Deep Learning-based Search and Recommendation systems using TensorFlow

Abhishek Kumar
Dr. Vijay Agneeswaran

MARCH 06, 2018

Strata Conference – San Jose (2018)
Session Logistics

1. Access the work environment using the following link [ ]


3. Connecting to the speakers [Please send introductory note in LinkedIn invite]

   1. Abhishek Kumar (http://bit.ly/kumarabhishek @meabhishekkumar)
   2. Dr. Vijay Agneeswaran (http://bit.ly/vijaysa @a_vijaysrinivas)

4. Don’t forget to tweet #stratadata
About the Speaker

ABHISHEK KUMAR

Senior Data Scientist, SapientRazorfish
Masters from University of California, Berkeley
Pluralsight Author

• Doing Data Science with Python
• R Programming Fundamentals
• Machine Learning with ENCOG
• Currently authoring: “Deploying Machine Learning Models with Tensorflow Serving”
About the Speaker

DR. VIJAY AGNEESWARAN

Senior Director and Head of Data Science, SapientRazorfish
MS (Research) & PhD, IIT Madras
Post doctoral research fellowship, LSIR Labs
Professional member: ACM, IEEE (Senior)
4 Full US Patents and multiple publications (including IEEE journals)
Regular Speaker @ O’Reilly Strata conference
Audience Profiling

1. Machine Learning?
2. Deep Learning?
3. Search and Recommendation Systems?
4. Tensorflow?
# Session Agenda

## 4 Levels of Learning

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5. Demo: Build DL based RecSys with Implicit Feedback using TF</td>
<td></td>
</tr>
</tbody>
</table>

**BREAK**
By the end of this session...

1. You will have basic foundation of Deep Learning.
2. You will have good understanding of Recommendation Systems, Search and Ranking Systems
3. You will be able to transform the concepts and build DL models using Tensorflow
   1. Deep learning based Image retrieval system
   2. Deep learning based hybrid RecSys on explicit feedback
   3. Deep learning based RecSys and Learning to Rank model on implicit feedback
4. You will have high level idea to take the lab scale solution to a production ready system
Quick Chat with Your Neighbor

1. Introduce yourself to your neighbor
2. What they are looking to learn from the tutorial
Problem Space: Search and Recommendation
We now live in the connected age

<table>
<thead>
<tr>
<th>Year</th>
<th>Economic Era</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>Manufacturing Economy</td>
<td>Mass manufacturing makes industrial powerhouses successful. “A customer can have a car painted any color he wants as long as it's black.”</td>
</tr>
<tr>
<td>1960</td>
<td>Distribution Economy</td>
<td>Global connections and transportation systems make distribution key. “Strategy is...globalization, taking your products around the world; be the low-cost producer.”</td>
</tr>
<tr>
<td>1990</td>
<td>Information Economy</td>
<td>Connected PCs and supply chains mean those that control information flow dominate. “The great challenge...is to make productive the tremendous new resource, the knowledge worker.”</td>
</tr>
<tr>
<td>2008+</td>
<td>Connected Economy</td>
<td>iPhone and Facebook launch in 2007/8 heralding a new era in transparency, empowerment, and experimentation. “The customer is the center of your universe.”</td>
</tr>
</tbody>
</table>

*Adapted from Forrester Research “Age of the Customer” graphic.*
The Connected Consumer is in charge

Empowered and demand transparency

Value experiences over most things

Embrace and seek new companies to engage with

Demand personalization and real-time relevancy
Research proves that consumer experience does matter

86% CUSTOMERS SAID THAT PERSONALIZATION HAS HAD SOME IMPACT ON PURCHASING DECISION
75% SHOPPING TIME IS SPENT ON PRODUCT DISCOVERY & RESEARCH ONLINE BY 50% CUSTOMERS
81% CUSTOMERS DEMAND IMPROVED RESPONSE TIME
95% DATA WITHIN ORGANIZATION REMAINS UNTAPPED

Problem Space: Search

Search Engines

Search term

Filtered Results

Re-Ranked Results

Personalized Search

User + Interactions

Engine

Indexing

Similarity Calculation

Challenges

- How to represent text, images, audios
  - TF-IDFs?
  - Metadata for binary?

