How eBay Puts Big Data and Data Science to Work

Mike Mathieson

Sr. Director, Marketing Engineering and Trust Science

eBay, Inc.

A Bit About Me

- MSCS from UC Santa Cruz, focused on machine learning
- Early career: predicting drug interactions, identifying insider database intrusions, web search spelling corrections, web search spam/quality scoring
- Director of Search Relevance at Yahoo! (2003-2008)
- VP of Research at Searchme (2008-2009)
- Sr. Director of Search Science at eBay (2009-2011)
- Sr. Director of Marketing Engineering and Trust Science at eBay (2011-2013)

What Is Data Science?

- Extraction of information from data
- Can incorporate method and algorithms from diverse sub-areas of math, statistics, and computer science
  - Most commonly machine learning, natural language processing, information retrieval, graph theory
- Usually requires one or more specialists in the above areas
- In commercial settings, frequently used to find insights and analyze large data stores

Why is Data Science Important in Engineering Projects

- We don't always know what our customers want but we can usually learn by example
- Personalization, recommendation, targeting, optimization, prediction, and detection are becoming table stakes in many domains.
- Storage and optimization technologies have been commoditized and democratized, but good processes and information creation can still give a competitive edge

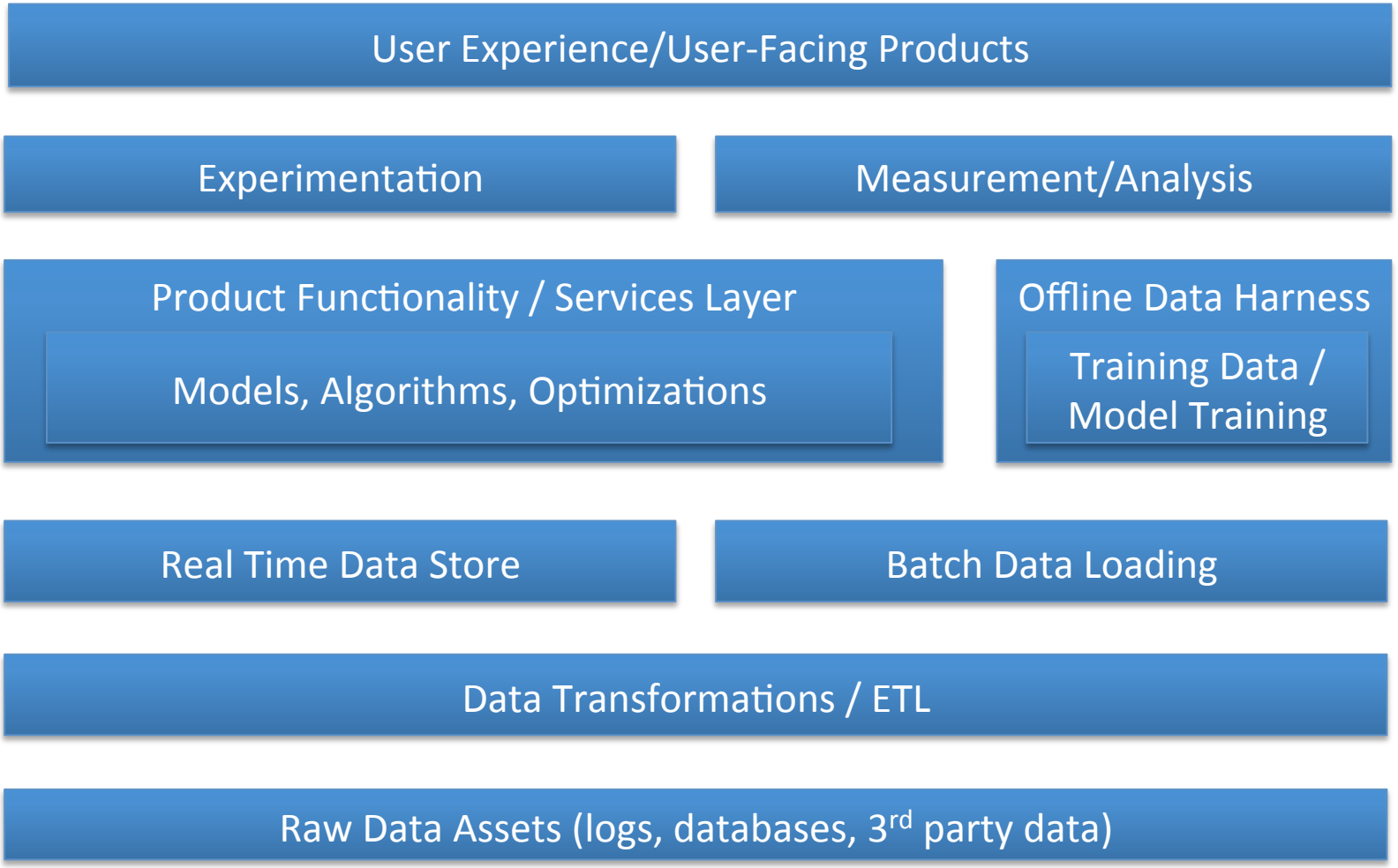
Challenges in Data Science

- Putting data to work in products is one of the generational challenges for engineers today
  - 5-10 years ago, it was about storing data and doing BI, today it's about personalization, targeting, or optimization
- Companies know their data is valuable, but don't know how to put it to use
  - Lack of experience and best practices make it easy for companies to get misled and on the wrong track
- Data is not the same thing as information.
- Non-deterministic results are hard to incorporate into engineering planning.
- Data science and Big Data have become buzzwords and are often treated as distinct from engineering.

Good/Bad Data Science

- Good Data Science…
  - Is iterative
  - Is scientifically controlled
