From Clicks to Models

The Wikimedia LTR Pipeline

Wikimedia Search Platform

- 300 languages
- 900 wikis
- 85% of search to top 20
- 4TB in primary shards
- 30M+ full text/day
- 50M+ autocomplete/day
- 150M+ more like/day
- 2 clusters in separate DCs
- Team of 5 engineers

WIKIMEDIA
FOUNDATION

MjoLniR - Machine Learned Ranking

- https://github.com/wikimedia/search-mjolnir
- Pyspark
- Some Scala

WIKIMEDIA
FOUNDATION

From Zero to Deployment

How we got there

- Start with offline POC
- Build major steps of transformation
- Reuse existing features of ranking function
- Build an ML ranker that learns the existing ranking function
- This means it works!

Click Logs

WIKIMEDIA FOUNDATION

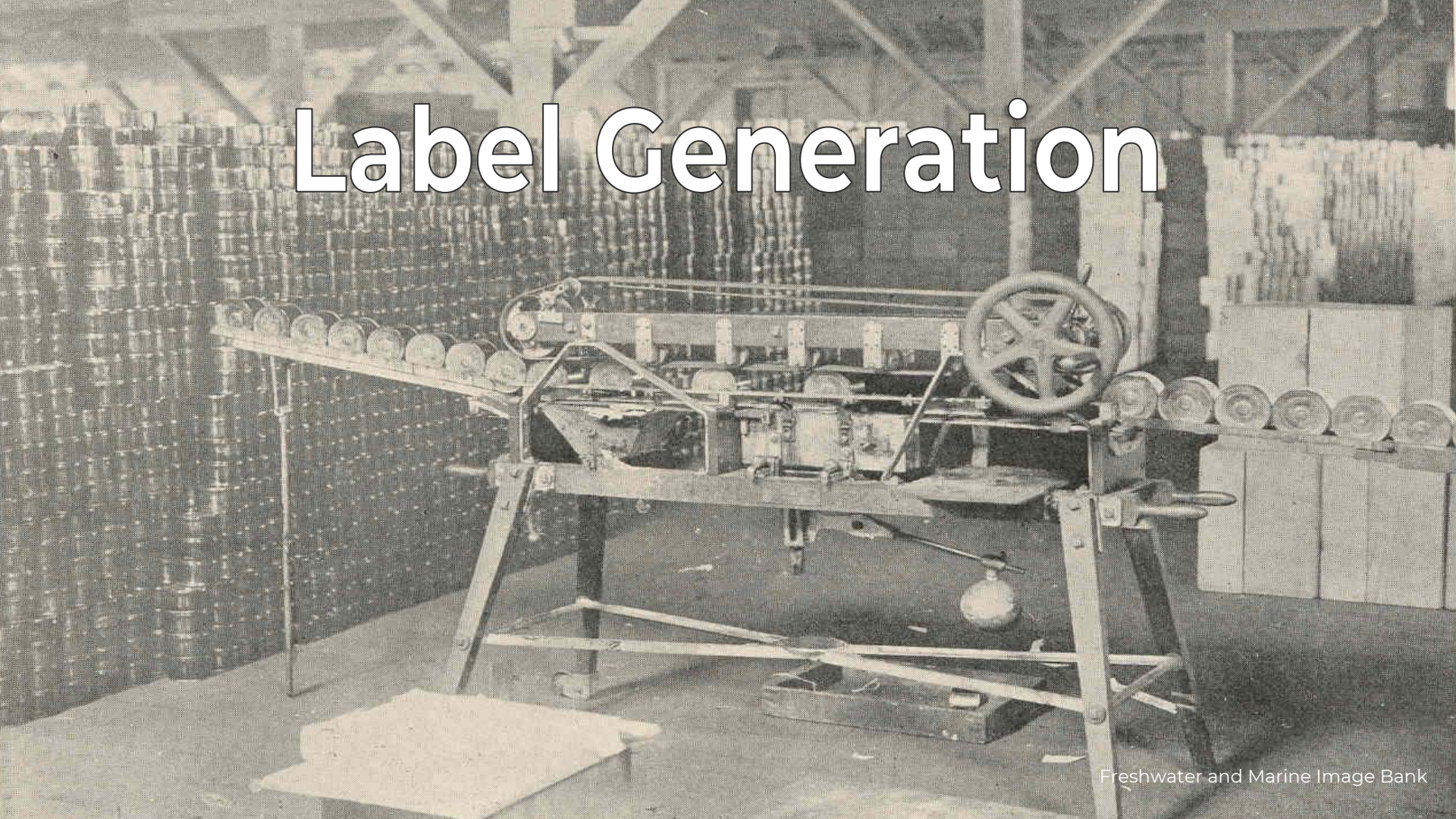
Collection

- Varnish -> kafka
- App server -> kafka
- Data retention of 90 days
- ~1M sessions with clicks per day
- ~500MB compressed, with debug info, per day
- Reused existing webrequest logging infrastructure

timestamp: int
site: string
session_id: string
query: string
hits: array<int>
clicks: array<int>

Challenges	Solutions
• Bot filtering • Skew when sessionizing • Unclear search logs	• Drop logs from busy ip's • Iterate on logging

Label Generation



Freshwater and Marine Image Bank

Click Models

- Click models provide a principled way to translate implicit preferences into unbiased labels
- Accounts for biases like result position and snippet attractiveness
- DbnModel implementation from python clickmodels library
- Operates on groups of sessions with the same intent
- No shared information between query groups makes this an embarrassingly parallel problem

Challenges	Solutions
• Shuffling data between JVM and python is slow • Python implementation was unoptimized	• Rewrote in scala with no allocation in the tight loops for 100x speedup.

