Early Bird Inc.

NYC Developer Week Hackathon Presentation

June 20th, 2018

Agenda

- **Project Introduction**
- Details of completed steps
- Next Steps for Early Bird

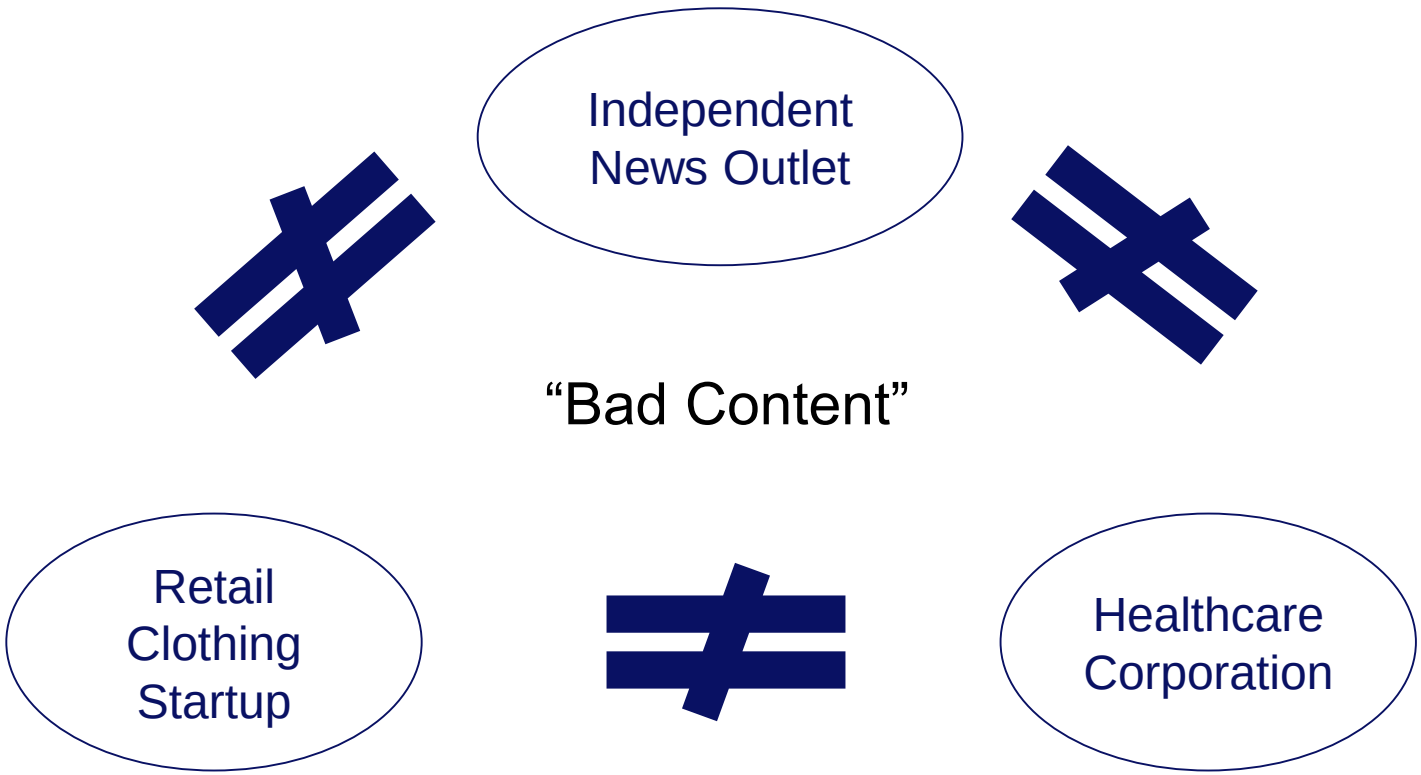
What is one of the largest issues - CONTENT!

- As information availability and ML targeting methods have become increasingly popular, Malicious Content is one of the world's most difficult topics to tackle

It is clear that big companies are being forced to take immediate action

However, this may create some issues for some of the very clients that they serve

Different small to medium size businesses often want to rely on different definitions of malicious content



Big Companies know this challenge and are offering solutions

Google expands its tool for publishers to combat ad blocking

APRIL 16, 2018 *by* Lucia Moses

- However, these solutions are often too clunky and do not let the client define what they would like to consider “Malicious” in the context of their business

So what is Early Bird?