  - Is informed by best practices
  - Is verifiable at increasing fidelity as it approaches deployment
- Bad Data Science…
  - Has long periods of silence
  - Changes too much all at once
  - Requires leaps of faith
  - Is not measurable until completion

Data Science Ecosystems



Flow of a Data Science Project

1. Find critical functionality that can be phrased as an optimization problem (eg. search ranking)
2. Implement a framework for A/B testing different versions (eg. split experience by user ID)
3. Implement metrics that can be run in each A/B segment and are indicative of product success (eg. avg. click position, revenue)
4. Identify a target to predict (eg. Probability of a click)
5. Collect easy to find features about the state (eg. query info, item info, historical click info)
6. Build a model to predict the target given the features.
7. A/B test it, triage to identify new opportunities. Launch if it improves core metrics identified in step 3.
8. Repeat steps 4 through 7 using different targets, features, data sizes, and modeling techniques.
9. When features stress infrastructure or it takes too long to iterate, re-factor or re-design relevant components.

Example #1: Search Ranking

- Given a set of results that are candidates to show a user in response to a search query, order those results to maximize user satisfaction.
- Can be turned into a regression problem by choosing a measurable proxy for user satisfaction (eg. probability of a click) and sorting by that probability to create the final rank ordering.

Early Approaches to Ranking

- Identification of strong attributes of query/item matches
  - TF/IDF
  - Historical click information
- Identification of deep factors of relevance
  - Authority (eg. PageRank)
  - Spam/Quality
- Combining information into a single heuristic function that attempts to balance the strengths of each feature
- When that fails for important queries, sometimes have human-designed result sets

- **Limitations:**
  - Most work goes into the weighting and optimization of the heuristic function.
  - Humans aren't good at designing complex tradeoff algorithms
  - Each new type of feature expands the difficulty of ranking

