

Netflix's Recommendation ML Pipeline using Apache Spark

DB Tsai

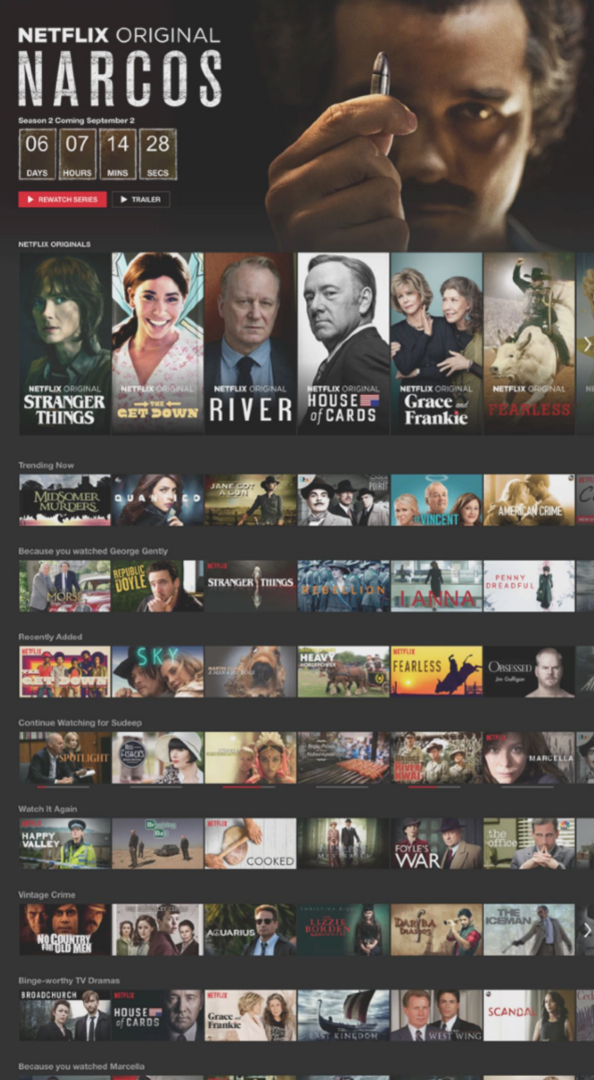
Spark Summit East - Feb 8, 2017

NETFLIX

At Netflix, we use ML everywhere

NETFLIX

Everything is a Recommendation



Over 80% of what members watch comes from our recommendations

Recommendations are driven by **Machine Learning Algorithms**

NETFLIX

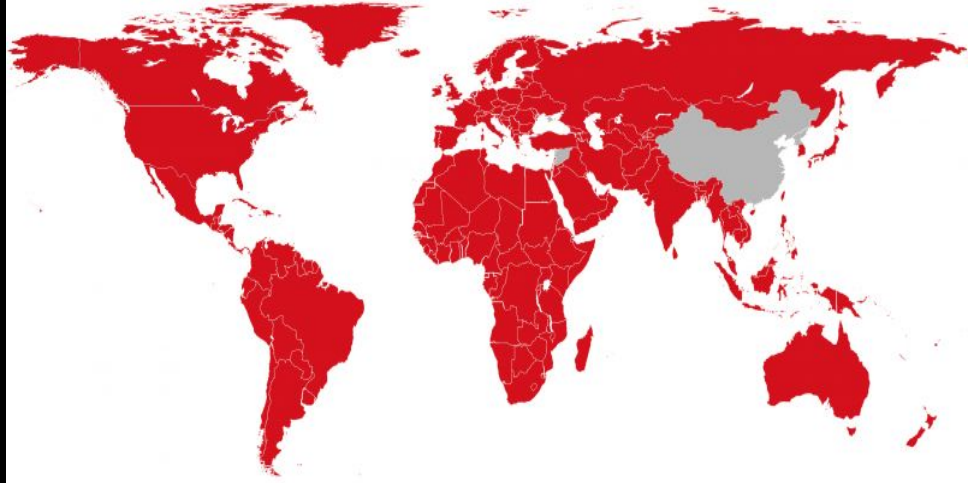
A person is standing on a stage, gesturing with their hands. Behind them is a large screen displaying the text '#netflixeverywhere' in a bold, sans-serif font. The word 'netflix' is red, and 'everywhere' is white. The background of the screen is a dark world map composed of many small, light-colored squares. The stage is dimly lit, and the audience is visible in the foreground as silhouettes.

