FUSING APACHE SPARK AND LUCENE FOR NEAR-REALTIME PREDICTIVE MODEL BUILDING

Debasish Das
Principal Engineer
Verizon

Pramod Lakshmi Narasimha
Principal Engineer
Verizon

Contributors

Platform: Pankaj Rastogi, Venkat Chunduru, Ponrama Jegan, Masoud Tavazoei
Algorithm: Santanu Das, Debasish Das (Dave)
Frontend: Altaff Shaik, Jon Leonhardt

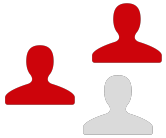
verizon

Data Overview

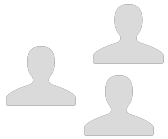
- Location data
  - Each srcIp defined as unique row key
  - Provides approximate location of each key
  - Timeseries containing latitude, longitude, error bound, duration, timezone for each key
- Clickstream data
  - Contains clickstream data of each row key
  - Contains startTime, duration, httphost, httpuri, upload/download bytes, httpmethod
  - Compatible with IPFIX/Netflow formats

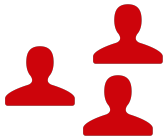
Marketing Analytics

- Anonymous aggregate analysis for customer insights

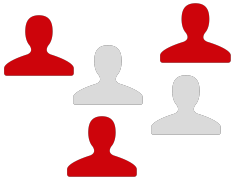


+ Lookalike modeling

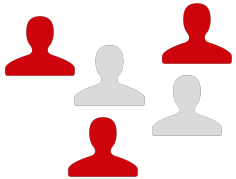




= Churn reduction



= Competitive analysis



+ Increased share of stomach

Data Model

- Dense dimension, dense measure
  Schema: srcip, date, hour, tld, zip, tldvisits, zipvisits
  Data: 10.1.13.120, d1, H2, macys.com, 94555, 2, 4
- Sparse dimension, dense measure
  Schema: srcip, date, tld, zip, clickstreamvisits, zipvisits
  Data: 10.1.13.120, d1, {macys.com, kohls.com}, {94555, 94301}, 10, 15
- Sparse dimension, sparse measure
  Schema: srcip, date, tld, zip, tldvisits, zipvisits
  Data: 10.1.13.120, d1, {macys.com, kohls.com}, {94555, 94301}, {macys.com:4, kohls.com:6}, {94555:8, 94301:7}
  Schema: srcip, week, tld, zip, tldvisits, zipvisits
  Data: 10.1.13.120, week1, {macys.com, kohls.com}, {94555, 94301}, {macys.com:4, kohls.com:6}, {94555:8, 94301:7}
  - Sparse dimension, sparse measure, **last N days**
    Schema: srcip, tld, zip, tldvisits, zipvisits
    Data: 10.1.13.120, {macys.com, kohls.com}, {94555, 94301}, {macys.com:4, kohls.com:6}, {94555:8, 94301:7}
- Competing technologies: PowerDrill, Druid, LinkedIn Pinot, EssBase

Document Dataset Representation

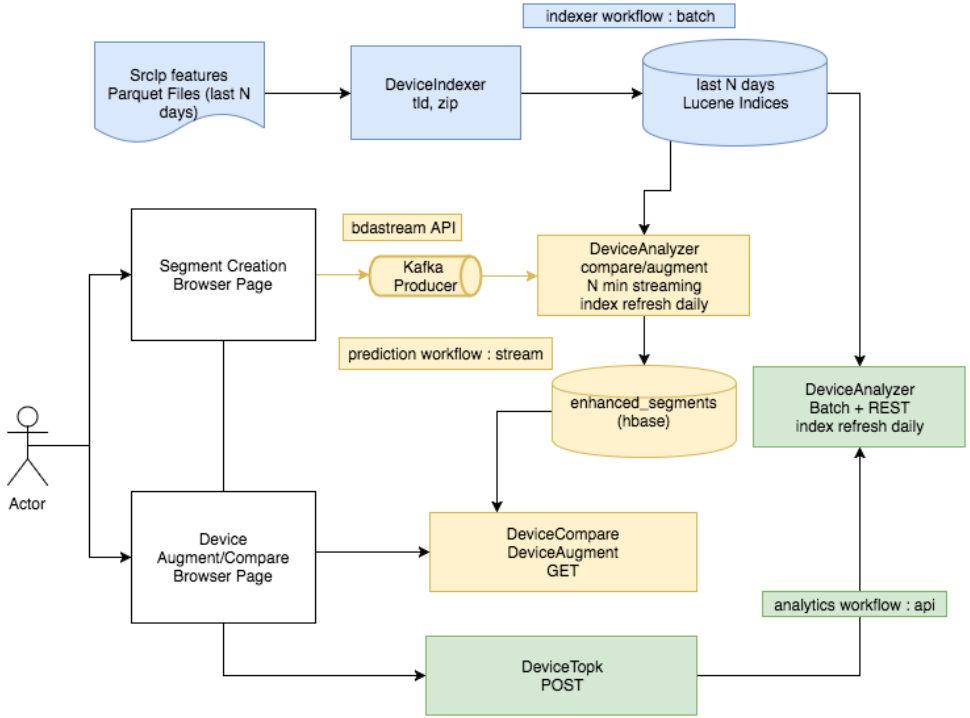
- Example
  Schema: srcip, tld, zip, tldvisits, zipvisits
  Data: 10.1.13.120, {macys.com, kohls.com}, {94555, 94301}, {macys.com:4, kohls.com:6}, {94555:8, 94301:7}
- DataFrame row to Lucene Document mapping

Store/schema	*Row*	*Document*
srcip	*primary key*	*docId*
tld *zip*	*String* *Array[String]*	*SingleValue/MultiValue* *Indexed Fields*
tldvisits *zipvisits*	*Double* *Map[String, Double]*	*SparseVector* *StoredField*

- Distributed collection of srcIp as RDD[Document]
  - ~100M srcip, 1M+ terms (sparse dimensions)