Machine Learned Ranking

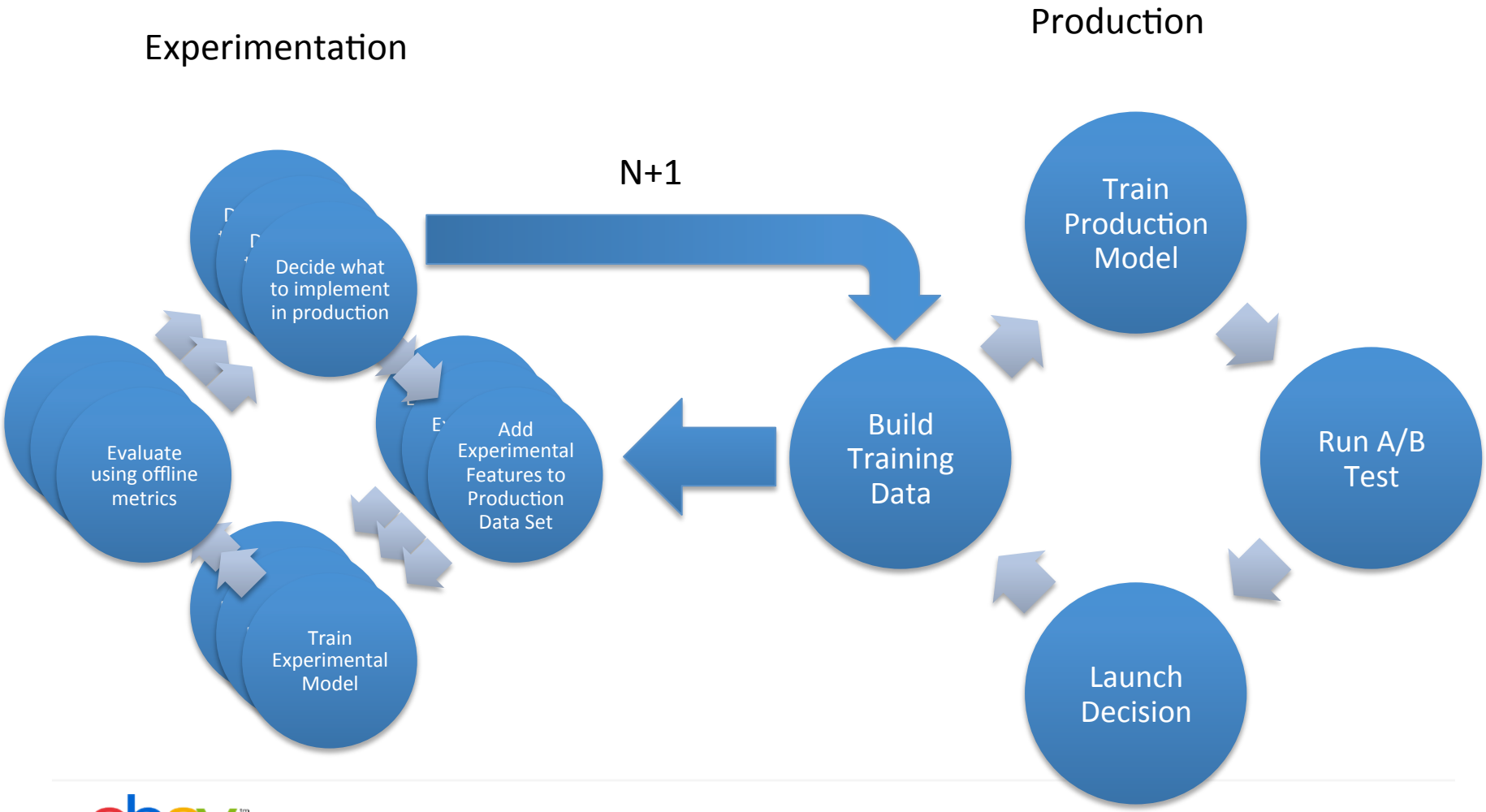
- Use thousands of weak features covering all aspects of queries and documents, in addition to strong features.
- Create a repeatable process for training a regression function that predicts a proxy for user satisfaction given a set of input features about a query and item

- **Benefits:**
  - Shifts focus to new feature creation
  - Allows new information to be incorporated quickly and effectively with minimal human involvement

Why does ML Ranking work?