#netflixeverywhere

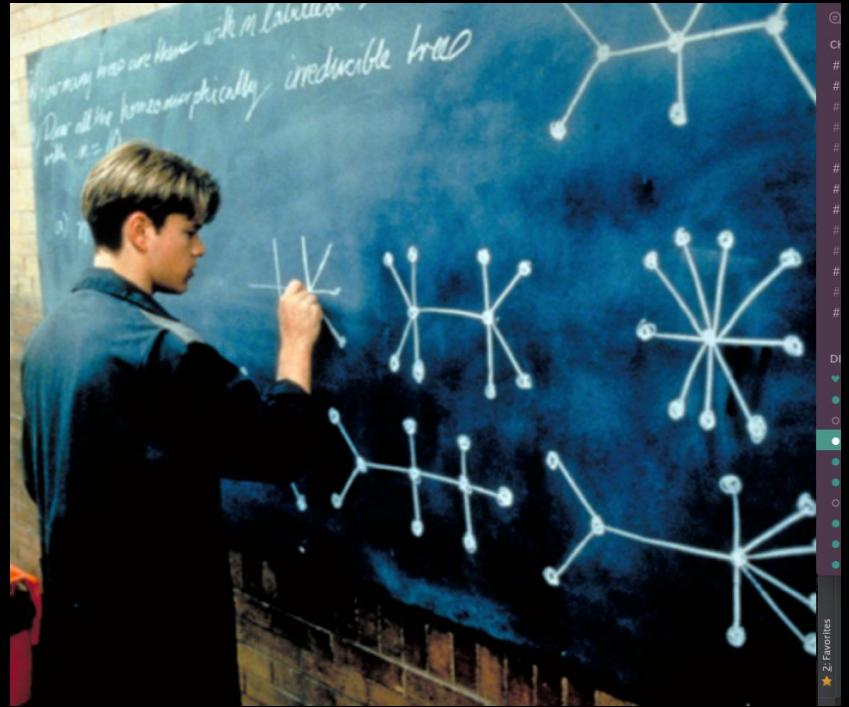
Jan 6th, 2016

#NetflixEverywhere

- 93+ Million Members
- 190+ Countries
- 125+ Million streaming hours / day
- 1000 hours of Original content in 2017
- 1/3 of US internet traffic during evenings



