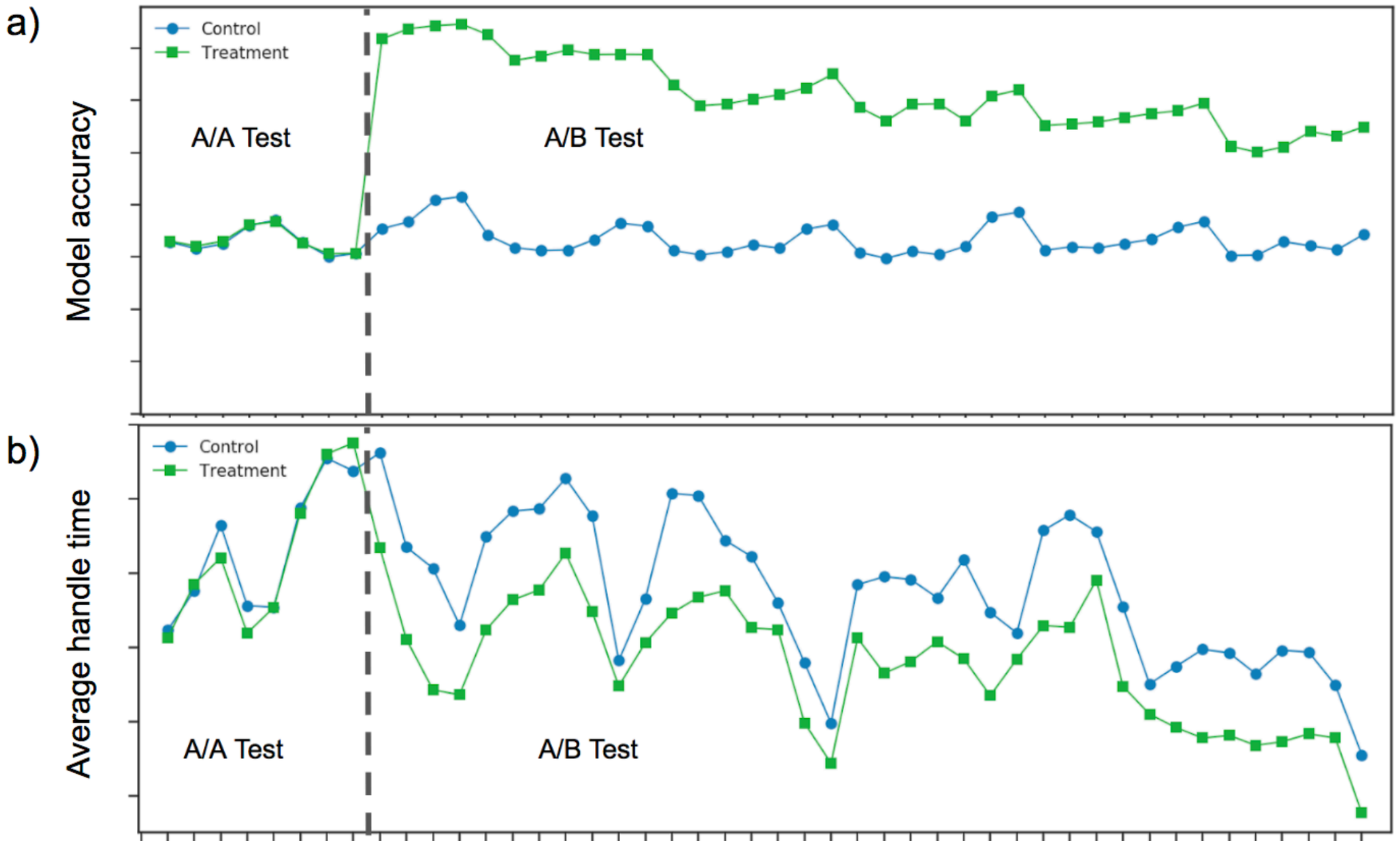
- The pipeline used for serving runs on a Java Virtual Machine (JVM). The performance we see while serving has a latency of **p95 < 10ms**, which demonstrates the advantage of low latency when using an existing JVM serving infrastructure for deep learning models.

- By extending Spark Pipelines to encapsulate deep learning models, we were able to leverage the best of both CPU and GPU-driven worlds: 1) the **distributed computation** of Spark transformations and **low-latency serving** of Spark Pipelines using CPUs and 2) the **acceleration of deep learning model training** using GPUs.

Overall test strategy to compare the COTA v1 and COTA v2 systems.



Given enough training data our COTA v2 deep learning models can significantly outperform the classical COTA v1 machine learning models



Summary

- We went through a ML system with traditional NLP pipeline that scale to Uber.

- We also discussed challenges that Uber faced and how they came up with a solution that scales well.

- We discussed transition to a deep learning pipeline that identifies features via ConvNets automatically without feature engineering.

- In addition to improving the customer support experience, COTA v2 will also save the company millions of dollars every year by streamlining the support ticket resolution process.

Summary

So now we have a much better understand of how a machine learning system looks like and what are the building blocks of such systems.