DeviceAnalyzer

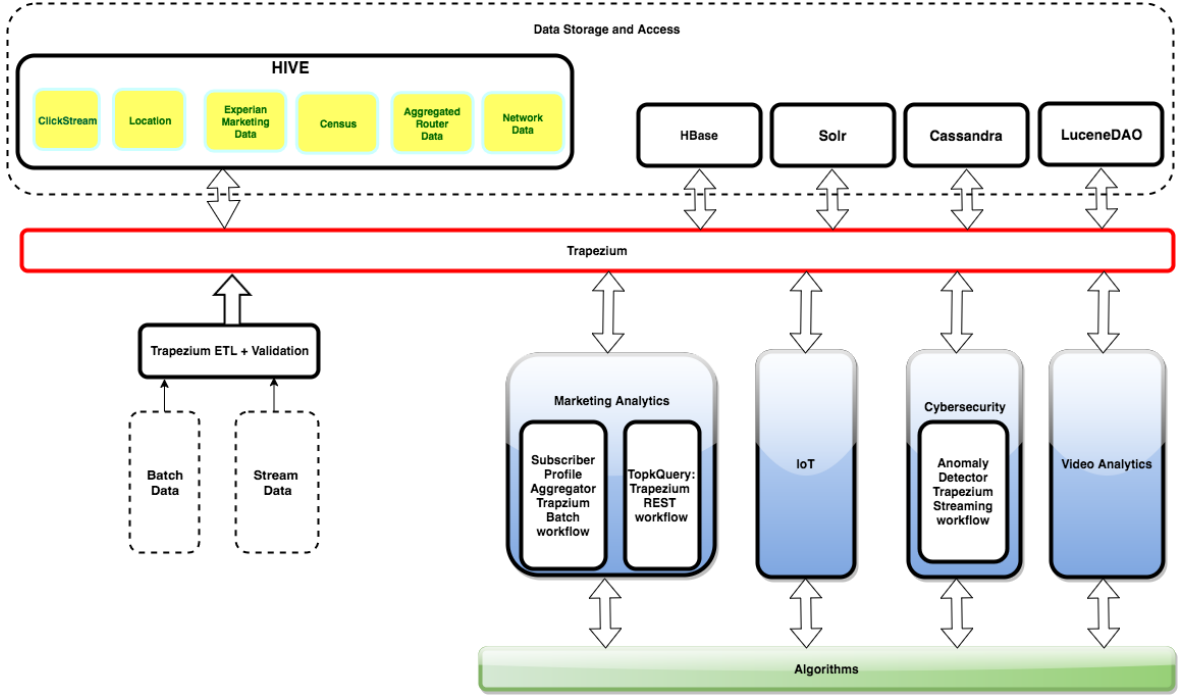
- DeviceAnalyzer goals
  - Search and retrieve devices that matched query
  - Generate statistical and predictive models on retrieved devices



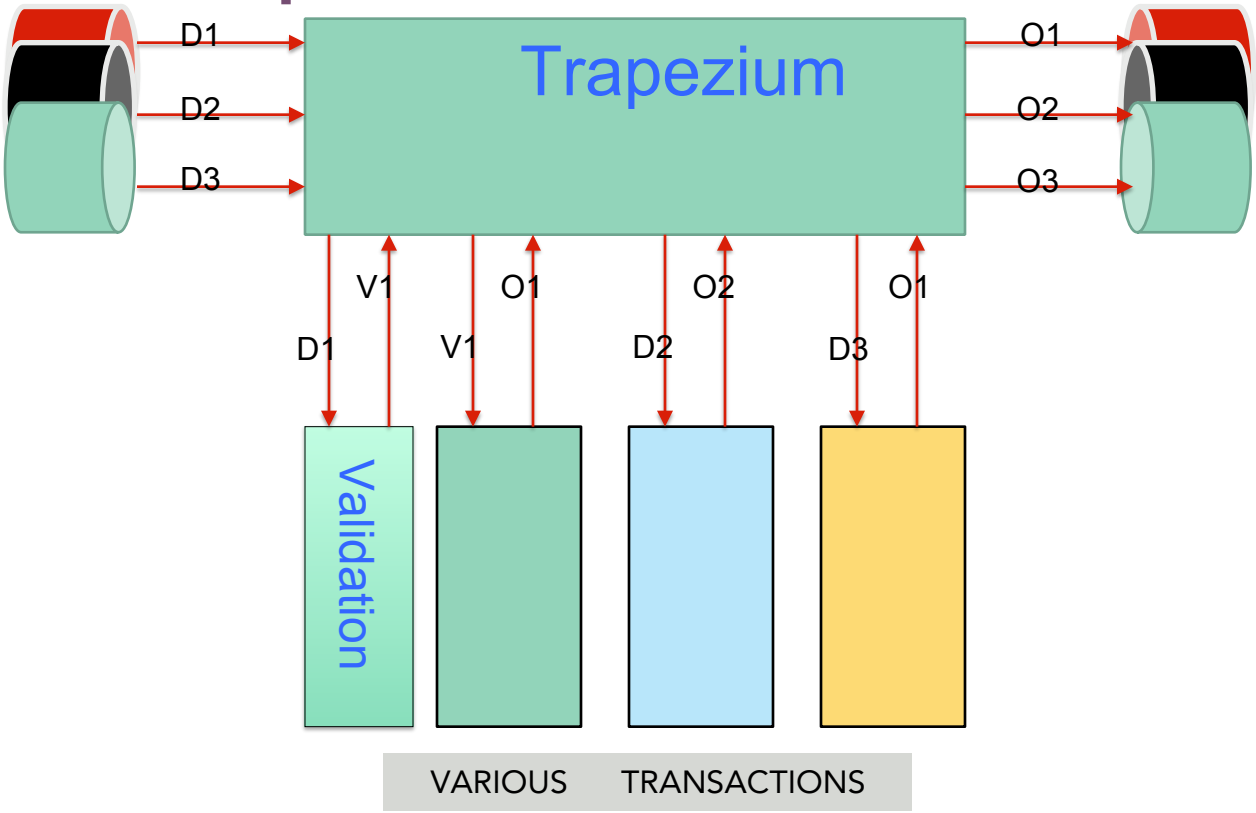
What is Trapezium ?