Constantly Innovating through A/B tests



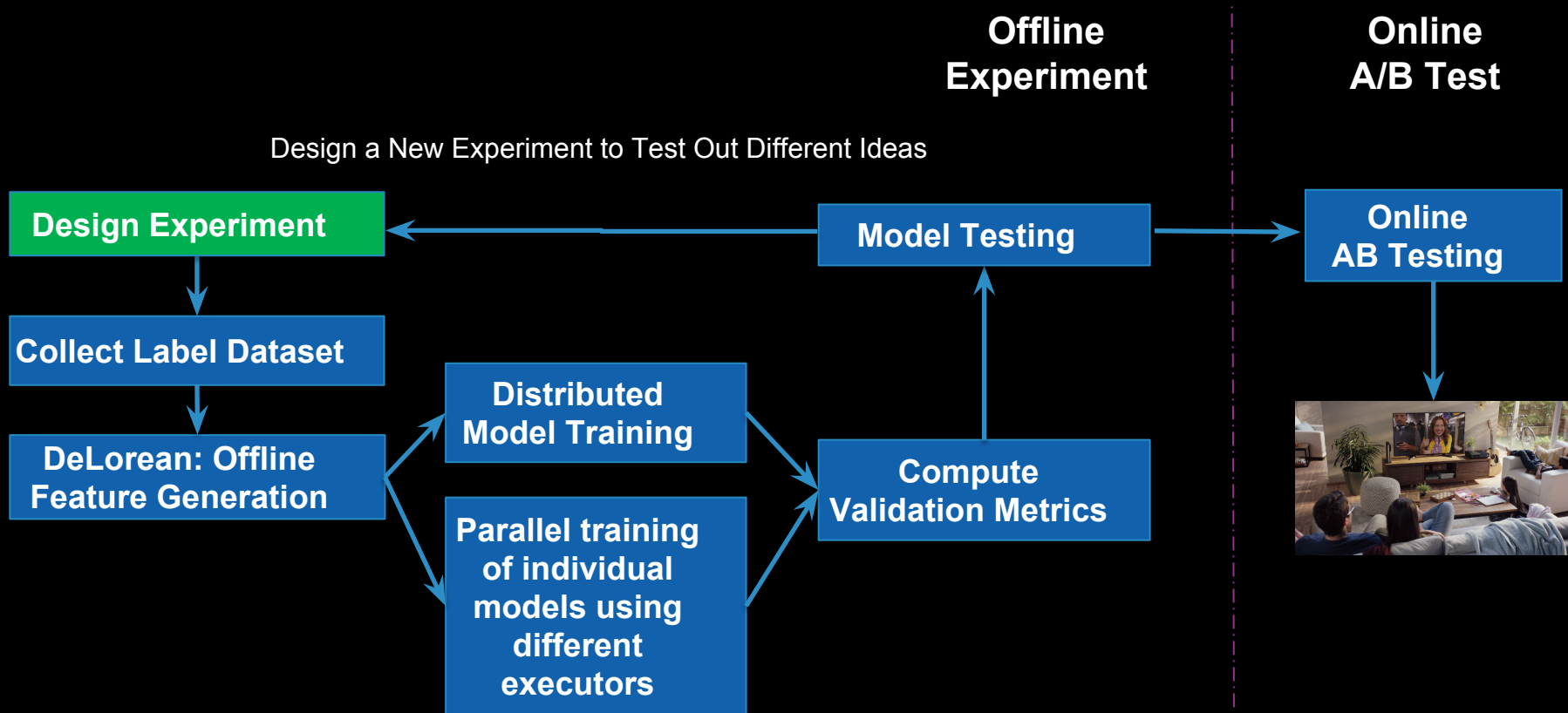
Data Driven

Try an idea offline using historical data to see if they would have made better recommendations

If it does, deploy a live A/B test to see if it performs well in Production

NETFLIX

Running an Experiment



We use a standardized data format across multiple ranking pipelines

This standardized data format is used by common tooling, libraries, and algorithms

NETFLIX

Ranking problems

Contexts: The setting for evaluating a set of items (e.g. tuples of member profiles, country, time, device, etc.)

Items: The elements to be trained on, scored, and/or ranked (e.g. videos, rows, search entities)

Labels: For supervised learning, this will be the label (target) for each item

NETFLIX

DeLorean Data Format a.k.a DMC-12

root

-- profile_id: long (nullable = false)

-- country_iso_code: string (nullable = false)

-- items: array (nullable = true)

| -- element: struct (containsNull = false)

| | -- show_title_id: long (nullable = false)

| | -- label: double (nullable = false)

| | -- weight: double (nullable = false)

| | -- features: struct (nullable = false)

| | | -- feature1: double (nullable = false)

| | | -- feature2: double (nullable = false)

| | | -- feature3: double (nullable = false)

NETFLIX

The nested data structure avoids an expensive shuffle when ranking

The features are derived from Netflix data or the output of other trained models

The features are persisted in HIVE using Parquet

Ensemble methods are used to build rankers

NETFLIX

Transformer



[https://en.wikipedia.org/wiki/Spark_\(Transformers\)](https://en.wikipedia.org/wiki/Spark_(Transformers)) **NETFLIX**

Transformer takes an input DataFrame and “lazily” returns an output DataFrame

Item Transformer

- **Extends Spark ML's Transformer**
- **Accepts DMC-12 DataFrame with contextual information**
- **Transforms DataFrame at the item level**

NETFLIX

Why DataFrame?