Challenges	Solutions
• Mediocre results for queries with few sessions • Limiting to queries with 10 repeats drops all but 22% of sessions.	• Normalize query strings • Naive grouping improves to 30% • Aggressive grouping improves to 45%

Normalize Query Strings

Better grouping:

- Better labels
- Inclusion of long tail

Simple but effective:

```
lower(trim(query))
```

WIKIMEDIA FOUNDATION

Can we do better?

Throw stemmers at it!

The Good

- the lucas brother, lucas brothers, lucas brotheres, lucas brother, the lucas brothers
- herbes provence, le herbs de provence, herbe provence, etc.
- julian dates, julian date, julian dating

The Bad

- marin, marine, mariner, mariners
- nature, natural, naturalism
- british colonial, british colonies, the british colonies

Break up groups

Collect Top N hits for every query and apply clustering within query groups

WIKIMEDIA FOUNDATION

Better:

- [Marin], [marine], [mariner, mariners]
- [nature], [natural], [naturalism]

But not great:

- [marine corp rank], [marine corp ranks]
- [witches], [the witch], [witchs]

Features

WIKIMEDIA
FOUNDATION

Feature Engineering

- Initial models used 10 similarity features and 2 document only features
- Training captured 20% of possible improvement in ndcg@10
- Translated into 1.5% increase in click throughs, 0.5% decrease in session abandonment

QD Features

- Match query for each field analyzed two ways
- Phrase match on specific fields
- Query explorer
- Dismax via feature expressions
- Future: SimSwitcher

Doc only Features

- Popularity score
- Incoming link counts
- Page length in bytes and tokens

Query only Features

- Per-field idf
- # of unique terms with limited and aggressive analysis

WIKIMEDIA
FOUNDATION

Collecting Features

Point the hadoop cluster at the elasticsearch cluster to collect vectors for millions of queries. What could go wrong?

Challenges	Solutions
• 250 features is slow (~300ms) • memory for training is linear with # of features	• mRMR feature selection • Achieves 80% of the improvement of 250 features with only 50 • Previous feature set achieved 60%

Training

WIKIMEDIA
FOUNDATION

CC by SA 2.0, Kurt Rasmussen

Resource Allocation

Challenges	Solutions
• Training data spans two orders of magnitude • Efficient use of limited compute resources	• Split sites into three groups by size • Heuristics to determine needs from data sizes

Hyper-parameter Search

- Using python hyperopt
- Customized for parallel search through spark
- Models train on single executor
- Train 50-150 models in parallel

Resource Usage

Challenges	Solutions
• Yarn killing executors • Unpredictable memory usage	• Don’t send training data through spark • Point xgboost at files on HDFS directly

Other Thoughts

WIKIMEDIA FOUNDATION

CC by SA 2.0, greyloch

Spark on Yarn

Never as easy as it looks

```
SPARK_HOME=/usr/lib/spark2 USER=ebernhardson PATH=/bin:/usr/bin HOME=/home/ebernhardson
PYSPARK_PYTHON=venv/bin/python SPARK_CONF_DIR=/etc/spark2/conf \
    /usr/lib/spark2/bin/spark-submit \
    --conf spark.dynamicAllocation.cachedExecutorIdleTimeout=120s \
    --conf spark.dynamicAllocation.executorIdleTimeout=60s \
    --conf spark.dynamicAllocation.maxExecutors=112 \
    --conf spark.task.cpus=4 --conf spark.yarn.executor.memoryOverhead=5748 \
    --archives /home/ebernhardson/mjolnir/mjolnir_venv.zip#venv \
    --driver-memory 3G  --executor-cores 4  --executor-memory 2G \
    --master yarn --queue nice \
    --packages
ml.dmlc:xgboost4j-spark:0.8-wmf-2,org.wikimedia.search:mjolnir:0.4,org.apache.spark:spark-streaming-kafk
a-0-8_2.11:2.1.2,sramirez:spark-infotheoret
ic-feature-selection:1.4.4,sramirez:spark-MDLP-discretization:1.4.1 \
    --repositories
https://archiva.wikimedia.org/repository/releases,https://archiva.wikimedia.org/repository/snapshots,https://
archiva.wikimedia.org/repository/mirrored \
    /srv/deploy/mjolnir/venv/bin/mjolnir-utilities.py training_pipeline \
    --cv-jobs 130 --final-trees 100 --iterations 100 \
    --input hdfs://analytics-hadoop/user/ebernhardson/mjolnir/20180316-folds-medium \
    --output /home/ebernhardson/training_results/20180316-medium \
    itwiki ptwiki frwiki ruwiki
```

WIKIMEDIA
FOUNDATION

Challenges	Solutions
● Takes a bazillion CLI args to configure	● Configuration driven script to call spark

Challenges	Solutions
● Doesn’t play nice with large off-heap memory allocation ● Many values have to be tuned based on the size of data being processed	● Split pipeline into multiple independent scripts by resource needs ● Save metadata next to data with stats on sizes ● Heuristics to translate into memory reqs

WIKIMEDIA
FOUNDATION

Metadata

- Keep as much as possible
- Record collection parameters with the output data.
- Add to the metadata at each step of the pipeline to report on what happened, why, etc.
- Data retention policies may require data to be deleted after N days, but aggregated data in the form of models, training history, etc should be kept for later analysis.

Public Data

Now

Weekly dumps of production search indices in elasticsearch bulk import format[1].

Soon

Public read-only access to elasticsearch with live updated indices in WMF Cloud[2].

[1] https://dumps.wikimedia.org/other/cirrussearch
[2] https://wikitech.wikimedia.org/wiki/Help:Cloud_Services_Introduction

THANK YOU

(Camel of knowledge) →

WIKIMEDIA
FOUNDATION