DAIS Open Source framework to build batch, streaming and API services

https://github.com/Verizon/trapezium

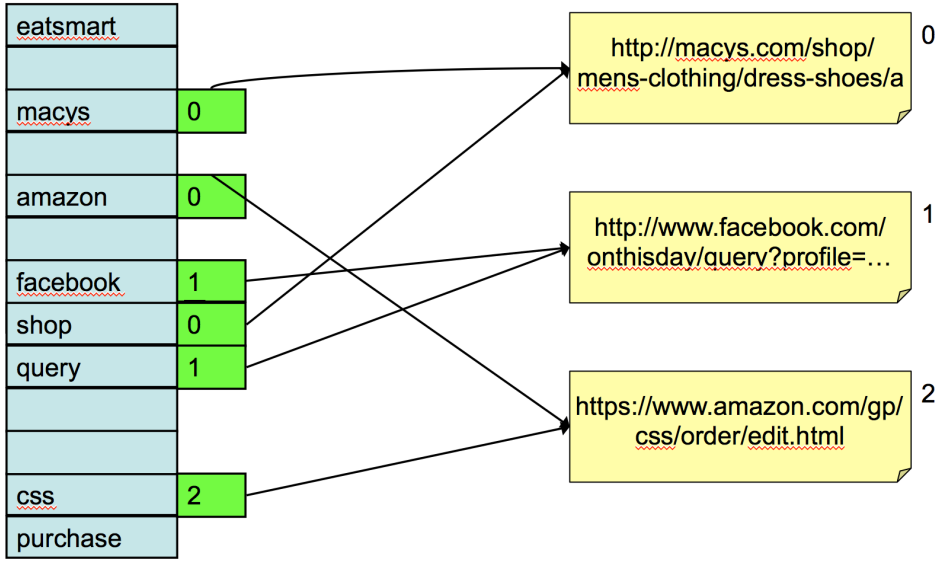


Trapezium Architecture



Lucene Overview

- Scalable, full-text search library
- Focus: Indexing + searching documents



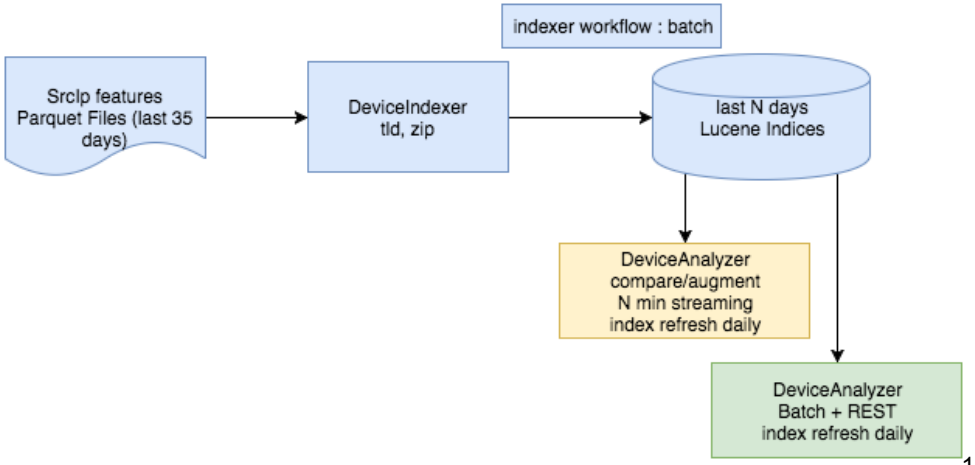
Trapezium LuceneDAO

- SparkSQL and MLlib optimized for full scan, **column indexing not supported**
- Why Spark + Lucene integration
  - Lucene is battle tested Apache Licensed Open Source Project
  - Adds column search capabilities to Spark
  - Adds spark operators (treeAggregate, treeReduce, map) to Lucene
- LuceneDAO features
  - Build distributed lucene shards from Dataframe
  - Save shards to HDFS for QueryProcessor (CloudSolr)
  - Access saved shards through LuceneDAO for ML pipelines

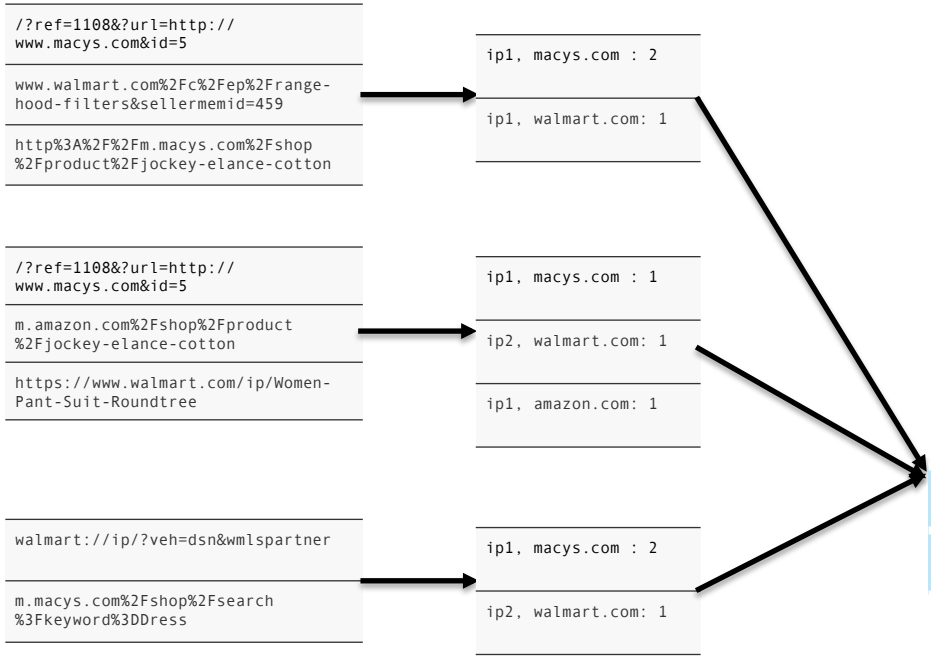
Trapezium Batch

```
runMode = "BATCH"
dataSource = “HDFS”
dependentWorkflows= {
    workflows=[aggregate]
    frequencyToCheck=100
}
hdfsFileBatch = {
  batchTime = 86400
  timerStartDelay = 1
  batchInfo = [{
     name = "DeviceStore"
     dataDirectory = {saiph-devqa=/aggregates}
     fileFormat = "parquet"
   }]
}
```

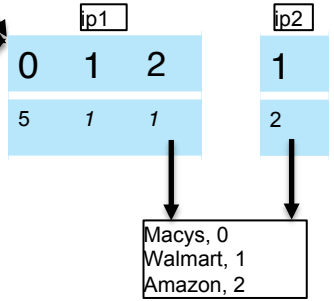
```
transactions = [{
  transactionName = “DeviceIndexer”
  inputData = [{name = "DeviceStore"}]
  persistDataName = “indexed"
}]
```