Catalyst Optimizations

Up-front Schema Verification

**We found a 4x speedup during feature generation
by migrating from RDD-based implementation to
DataFrame implementation**

NETFLIX

Negative Generator

Creating negatives from what
member plays for supervised learning

Facts

```
root
|-- profile_id: long (nullable = false)
|-- country_iso_code: string (nullable = false)
|-- items: array (nullable = true)
|   |-- element: struct (containsNull = false)
|   |   |-- show_title_id: long (nullable = false)
|   |   |-- label: double (nullable = false)
|   |   |-- weight: double (nullable = false)
```



Facts with synthetic negatives

```
root
|-- profile_id: long (nullable = false)
|-- country_iso_code: string (nullable = false)
|-- items: array (nullable = true)
|   |-- element: struct (containsNull = false)
|   |   |-- show_title_id: long (nullable = false)
|   |   |-- label: double (nullable = false)
|   |   |-- weight: double (nullable = false)
```

NETFLIX

DeLorean Feature Generator

Creating features based on common code base
in offline and online system

```
root
|-- profile_id: long (nullable = false)
|-- country_iso_code: string (nullable = false)
|-- items: array (nullable = true)
|   |-- element: struct (containsNull = false)
|   |   |-- show_title_id: long (nullable = false)
|   |   |-- label: double (nullable = false)
|   |   |-- weight: double (nullable = false)
```



```
root
|-- profile_id: long (nullable = false)
|-- country_iso_code: string (nullable = false)
|-- items: array (nullable = true)
|   |-- element: struct (containsNull = false)
|   |   |-- show_title_id: long (nullable = false)
|   |   |-- label: double (nullable = false)
|   |   |-- weight: double (nullable = false)
|   |   |-- features: struct (nullable = false)
|   |   |   |-- feature1: double (nullable = false)
|   |   |   |-- feature2: double (nullable = false)
|   |   |   |-- feature3: double (nullable = false)
```

Creating the Dataset for Algorithms

```
import org.apache.spark.ml.PipelineModel
val featurePipeline = new PipelineModel(Array(
  taggerTransformer,
  countryTenureStratifiedSampler,
  negativeGenTransformer,
  featureGenTransformer
))
val featureDF = featurePipeline.transform(playFeedDF)
```

Multithreading Model Training

For single machine multi-threading algorithms, we allocate one task to one machine. Multiple tasks are running in Spark for different parameters

Broadcast in Spark has datasize limitation, we write data into HDFS, and stream the data into the trainers in executors which run single-machine multi-threading algorithms

Distributed Model Training

We use both Spark ML's algorithms and in-house ML implementations

We keep the interface similar for both multi-threading and distributed algorithms, so experimenters can try different ideas easily

Scoring and Ranking

**Scorer is also a Transformer
returned from the Trainer**

**Multiple models can be scored at
the same time in parallel**

**The ranks are derived from sorted
scores**

**Together with labels, we can
compute metrics, NMRR, NDCG,
and Recall, etc**

```
root
|-- profile_id: long (nullable = false)
|-- country_iso_code: string (nullable = false)
|-- items: array (nullable = true)
|   |-- element: struct (containsNull = false)
|   |   |-- show_title_id: long (nullable = false)
|   |   |-- label: double (nullable = false)
|   |   |-- weight: double (nullable = false)
|   |   |-- features: struct (nullable = false)
|   |   |   |-- feature1: double (nullable = false)
|   |   |   |-- feature2: double (nullable = false)
|   |   |   |-- feature3: double (nullable = false)
|   |   |-- scores: struct (nullable = false)
|   |   |   |-- model1: double false
|   |   |   |-- model2: double false
```

Lessons Learned - Pipeline Abstraction

Pros

- **Modularity + Tests**
- **Plug-N-Play**
- **Notebook Prototyping**
- **Customizability**
- **Schema Verification**
- **Serializability (AC)**

Cons

- **Dependent on Spark platform**
- **Not easy to bring to production**
- **Default Metric Evaluator doesn't support ranking multiple models and type of metrics in one pass**



— ANY —
QUESTIONS