- Exposing new factors is the primary way to achieve relevance improvements
  - Pace of factor development is critical
  - Fast iteration is the top priority, fast deployability is a close second
- Ideal ranking requires complex factor combination beyond human abilities
  - Some factors are non-linearly related to relevance
  - Factor combination only tractable for humans when looking at single queries
  - Systematic patterns across large sets of queries cannot be visualized by humans
- Factor development can be highly parallelized - asynchronous
  - Most factor research projects need 1-2 people
- Factor combination is a bottleneck
  - New factors and training data change the ideal function
  - Do not want this process gated on humans

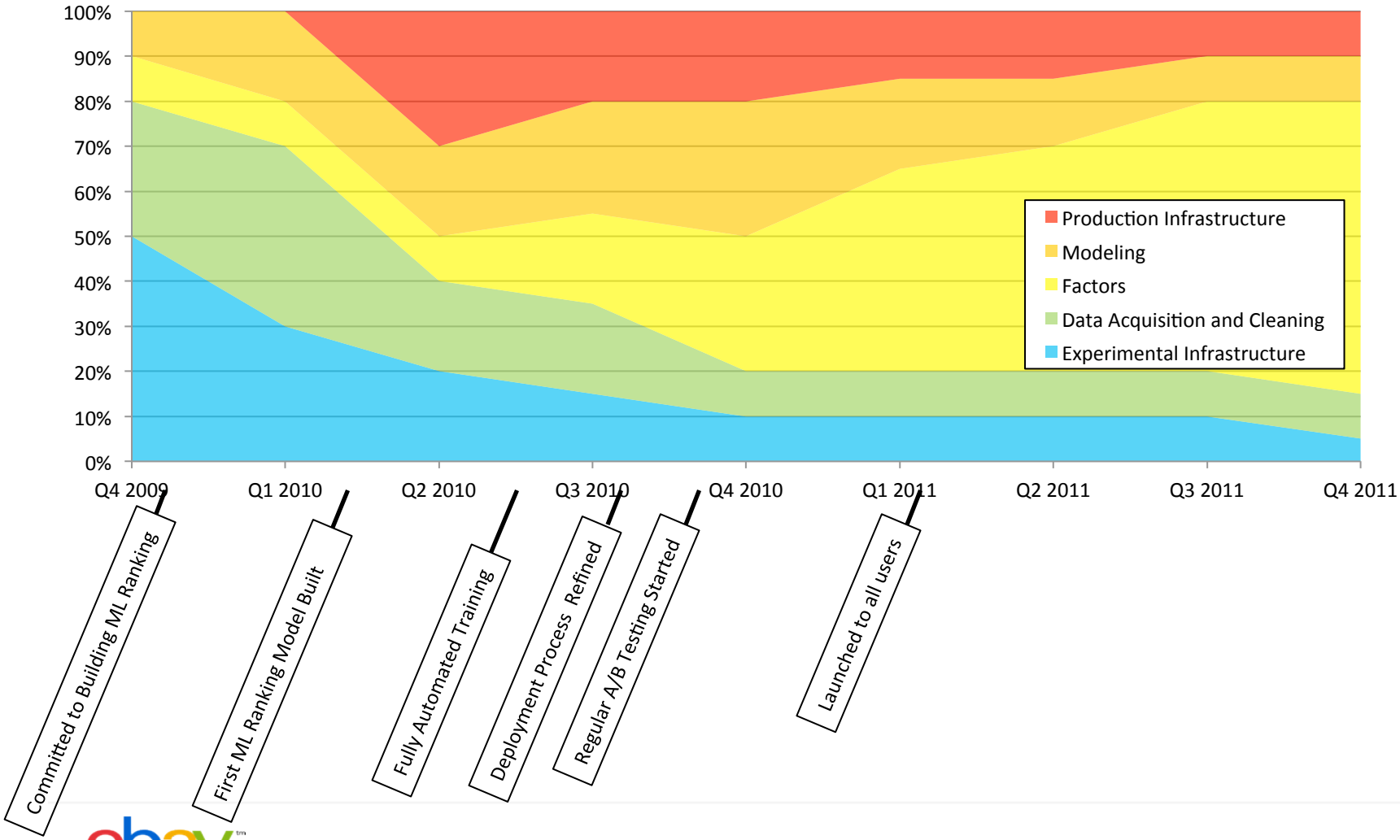
Experimentation/Production Cycles



Project Progression

- Front-loaded build-out of A/B testing and log data processing/cleaning
- Started with simple target: P(click)
- Started with simple, easy to get features
- Added abstraction and modularity to production feature code
- Refined target to include revenue
- Focused on new feature experiments and deployments

Resource Investment Over Time – ML Ranking



Example #2: Paid Search Optimization

- eBay is one of the largest keyword and PLA bidders
- Major traffic driver to bring users to eBay
- eBay (and other marketers) set a bid on how much they will pay for a click on a paid search ad. Google determines the winning bid and places ads on search pages.
- Historically, bids were manually assigned by paid search experts at eBay.
- However, rapid changes in budget level and low efficiency led us to explore automated bidding

Early Approaches to Paid Search Optimization

- Manual bid setting based on average return and “strategic” spend
- Simple bucketing of keywords based on average ROI, and setting bids based on a multiplier on that ROI
- Focusing on “head terms” with high volume and using historical data.

A new approach to Paid Search Optimization

- Grouping similar keywords based on co-purchases, text similarity, and ROI characteristics
- Establishing a predictive model of user behavior to establish baseline purchase expectations
- Train a model targeting iROI by looking at revenue and subtracting out expected purchases

Project Progression

- Front-loaded build-out of A/B testing and log data processing/cleaning
- Started with simple target: E(ROI)
- Started with simple, easy to get features
- Added abstraction and modularity to production feature code