- Search in other languages?

- Search Quality
  - Well-ranked results
  - By providing better search results, Netflix estimates that it is avoiding canceled subscriptions that would reduce its revenue by $1B annually. [Link]
Problem Space: Recommendation

Challenges

- How to represent users and items?
- How to build hybrid systems with both interactions (collaborative) and user/item metadata?
- How to use dynamic user behaviors?
- How to use implicit (view, share) feedback?

Recommendation Engines

- Re-Ranked Results
- Recommended Results
- Results Diversity
- Recency
- Impression Discounting
- Engine
- Item
- Interactions
- Other Users Interactions
- User
Why Deep Learning for Search and Recommender System?

- Direct content Feature extraction instead of metadata
  - Text, Image, Audio

- Better representation of users and items for Recsys

- Hybrid algorithms and heterogeneous data can be used

- Better suited to model dynamic behavioral patterns and complex feature interactions
Deep Learning Primer
What is Deep Learning?

Class of machine learning algorithms

- That uses hierarchy of non-linear processing layers and complex model structures
- Layers learn to represent different representation of data
- Higher level features are constructed from lower level abstract features
- Trendy name for “Neural Networks with deep layers”
Simple Neural Network With 2 Layers

Limitation: Can learn only linear relationship
Simple Neural Network with At Least One Hidden Layer

Universal Approximator
Neural Network Training: Backpropagation

- **Input Layer**
- **Hidden Layer**
- **Output Layer**

Output Calculation

Error Propagation and Weights Update (using Gradients)

Error Calculation
What Changed Now?

- **More data**
  - More complex models need more data to avoid overfitting
  - Deep learning models have higher VC dimension

- **Computing Power**
  - Computing power have increased significantly
  - Specialized hardware such as GPUs and TPUs

- **Research Breakthrough**
  - Hinton’s work on layerwise training led a new paradigm to train deep networks
  - Non-saturating activation functions (variation of ReLUs)
  - Dropouts helped to achieve regularization easily
  - Adaptive learning rate helped to avoid problems of local minima and led to better convergence
Popular Neural Network Architectures: Deep Feedforward
Popular Neural Network Architectures: Convolution Neural Network (CovNet)

Image Credit: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
Convolution Neural Network (CovNet): Components

CONVOLUTION
Mathematical Operation on two sets of information

NON-LINEARITY

POOLING

Input

Filter / Kernel

Feature Map

Mathematical Operation on two sets of information

Image Credit: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
Convolution Neural Network (CovNet): Components

ReLU (Rectified Linear Unit)

- Capture Interaction
  - E.g Input: 3 * x1 + 4 * x2, Output: f(Input)

- Introduce Non-Linearity
  - Slope is not constant (zero for negative value, 1 for positive)

- Reduce the chances of vanishing gradient
  - Average derivative rarely become 0 (some data points have positive derivative)

Image Credit: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
Convolution Neural Network (CovNet) : Components

Max Pool

CONVOLUTION
NON-LINEARITY
POOLING

Downsample feature map to reduce dimensionality

Image Credit: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
Popular Neural Network Architectures: LSTM (Long Short Term Memory)

- Special kind of Recurrent Neural Network (RNN)
- Can learn long-term dependencies (as default behavior)
- Use gates
  - Forget gate
  - Input gate
  - Output gate

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Popular Neural Network Architectures: RBM (Restricted Boltzmann Machine)

- Shallow network
- Can be used for unsupervised learning
- Reconstruct input
- Belongs to auto-encoder family
- Useful in Collaborative filtering, dimensionality reduction
Popular Neural Network Architectures: Deep Belief Networks

- Boltzmann Machine is a specific energy model with linear energy function.
- This is a deep neural network composed of multiple layers of latent variables (hidden units or feature detectors)
- Can be viewed as a stack of RBMs
- Hinton along with his student proposed that these networks can be trained greedily one layer at a time
Popular Neural Network Architectures : Auto-Encoders

- Aim of auto encoders network is to learn a compressed representation for set of data
- Unsupervised learning algorithm that applies back propagation, setting the target values equal to inputs (identity function)
- Denoising auto encoder addresses identity function by randomly corrupting input that the auto encoder must then reconstruct or denoise
- Best applied when there is structure in the data
- Applications : Dimensionality reduction, feature selection

Source : http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Why Tensorflow for Deep Learning?
Why Tensorflow for Deep Learning?