DeviceAnalyzer: Indexing

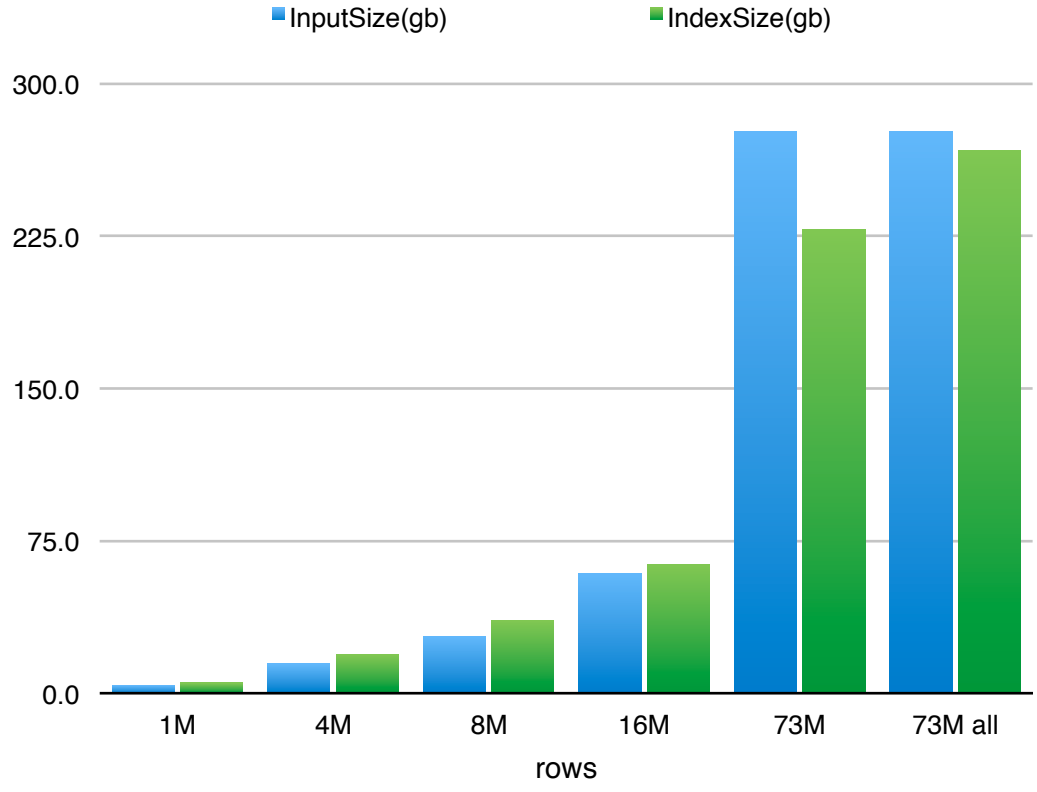



```
object DeviceIndexer extend BatchTransaction {
process(dfs: Map[String, DataFrame], batchTime: Time): {
    df = dfs("DeviceStore")
    dm = generateDictionary(df)
    vectorizedDf = transform(df, dm)
}
persist(df: DataFrame, batchTime: Time): {
    converter = SparkLuceneConverter(dm.size)
    dao = LuceneDAO(batchTime,…).setConverter(converter)
    dm.save(path, batchTime)
    dao.index(df, numShards)
}
```



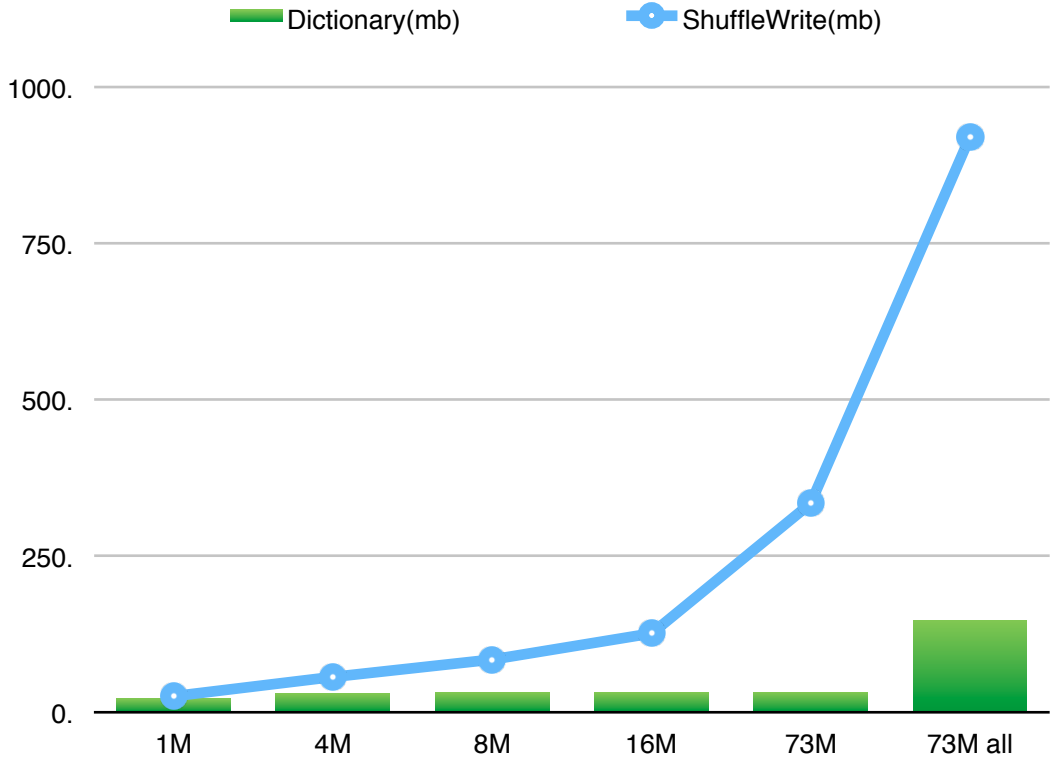
LuceneDAO Index Size

rows	InputSize(gb)	IndexSize(gb)
1M	4.0	5.1
4M	14.4	19.0
8M	27.9	35.7
16M	58.8	63.2
73M	276.5	228.0
73M all	276.5	267.1



LuceneDAO Shuffle Size

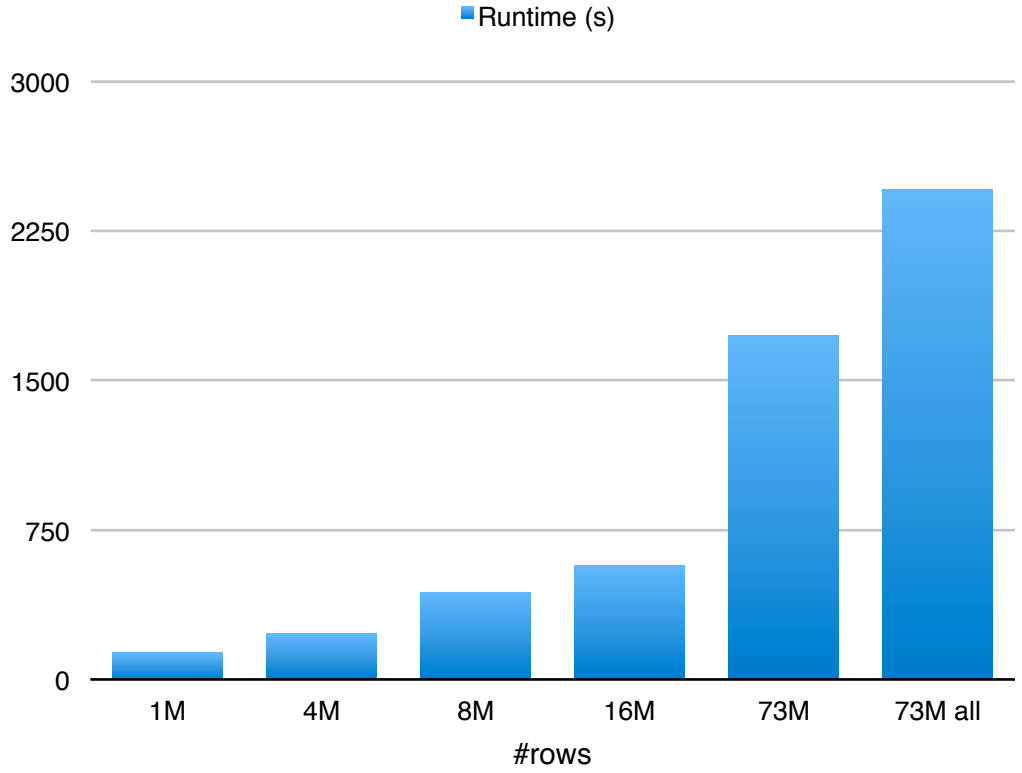
rows	ShuffleWrite(mb)	Dictionary(mb)
1M	25	22.0
4M	56	30.0
8M	85	31.6
16M	126	32.2
73M	334	32.4
73M all	921	146.5



