Overview

• ML Use Cases at Twitter
• ML Platform Requirements & Challenges
• Unifying Twitter Around a Single ML Platform
• Technology Migrations
• Health ML Use Case
• Summary of Lessons Learned
• Future of Our ML Platform
Overview

- **ML Use Cases at Twitter**
  - ML Platform Requirements & Challenges
  - Unifying Twitter Around a Single ML Platform
  - Technology Migrations
  - Health ML Use Case
  - Lessons Learned
  - Future of Our ML Platform
ML Use Cases: Tweet Ranking

- User Features Split Group
- User-Author Split Group
- Tweet Features Split Group

User Embedding

U-A Embedding

Tweet Embedding

tf.concat

Dense Neural Network

Objective
pCTR = p ( "click" | if we show this Candidate Ad to this User in this Context)
ML Use Cases at Twitter

• Other use cases
  • Recommending Tweets, Users, Hashtags, News, etc.
  • Detecting Abusive Tweets and Spam
  • Detecting NSFW Images and Videos
  • And so on …
ML Use Cases at Twitter

ML is Everywhere

Eleanor Harding • @t... • 08/01/2018
Couple of weeks at home in SA and my
gallery is now 80% cat photos. She’s
called Phish. She likes to mlem.

Eleanor Harding • @t... • 08/01/2018
Couple of weeks at home in SA and my
gallery is now 80% cat photos. She’s
called Phish. She likes to mlem.
Overview

• ML Use Cases at Twitter
• **ML Platform Requirements** & Challenges
  • Unifying Twitter Around a Single ML Platform
  • Technology Migrations
  • Health ML Use Case
• Summary of Lessons Learned
• Future of Our ML Platform
Requirements of ML Platform

Data Scale
PBs of data per day
Some models train on Tens of TBs of data per day
Requirements of ML Platform

Prediction Throughput

Tens of millions of predictions per second
Requirements of ML Platform

Prediction Latency Budget
tens of milliseconds
Example Use Case
Ads Prediction

- Predictions every second: $10+M$
- Serving latency: 40ms
- Features: $1+M$
- Training examples everyday: $1+B$
Overview

• ML Use Cases at Twitter
• **ML Platform Requirements & Challenges**
  • Unifying Twitter Around a Single ML Platform
  • Technology Migrations
  • Health ML Use Case
  • Summary of Lessons Learned
  • Future of Our ML Platform
Challenges of Old ML Platform

Fragmentation of ML Practice

- VW
- Scikit Learn
- TensorFlow
- PyTorch
- Lua Torch
- In-house Frameworks
Challenges of Old ML Platform

- Difficulty in Sharing
- Models
- Knowledge
- Tooling & Resources
Challenges of Old ML Platform

- Inefficiencies
- Work Duplication
Example Duplicate Work

Various Ways to do
Model Training & Serving
Model Refreshes
Data Cleaning and Preprocessing
Experiment Tracking
Etc.
Overview

• ML Use Cases at Twitter
• ML Platform Requirements & Challenges
• **Unifying Twitter Around a Single ML Platform**
• Technology Migrations
• Health ML Use Case
• Lessons Learned
• Future of Our ML Platform
New Unified ML Platform Overview

A Single Consistent ML Platform Across Twitter

1. Pipeline Orchestration
2. Preprocessing and Featurization
3. Model Training and Evaluation
4. Experimentation Tracking
5. Production Model Serving
Overview

• ML Use Cases at Twitter
• ML Platform Requirements & Challenges
• Unifying Twitter Around a Single ML Platform
• Technology migrations
• Health ML Use Case
• Summary of Lessons Learned
• Future of Our ML Platform
Technology Migrations

- Data Analysis: Scalding + PySpark/Notebooks
- Featurization: Feature Store
- ML Frameworks: Java ML -> Lua Torch -> TensorFlow
- Training and deployment cycles: Apache Airflow
Data Analysis: Scalding

- **Scala**
- Abstraction over hadoop
- Distributed data processing
- Great for large scale data
- Slow-iteration

```scala
package com.twitter.scalding.examples
import com.twitter.scalding._
import com.twitter.scalding.source.TypedText

class WordCountJob(args: Args) extends Job(args) {
  TypedPipe.from(TextLine(args("input")))
    .flatMap { line => tokenize(line) }
    .groupBy { word => word } // use each word for a key
    .size // in each group, get the size
    .write(TypedText.tsv((String, Long))(args("output")))

  // Split a piece of text into individual words.
  def tokenize(text: String): Array[String] = {
    // Lowercase each word and remove punctuation.
    text.toLowerCase.replaceAll("[^a-zA-Z0-9\s]", "").split("\s+")
  }
}
```
Data analysis: Notebook + Spark