- Extensive built-in support
- Assemble neural networks
- Mathematical functions
- Auto-differentiation & first-rate optimizers
- Versatility
Why Tensorflow for Deep Learning?

Align cognitive model to programming model
Why Tensorflow for Deep Learning?

Use Tensorboard to Visualize and Debug Deep Learning Network

Network Graph

Cost Trends

Visualize Embedding
Why Tensorflow for Deep Learning?

Tensorflow Playbook: http://playground.tensorflow.org/
Deep Learning in Search
Representation: A Key Aspect

Search Engines

Search term

User + Interactions

Personalized Search

Filtered Results

Re-Ranked Results

Search or Query

Word
Phrase
Sentence
Image
Audio

Collection

Set of Documents
Set of Documents
Set of Images
Set of Audio

Representation

Similarity
Representation: A Key Challenge

Treat “Representation” as the problem of “Embedding” to encode objects (text, images) into continuous space (set of numeric values).
Search Problem In Context of Embedding

1. Create embedding for objects into continuous space

   Objects

   - Sneakers
   - Shoes

   How to tie laces?

2. Put similar objects together based on embedding

   - Black holes
   - How time changes in black hole?

3. Given a query embedding, find neighbors quickly

   - Black holes
   - How time changes in black hole?
**Word Embedding : One-Hot Encoding**

<table>
<thead>
<tr>
<th></th>
<th>shoe</th>
<th>sneakers</th>
<th>tree</th>
<th>book</th>
<th>black</th>
<th>...</th>
<th>..</th>
<th>..</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sneakers</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shoe</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tree</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Number of columns = Number of unique words in the vocabulary

**Issues :**

- Sparse ( all values are zero except one )
- Large embedding dimension ( equal to vocab size )
- Semantic meaning not captured
  - “shoe” is at same distance as “tree”
Word Embedding: Prediction Based Encoding

**Sneakers**  
[ 0.322 0.122 0.231 0.111 0.222 ....... 0.445 ]

Embedding Size: d

**Benefits:**
- Dense representation
- Smaller embedding dimension (equal to embedding size: d)
- Semantic meaning captured
  - “shoe” is at smaller distance than “tree”

---

Word2Vec Model  
(Google, 2013)

CBOV  
Use surrounding words to predict target word

Skip-Gram  
Use target word to predict surrounding words

GloVe Model  
(Stanford, 2014)

Use word-word co-occurrence matrix and nearest neighbor to create embedding
**Demo**: Short Introduction to Embedding  
**Goal**:  
- Embedding in Tensorflow  
- Word Embedding using GloVe pre-trained model
I like white sneakers.

<table>
<thead>
<tr>
<th>I</th>
<th>Like</th>
<th>White</th>
<th>Sneakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>x1</td>
<td>x2</td>
<td>x3</td>
<td>x4</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

- Sentence embedding can be considered as word embedding matrix where each word is represented as a column of size $d$.
- Leverage pre-trained word embedding to learn sentence embedding.
- Weights are further tuned during training process.
Sentence Embedding Using Convolution Neural Network

Paper: Convolutional Neural Networks for Sentence Classification" by Yoon Kim [Link]
Image Embedding: Option 1: Flattened Arrays

Each image is set of pixels
Flatten the pixels matrix into arrays

100 px by 100 px image: 10000 dimensional array

Issues:
- Very large embedding dimension
- Search in a very large embedding space will be very expensive
- Spatial features (edges, contours, textures) are not captured: Poor search results
Image Embedding: Option 2: Pre-Trained Deep Learning Models

Benefits:
- Smaller embedding dimension
- Spatial features (edges, contours, textures) are captured in intermediate layers
- Enhanced search experience

Image Source: https://www.saagie.com/blog/object-detection-part1
Demo: Image search using Alexnet Pre-Trained Model

Goal:
- Use Alexnet Pre-trained model to create image embedding
- Image Search
Deep Learning in Recommendation System
RecSys 101: What is RecSys?