LuceneDAO Index Runtime

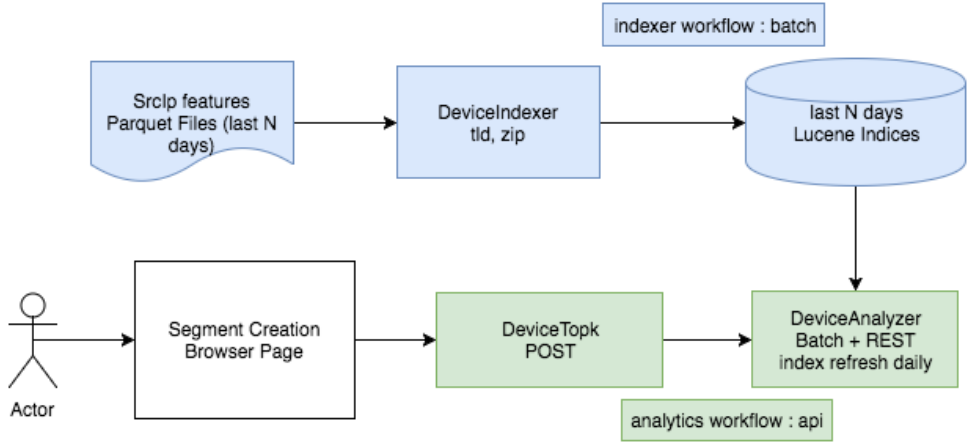
20 executors 16 cores
Executor RAM 16 GB
Driver RAM 8g

rows	Runtime (s)
1M	135
4M	228
8M	434
16M	571
73M	1726
73M all	2456



Trapezium Api

```
runMode = "BATCH"
dataSource = "HDFS"
httpServer = {
  provider = "akka"
  hostname = "localhost"
  port = 19999
  contextPath = "/"
  endPoints = [{
      path = “analyzer-api"
      className = "TopKEndPoint"
    }]
}
```



DeviceAnalyzer: Topk

- Given a query select * from devices where tld='macys.com' OR 'nordstorm.com' AND (city='SanFrancisco' OR 'Brussels') AND (device='Android') …
  - ML: Find topk dimensions highly correlated with selected device
  - BI: group by tld order by sum(visits) as tldVisits limit topk

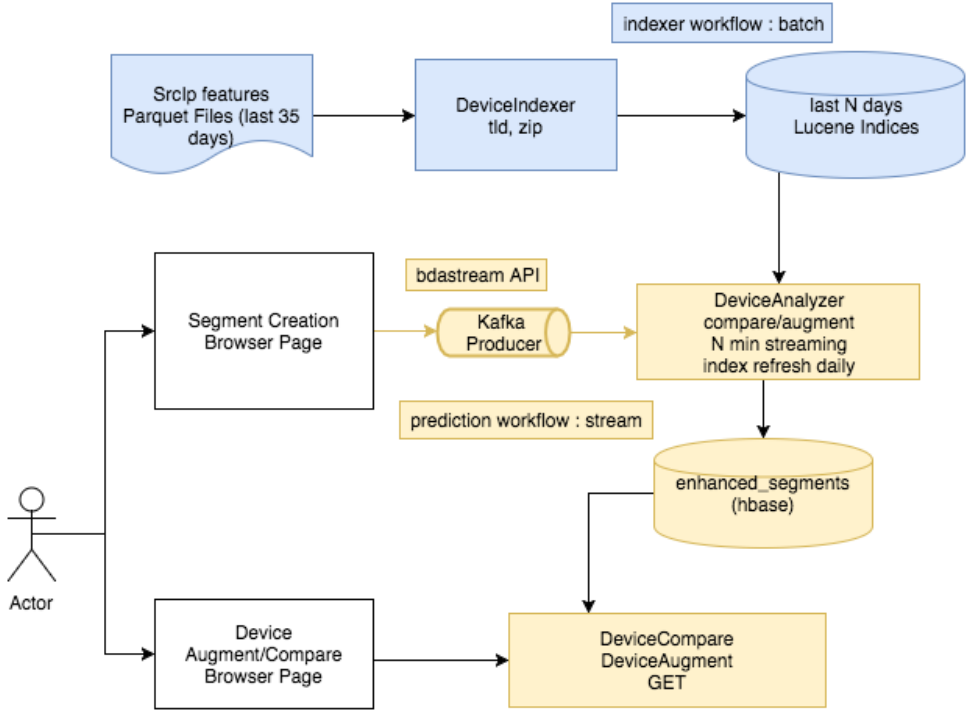
```
class TopkController(sc: SparkContext) extends
SparkServiceEndPoint(sc) {
override def route : topkRoute
converter = SparkLuceneConverter(dm.size)
batchTime = Trapezium.getSyncTime("indexer")
dao = LuceneDAO(batchTime…)
      .setConverter(converter).load(sc, indexPath)
dict = loadDictionary(sc, indexPath, batchTime)
def topkRoute : {
  post { request => {
    devices = dao.search(request)
    response = getCorrelates(devices, dict, topk)
  }
}
```

df[deviceId, vector]

sum, support
mean, median, stddev

Trapezium Stream

```
runMode = "STREAM"
dataSource = “KAFKA"
kafkaTopicInfo = {
  consumerGroup = "KafkaStreamGroup"
  maxRatePerPartition = 970
  batchTime = "5"
  streamsInfo = [{
    name = "queries"
    topicName = “deviceanalyzer”
}]
}
transactions = [{
  transactionName = DeviceAnalyzer"
  inputStreams = [{name: "queries"}]
  persistStreamName = "deviceanalyzer"
  isPersist = "true"
}]
```



DeviceAnalyzer: Compare

- Given two queries

select * from Devices where tld='macys.com' OR 'nordstorm.com' AND (city='SanFrancisco') AND (device='Android')

select * from Devices where tld='macys.com' OR 'nordstorm.com' AND (city='Brussels') AND (device='Android')