- iPython Notebook + PySpark
- Easier for Python engineers
- Data visualization
- Faster iteration
Lessons learned

ML Practitioner Diversity

Production ML Engineers
Deep Learning Researcher
Data Scientists
Featurization: Ad Hoc

- Teams use **common data sources**
  - E.g. user data, tweet data, engagement data
- Every team does their own featurization
  - Duplication of effort
- Difficult to validate features at serving time
  - Inconsistent featurization schemes for training vs serving
Featurization: Feature Store

- Teams can **share, discover and access** features
- **Consistent** training-time vs serving-time featurization

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Family</th>
<th>Group</th>
<th>Entity type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>core</td>
<td>user_profile</td>
<td>User</td>
<td>The bio of the user, limited to 140 characters. May include mentions of people,</td>
</tr>
<tr>
<td>ID: core.user_profile.description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>has_profile_description</td>
<td>core</td>
<td>user_profile</td>
<td>User</td>
<td>Whether the user has profile description</td>
</tr>
<tr>
<td>ID: core.user_profile.has_profile_description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lessons learned

Consistency

Consistency across teams => sharing & efficiency
Important: feature consistency between training and serving
ML Frameworks: Java ML

- Logistic regression
  - Relies on feature discretization
- Typically used in an online learning environment:
  - Model learns new data as it becomes available (~15 min delay)
ML Frameworks: Lua Torch

- Deep learning
- Feature discretization parity
- ML Engineers didn’t want to learn Lua:
  - Lua hidden via YAML
  - Hard to debug and unit test
- Complex production setup
  - JVM -> JNI -> Lua VMs -> C/C++
ML Frameworks: TensorFlow

- Google support
- Production ready
  - Export graphs as protobuf
  - Serve graphs from Java/Scala:
    - JVM -> TensorFlow
- TensorBoard
- Large ecosystem (E.g. TFX)
Lessons learned

Reproducibility is hard

... across different ML framework: small differences, large impacts
Online experiments take time
Need simple setup, fast iterations
Train and Deploy Cycles

Different approaches to productionizing training algorithms:

- **Manually** re-train and re-deploy the model periodically
  - Retraining frequency varies
- **Automate** training and deployment cycles:
  - Cron, Aurora, **Airflow** Jobs
  - Helps reduce model staleness
Train and Deploy Cycle

Apache Airflow: DAGs
Hyperparameter Tuning
Lessons learned

**Automation is crucial**

ML models become stale over time
ML Hyperparameter tunings are often tedious
Overview

- ML Use Cases at Twitter
- ML Platform Requirements & Challenges at Twitter
- Unifying Twitter Around a Single ML Platform
- Technology Migrations
- **Health ML Use Case**
- Summary of Lessons Learned
- Future of Our ML Platform
Health ML Case Study

● Situation:
  ○ Models still running using Lua Torch
  ○ Retrained manually every ~6 months.

● Mission:
  ○ Migrate Health ML models to new ML Platform
  ○ Reach metric parity with existing models (minimum)
ML Pipeline Overview

Training Data

Data Exploration

Preprocessing

Feature Store

Training

Experiment Loop

Offline Evaluation

Model Tuning

Experiment

Online A/B Testing

Production

Prediction Servers
Lessons Learned

Teamwork: Platform, Modeling, Product Integration of All Components
Overview

• ML Use Cases at Twitter
• ML Platform Requirements & Challenges at Twitter
• Unifying Twitter Around a Single ML Platform
• Technology Migrations
• **Summary of Lessons Learned**
• Future of Our ML Platform
Summary of Lessons Learned

- **Consistency** brings efficiency
- **DL Reproducibility** is hard
- **Automation** is crucial
- **ML practitioner Diversity**
  - ML engineers vs DL researchers
  - Production vs exploration
- **Collaboration** of platform, modeling, product teams
Overview

• ML Use Cases at Twitter
• ML Platform Requirements & Challenges at Twitter
• Unifying Twitter Around a Single ML Platform
• Technology Migrations
• Summary of Lessons Learned

• Future of Our ML Platform
Future

2018 Strategy: **Consistency & Adoption**
2019 Strategy: **Ease of Use & Velocity**

- 10x, 50x training speed
- Auto model evaluation & validation
- Auto model deploy & auto scaling
- Auto hyperparameter tuning & architecture search
- Continuous Deep Learning Model Training
  and so on ...
Thank You

If you are interested in learning more about Twitter Cortex, please contact: @yz @strife076