“Serve the relevant items to users in an automated fashion to optimize short and long term business objectives”

<table>
<thead>
<tr>
<th>RELEVANT (WHAT)</th>
<th>AUTOMATED (HOW)</th>
<th>BUSINESS OBJECTIVES (WHY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Novelty</td>
<td>1. No manual intervention</td>
<td>1. Short Term Business Objectives</td>
</tr>
<tr>
<td>2. Serendipity</td>
<td>2. Scale Up</td>
<td>a. High clicks</td>
</tr>
<tr>
<td>3. Diversity</td>
<td></td>
<td>b. Revenue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c. Positive explicit ratings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Long Term Business Objectives</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Increased engagement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Increase in social action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c. Increase in Subscriptions</td>
</tr>
</tbody>
</table>
RecSys 101: Internals

- **What**
  - Item filtering, Understanding
- **Who**
  - User Profile, Intent
- **Context**
  - Interaction
- **Machine Learning**
  - Rank Items
    - Sort by Business Objective
  - Score Items
    - P (click), P (share)
RecSys 101: Content Based Recommendation

Recommends an item to a user based upon a description of the item and a profile of the user’s interests

Representing Items using Features

<table>
<thead>
<tr>
<th>Drama</th>
<th>Arty</th>
<th>Comedy</th>
<th>Action</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>Commercial</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0.8</td>
</tr>
</tbody>
</table>

Creating a user profile that describes the types of items the user likes/dislikes
RecSys 101: Content Based Recommendation

- More than 100 million monthly active users
- Over 30 million songs

Track: May 16
Artist: Lagawagon
Album: Let’s Talk About Feelings
Release: 1998
RecSys 101: Content Based Recommendation

**Pros**

- No need of other users data
- Easy to understand reason behind recommendation
- Capable of recommending new and unknown items

**Cons**

- Can only be effective in limited circumstances
- No suitable suggestions if content doesn’t have enough information
- Depend entirely on previous selected items and therefore cannot make predictions about future interests of users
RecSys 101: Internals

What

Item filtering, Understanding

Who

User Profile, Intent

Context

Collaborative

Interaction

Score Items
P (click), P (share)

Rank Items
Sort by Business Objective

Machine Learning
RecSys 101: Collaborative Filtering

Unlike Content based filtering, Collaborative Filtering doesn’t require any product description at all.

Which movie should I watch?

Which Restaurant to go?

Which book to read next?

I should ask my “FRIENDS”!
RecSys 101: Collaborative Filtering: Interactions / Feedback

Explicit

Ratings

Implicit

Purchased
Add to cart
Viewed
Shared
RecSys 101: Collaborative Filtering: Interactions / Feedback

**Explicit**
- Very few users leave ratings
- Very less explicit data
- Ratings are biased
- Often not easy for user to express likeness in terms of Ratings or score

---

**Implicit**
- Easy to track & Store web logs data
- Lots of implicit data generated for each user
- More the data, better the recommendations
- Noisy
- Difficult to infer Negative Feedback
RecSys 101: Collaborative Filtering

Collaborative Filtering

- Nearest Neighbor
- Latent Factor

User based

Find similar users like me and recommend what they liked

Item based

Find similar items to those I have previously liked

Factor based techniques (Matrix Factorization, Factorization Machine)

- Scalability
- Predictive accuracy
- Can model real-life situations (e.g. Biases, Additional Input sources, Temporal Dynamics)

$1 Million Netflix Challenge
RecSys 101 : Collaborative Filtering : Latent Factor

Take the users and their feedback for different items and identify hidden factors that influence the user feedback.