- Find the dimensions that discriminate the devices associated with two groups

```
def processStream(streams: Map[String, DStream[Row]], workflowTime: Time): {
 streams("queries").collect().map{ requests =>
    group1 = dao.search(requests(0))
    group2 = dao.search(requests(1))
    response = runLDA(aud1, aud2, dict)
}
```

- Sparse weighted least squares using Breeze QuadraticMinimizer
- L1 Regularized logistic regression

```
def persistStream(responses: RDD[Row], batchTime: Time) {
     HBaseDAO.write(responses)
}
```

DeviceAnalyzer: Augment

- Given a query

select * from Devices where
tld='macys.com' OR 'nordstorm.com'
AND (city='SanFrancisco' OR 'Brussels')
AND (device='Android')…

  - Find devices similar to seed as lookalikes
  - Find dimensions that represent lookalikes

```
object DeviceAnalyzer extends StreamingTransaction {
converter = SparkLuceneConverter(dm.size)
batchTime = Trapezium.getSyncTime("indexer")
dao = LuceneDAO(batchTime…)
      .setConverter(converter).load(sc, indexPath)
dict = loadDictionary(sc, indexPath, batchTime)
all = dao.search("*:*")
def processStream(streams: Map[String, DStream[Row]]) :
{
  streams("queries").collect().map{ request =>
    audience = dao.search(request)
    response = getLookalikeDimensions(all, audience, dict)
}
```

- Sparse weighted least squares using Breeze QuadraticMinimizer
- L2 regularized linear regression

FastSummarizer

- Statistical and predictive operators
  - sum: sum over numeric measures
  - support: sum over distinct docID
  - sumSquared: L2 norm
  - gram: Uses BLAS sspr
  - solve: Uses Breeze QuadraticMinimizer to support L1
- Implemented using Array[Float] for shuffle opt
- Scala/Java for Level1 operations
- OpenBLAS for Level3 operations