- First target failed! Realized we needed to change our target to include incrementally.
- Focused on new feature experiments and deployments

Conclusions

- Complexity comes in different forms
  - Search: tight SLAs and huge data scale
  - Paid Search Optimization: understanding what users would have purchased anyway and only giving credit for incremental activity
  - E-Mail: personalizing by understanding each user's unique needs
  - Trust: "arms race" with malicious actors
- However similar templates can be applied successfully

ebay

Best Practices for Success

- Model speed
  - Needs to be fast enough for run time, rich enough for problem complexity
- Training data generation
  - Needs to be reliable
  - Define a regular cadence
- Experimentation
  - Speed!
  - Don't wait for new training data – join to "last known good"
- Deployment and model organization
  - Prepare for hundreds of models
  - Try to deploy as data, not code
- Debugging tools
  - Even if you can't understand the models, try to understand the effects

Team Dynamics

- Tensions are natural in hybrid teams
- Hire and develop people who will compromise and work towards mutual understanding
- Beware dogma vs dogma – make questions empirical
- Bake-offs should be short lived. Debate, explore, decide, deliver
- Encourage scientists to write code
- Define metrics around time spent on various activities to ensure efficiency
- Celebrate failed tests—as long as you learned something

How to Make Science Work in an Engineering Team

- Scientists should be part of an engineering team, even if that engineering team is solely responsible for services based on the science.
- Training and data systems must be treated as first-tier products with both engineers and scientists involved.
- Where possible, don't simulate training data, dump it from live production.
- A deployment rhythm must be established.
- "Science" happens asynchronously (but tracked as part of project).
- Once results are promising enough, "dev" tasks are spawned that will be deployed in a later release.

How can an Engineer support Data Science?

- Be curious
- Call out manual steps and bottlenecks
- Make sure science is part of engineering processes, not exempt from it
- Add a configuration-based feature calculation layer
- Add a configuration-based model scoring layer
- Take an interest in data and the data pipeline
- Spend a couple days pair programming with a scientist

Question Time!

Thank you!