The idea is to factorize or decompose the user item matrix into two matrices:

- Users are mapped on to hidden factors
- Items are mapped on to hidden factors
RecSys 101: Collaborative Filtering

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content information not required either of users or items</td>
<td>Cannot produce recommendations if there is no interaction data available (Cold Start Problem)</td>
</tr>
<tr>
<td>Personalized recommendations using other user’s experience</td>
<td>Often demonstrate poor accuracy when there is little data about users’ ratings (Sparsity)</td>
</tr>
<tr>
<td>No domain experience required</td>
<td>Popular items get more feedback (Popularity bias)</td>
</tr>
</tbody>
</table>
RecSys 101: Internals

What
- Item filtering, Understanding

Who
- User Profile, Intent
- Context

Rank Items
- Sort by Business Objective

Score Items
- P (click), P (share)

Interaction

Machine Learning

Hybrid
RecSys 101 : Hybrid Recommendation Engine

**Pros**

- Solve the issue of Cold Start by leverage both content and collaboration
- Use of Implicit feedback reduces the sparsity issues to a large extent
- Can include higher order feature interactions as well

**Cons**

- Difficult to implement
Representation: A Key Aspect

Recommendation Engines

- User
- Item
- Engine
- Interactions
- Other Users
- Interactions

User ID
Item ID
User / Item Metadata

Re-Ranked Results
Recommended Results

SAPIENTRAZORFISH
Matrix Factorization

How to better represent users and items?

What about item and user metadata?
Matrix Factorization

\[
\begin{align*}
R & \quad U \\
X & \quad V
\end{align*}
\]

\[
\text{user} \quad \begin{array}{c}
\mathbf{u}_i^{T} \cdot \mathbf{v}_j \\
\mathbf{u}_i \\
\mathbf{v}_j
\end{array}
\]

\[
||\mathbf{u}_i^{T} \cdot \mathbf{v}_j - y||^2 + \lambda(||\mathbf{V}||^2 + ||\mathbf{U}||^2)
\]

Credit: Olivier Grisel
**Demo**: User and Item Embedding in Matrix Factorization using Tensorflow

**Goal**:
- Use embedding in RecSys
- How to add metadata for building hybrid RecSys
Matrix Factorization: Deep Neural Networks

\[ ||f(u_i, v_j) - y||_2^2 + \lambda(||V||_2^2 + ||U||_2^2) \]

Credit: Olivier Grisel
Matrix Factorization: Deep Neural Networks with Metadata

Credit: Olivier Grisel
Learning to Rank
Problem Space: Re-Ranking

Search Engines

Search term:

Search Engines

Filtered Results

User + Interactions

Indexing

Similarity Calculation

Personalized Search

Re-Ranked Results

Recency

Impression Discounting

Search +

Re-Ranking Using Machine Learning = Learning to Rank
Why Learning to Rank is Required?

Classical Information Retrieval
- Inputs
  - Query, Document
    - TF, IDF, |D|, P(t|D)
- Output
  - Relevance Score (q, d)
- Manual function: VSM Cosine, BM25, LM Dirichlet
- Not many factors to tune

Learning to Rank
- To improve search results quality need to consider many features
  - Query word in anchor text
  - Number of images
  - Number of out links
  - Page rank
- Classical IR don’t work as many factors and their combinations have to be tuned
- Machine learning based ranking system learn a function to automatically rank results
Learning to Rank Formulation

Query $q_i$

Documents $D_i = d_{i1}, d_{i2}, d_{i3}, ..., d_{in}$

Relevance $[0, 1]$ $y_{i1}, y_{i2}, y_{i3}, ..., y_{in}$

Learn a ranking function $H$

$H(w, \text{func}(q_i, D_i)) \rightarrow \text{Optimal Ranking } R_i$

Query + Documents Representation

Re-ranked Documents based on predicted relevance
Learning to Rank Techniques: Point Wise Approach

Query \( q_i \)

Documents \( D_i = d_{i1}, d_{i2}, d_{i3}, \ldots, d_{in} \)

Relevance \([0, 1]\) \( y_{i1}, y_{i2}, y_{i3}, \ldots, y_{in} \)

\( H(w, \text{func}(q_i, D_i)) \rightarrow \text{Optimal Ranking} \ R_i \)

Binary Classification

Use Probability as Relevance Score

Triples

Input (Query, Document Pair Representation)

Output Class
Learning to Rank Techniques: Pair Wise Approach

Query \( q_i \)

Documents \( D_i = d_{i1}, d_{i2}, d_{i3}, \ldots, d_{in} \)

Relevance \([0, 1]\) \( y_{i1}, y_{i2}, y_{i3}, \ldots, y_{in} \)