Sync API Benchmark

73M rows 1M+ search terms

1 measure on 250K sparse dimensions

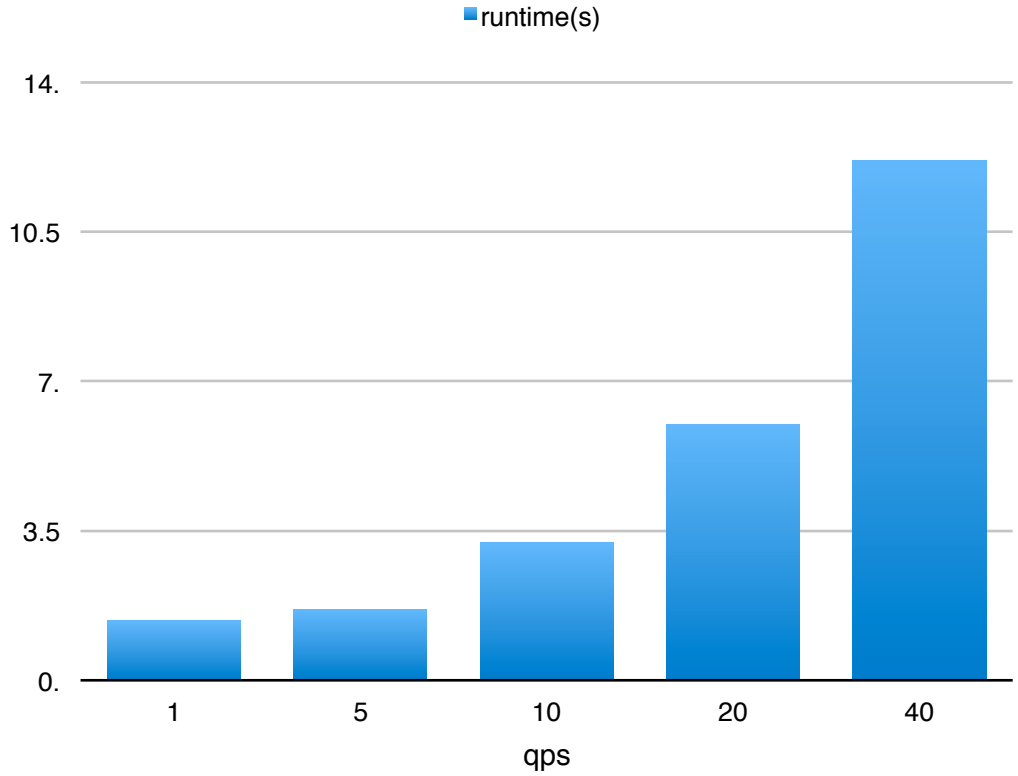
20 executors 8 cores

32 GB driver RAM 16 GB executor RAM

akka-http cores: 24 default

topk

qps	runtime(s)
1	1.389
5	1.663
10	3.214
20	5.992
40	12.174



Async API Benchmark

73M rows, 1M+ search terms

1 measure on 250K sparse dimensions

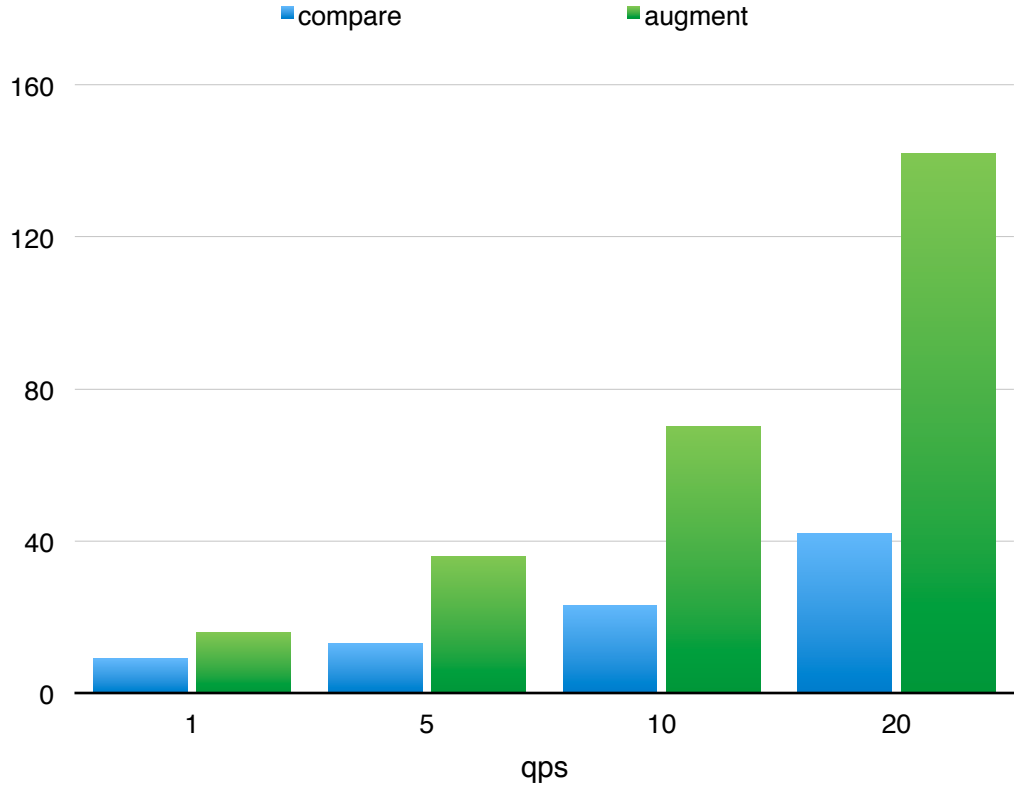
20 executors 8 cores

32 GB driver RAM 16 GB executor RAM

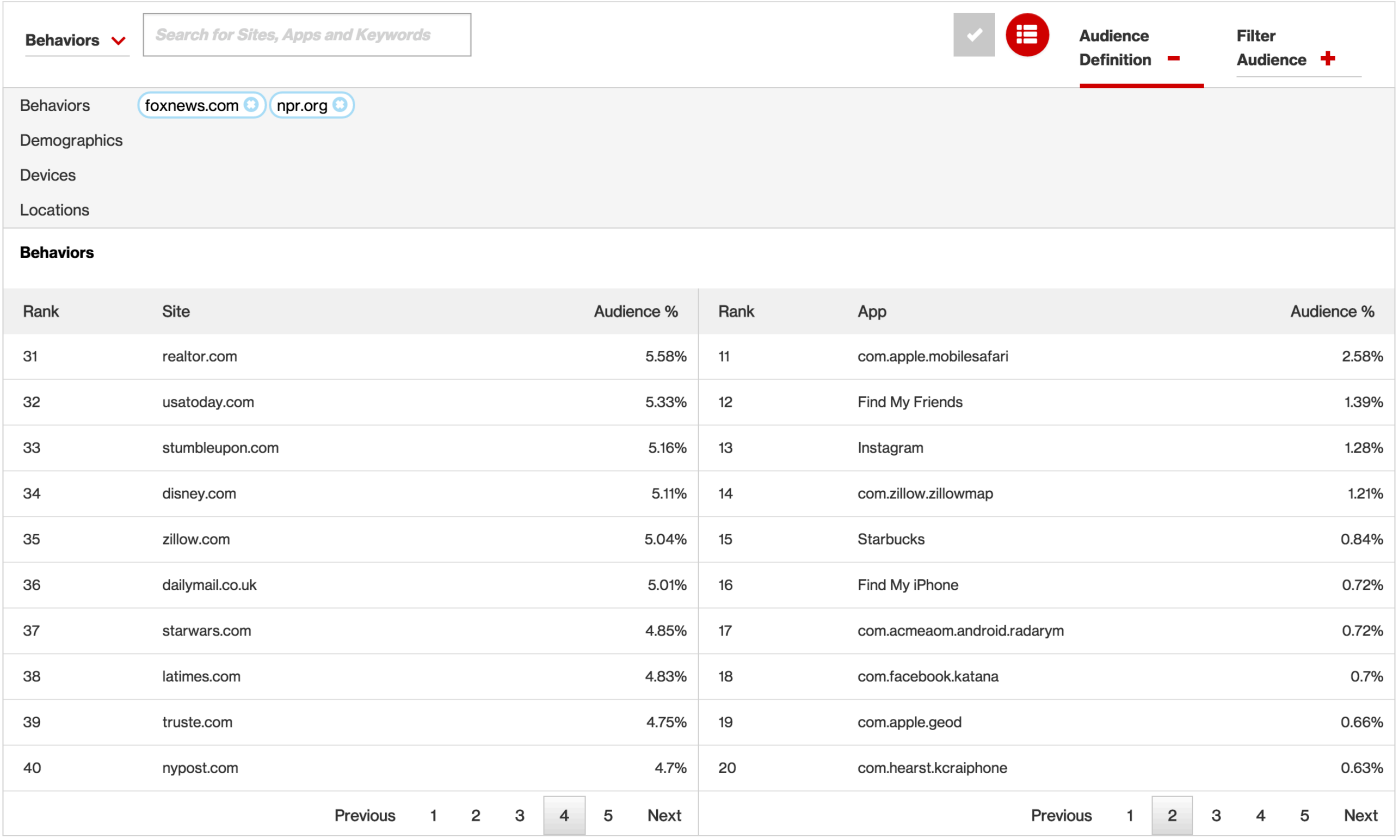
forkjoinpool = 40

Kafka Fetch + compare/augment + HBase Persist

predictions		
qps	**compare(s)**	**augment(s)**
1	9	16
5	13	36
10	23	70
20	42	142



topk tld + apps



Behaviors

Rank	Site	Audience %
31	realtor.com	5.58%
32	usatoday.com	5.33%
33	stumbleupon.com	5.16%
34	disney.com	5.11%
35	zillow.com	5.04%
36	dailymail.co.uk	5.01%
37	starwars.com	4.85%
38	latimes.com	4.83%
39	truste.com	4.75%
40	nypost.com	4.7%

Previous 1 2 3 4 5 Next

Rank	App	Audience %
11	com.apple.mobilesafari	2.58%
12	Find My Friends	1.39%
13	Instagram	1.28%
14	com.zillow.zillowmap	1.21%
15	Starbucks	0.84%
16	Find My iPhone	0.72%
17	com.acmeaom.android.radarym	0.72%
18	com.facebook.katana	0.7%
19	com.apple.geod	0.66%
20	com.hearst.kcraiphone	0.63%

Previous 1 2 3 4 5 Next

Augment: Auto Enthusiastic

AutoEnthusiasticE2-31Days

Auto Enthusiastic

Top Prediction Features

The number of 'Reach' subscribers is the remainder, or difference, between the Seed Audience counts and the Look-alike Audience counts.

Min — Max — Apply

All Features

Rank	Feature	Weight
1.	Site: autotrader.com	
2.	Site: kia.com	
3.	Site: carcomplaints.com	
4.	Site: autobytel.com	
5.	Site: car.com	
6.	Site: showroomlogic.com	
7.	Site: dealerfire.com	
8.	Site: hyundaiusa.com	
9.	Site: ford.com	
10.	Site: jeep.com	

Look-alike Audience

Look-alike Audience: 5.55M

Seed Audience: 4.22M

Create This Audience | Save This Audience | Show Seed Audience Stats

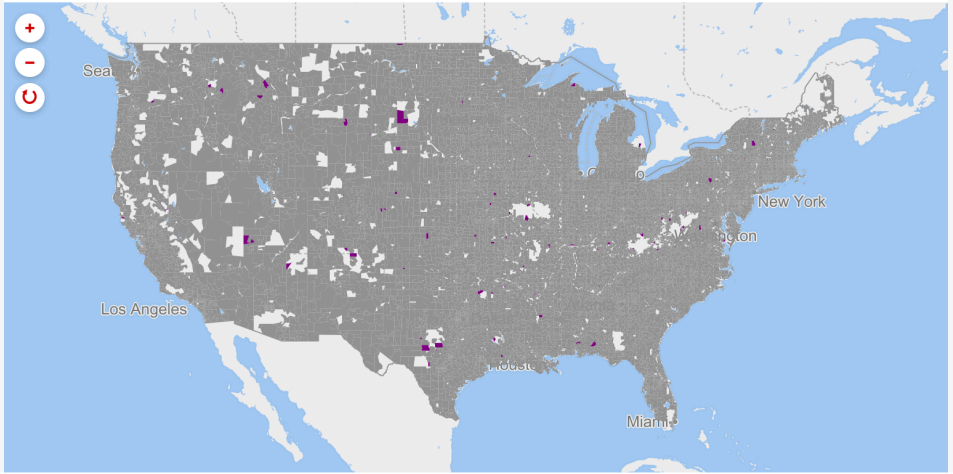
Seed Audience: AutoEnthusia ...

Seed Audience Definition: Site: autoweek.com, caranddriver.com, edmunds.com, topspeed ...

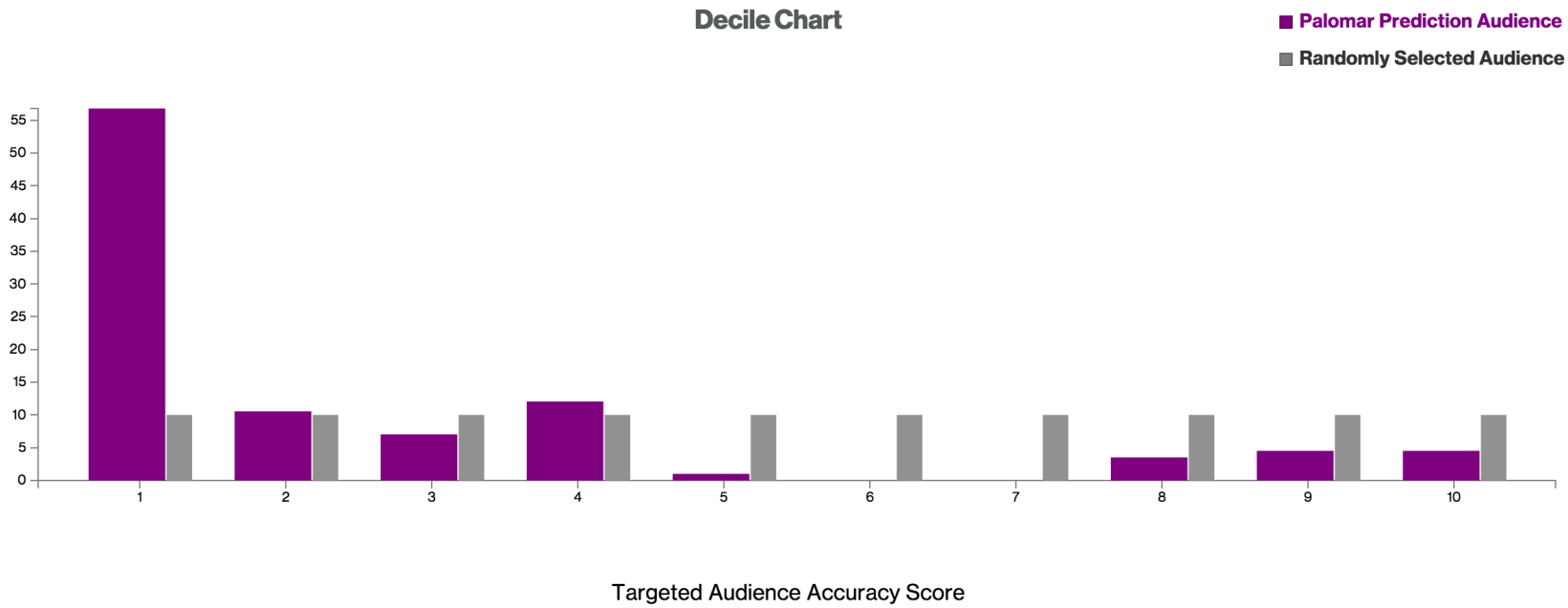
Top Prediction Features

Behaviors: 1. autotrader.com | 2. kia.com | 3. carcomplaints.com | 4. autobytel.com | 5. car.com | 6. showroomlogic.com | 7

Locations



Augment Model Performance



Compare: Leisure vs Business Travellers

LeisurevsBusinessTravellerE2

Leisure vs Business Traveller

LeisureTravelersAud

Audience Definition:

Site: orbitz.com, expedia.com, kayak.com, cheapcaribbean.com, bookit.com, fun...

BusinessTravellerAud

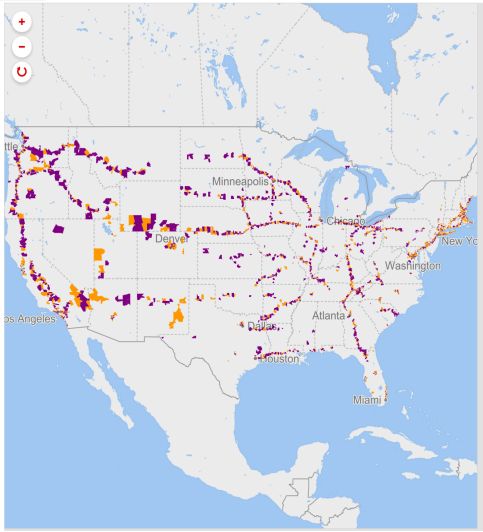
Audience Definition:

Site: expedia.com, travelocity.com, kayak.com, united.com, aa.com, virginamer...

Behavioral Differentials

Min Max Min Max

Sites & Apps	Sites & Apps
Site : estara.com	Site : journeys.com
App : com.hotwire.hotwire	Site : tsa.gov
Site : travelnow.com	App : Marriott International
App : Priceline - Book Chea...	App : com.hotels.hotelsnea...
Site : aircanada.com	App : com.apple.mobilemail
Site : riu.com	Site : ifly.com
Site : trivago.com	Site : passkey.com
Site : applevacations.com	Site : supershuttle.com
Site : cruise.com	Site : medallia.com
App : Orbitz Flights, Hotels,...	Site : hilton.com
Site : hotels.com	Site : usairways.com



Location Differentials

Min Max Min Max

Zip Codes	Zip Codes
89830	30337
59832	55111
92364	48242
19079	23250
19142	78719
87711	11096
59024	95837
50048	19153
63801	98158
80019	94128
30344	84116
83335	32827
98857	76051
58622	80249
30349	97218
93215	64153

THANK YOU.
Q&A

Join us and make machines intelligent
Data & Artificial Intelligence Systems
499 Hamilton Ave, Palo Alto
California

verizon