There is also a “List Wise Approach” that ranks all documents in one go. But complicated to implement.

\[ H(w, \text{func}(q_i, D_{i\text{pos}})) \geq H(w, \text{func}(q_i, D_{i\text{neg}})) + \text{margin} \]
Learning to Rank In Recommendation System

Query $q_i$

Documents $D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$

Relevance $[0, 1]$ $y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

$H(w, \text{func}(q_i, D_i)) \rightarrow$ Optimal Ranking $R_i$

User $Q_i$

Items $D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$

Implicit Feedback $[0, 1]$ $y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

$H(w, \text{func}(q_i, D_i)) \rightarrow$ Optimal Ranking $R_i$
Triplet Loss with Implicit Feedback

\[ u_i^T \cdot v_j \]

\[ u_i^T \cdot v_k \]

\[ \max(0, u_i^T v_k - u_i^T v_j + \alpha) \]
Deep Triplet Network with Implicit Feedback

Credit: Olivier Grisel
Demo: Using Triplet Loss for Implicit Feedback using Tensorflow

Goal:
- Use implicit feedback
- Use triplet network
Popular Alternatives

**Deep FM**

![Deep FM Diagram]

**Wide & Deep (Google)**

![Wide & Deep Diagram]

Paper: DeepFM: A Factorization-Machine based Neural Network for CTR Prediction (Link)
Production
How Recommendation System Works

1. Enter
2. Interact (click / buy / rate)
3. Server Request
4. Run Recommendation Engine
5. Run Recommendation
6. Personalized Recommendation
7. Response with Personalized Recommendation
8. Save Interactions
9. Pass to the model
10. Updated Model
11. Train Model

Recommendation System (Run in Subseconds)
Machine Intelligence Architecture

1. Data Ingestion and Processing Pipeline
   - Web Trackers
   - Mobile Trackers
   - Adobe/Google Analytics
   - 3rd Party Service (like Comscore)

2. Model & Testing Framework
   - Content & Attribute Based Models
   - Optimization Models
   - Deep Learning Models
   - Model Deployment Engine (Spark, Google Cloud ML or Azure ML)
   - Metrics
   - Model Ensembles
   - Tuning
   - A/B Test
   - Offline Evaluation
   - Real time adjustment with bandit algorithm

3. Machine Learning Server
   - Learning Server (Serving, TensorFlow
   - Machine Learning Server (Spark, Azure ML, Spark Velox)
   - Scheduled retraining
   - Application (Recommender Decoupled)
   - Gated Default Recommendation
   - Too long load time
   - Metrics

Enhanced Domain Data

Real-time Pipeline

Batch Pipeline

Hadoop Distributed File System

APIs

Kafka

Barnes Storage

API/Cloud Storage

Sqoop

APIs

Cached Default Recommendation

Two long load time

Application (Recommender Decoupled)
Tensorflow in Production

**Training**
- **Distributed training**
  - Distributed Tensorflow
  - Leverage Kubernetes to auto-scale training process
  - Tensorflow + Spark to get the best of both world
  - GCP Cloud ML for serverless processing
- **HyperParameter Tuning**
  - Kubernetes for parallel execution of hyperparameter tuning
  - Leverage Bayesian Optimization (Scikit-Optimize)
- **Specialized Hardware**
  - Leverage GPUs, TPUs for faster training
  - Leverage Estimator and Experiment APIs

**Inference**
- **Model Object Optimization**
  - Graph Transformation Tool (GTT) to remove unreachable nodes, fuse adjacent operators, round weights for compression
  - Ahead of Time (AOT) compiler built on XLA (Accelerated Linear Algebra) for minimal Tensorflow Runtime
- **Low latency inference**
  - Tensorflow Serving to serve TF models
Tensorflow Serving
Tensorflow Serving

- Low latency inference
- Model versioning and rollback
- Custom version policy for A/B and Bandit tests
- Uses highly efficient gRPC and Protocol Buffers
Next Steps

- Provide feedback on the tutorial
- Download & review tutorial material
  - Concepts
  - Demos
- Share
  - Progress, Issues, Use-cases
  - Twitter
    - @a_vijaysrinivas
    - @meabhishekkumar