- A solution that allows anyone to tune their own definition of “Malicious Content”
- Does this based upon user needs using the power of “implicit” interpretation through machine learning
- Technical Fundamentals
  - Train multiple types of ML algorithms to detect Malicious content
  - Combine results together to create a most conservative approach
  - Show the user results of what “MC” content they would want to keep

Use Cases for Early Bird

- Publisher provider (AppNexus, Google, etc.) can provide the application to clients who can “tune” their definition of appropriate and malicious content

- A growing business that allows users to share their own content
  - Example: Pinterest

Agenda

- Project Introduction
- **Details of completed steps**
- Next Steps for Early Bird

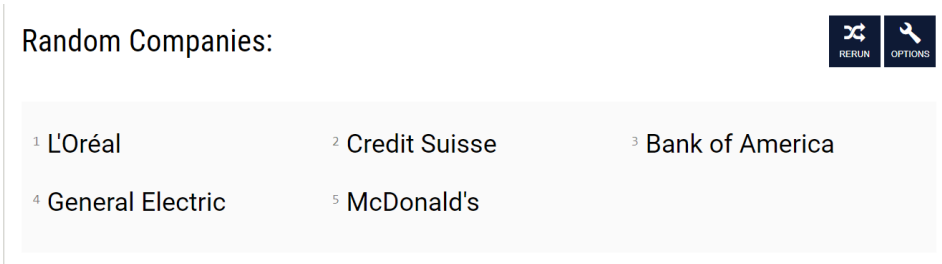
Our Development Process So Far

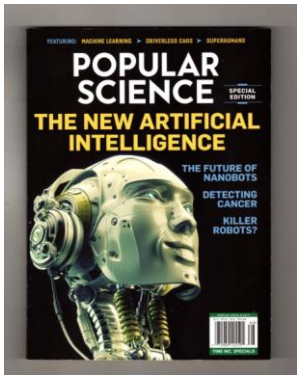
- Acquire Data for “Quality” Advertisements creating three distinct groups:
  - Approved, Misleading, and Tabloid Ads

- Decide on 2 Test ML Algorithms
  - Language Sentiment Analysis
  - Image Classification

Aggregation of Approved Ads

- Without a defined data set, this presented a key challenge
- Decided on “Established Fortune 500” company advertisements for first round of training
  - Utilized online Random List Generator to remove bias from selection
- Additionally, used random Print Advertisements from sources like magazines that would be fundamentally different than web ads





Language Sentiment Analysis

- Overview of Steps
  - Extract Text from Photos
  - Sentiment Analysis on Text
  - Pattern Recognition

- Libraries Used
  - Natural Language Toolkit (Academic Research)
  - Stanford NLP
  - Sci-kit Learn
  - Tesseract

Language Sentiment Analysis: Text Extraction

In [42]:
```
def getText(tabloidDir, misleadingDir, goodDir):
    texts = []

    directory = os.fsencode(tabloidDir)
    for filename in os.listdir(directory):
        filename = str(filename)
        filename = filename[2:len(filename)-1]
        texts.append(readText(tabloidDir+"/"+filename))

    directory = os.fsencode(misleadingDir)
    for filename in os.listdir(directory):
        filename = str(filename)
        filename = filename[2:len(filename)-1]
        texts.append(readText(misleadingDir+"/"+filename))

    directory = os.fsencode(goodDir)
    for filename in os.listdir(directory):
        filename = str(filename)
        filename = filename[2:len(filename)-1]
        texts.append(readText(goodDir+"/"+filename))

    return texts
```

Language Sentiment Analysis: Clean Text

```
def readText(filename):
    image = np.asarray(Image.open(filename))
    image = rescale(image, 3, mode = "reflect")
    image *= 255
    image=Image.fromarray(image.astype('uint8'))

    clean = image_to_string(image).replace("\n", ". ")
    clean= ''.join([x for x in clean if (x in string.ascii_letters + string.digits + " " + ".") ])
    return clean
```

In [18]:
```
texts = getText("C:/Users/tectonic/Downloads/Tabloid Advertorial", "C:/Users/tectonic/Downloads/Misleading
 Claims", "C:/Users/tectonic/Downloads/Good Ads")
```

In [117]:
```
texts = [msg.lower() for msg in texts]
```

Language Sentiment Analysis: Sentiment Analysis

In [35]:
```
nlp = StanfordCoreNLP('http://localhost:9000')

def getSentiments(texts):
    sentiments = []
    for adMessage in texts:
        res = nlp.annotate(adMessage,
                           properties={
                               'annotators': 'sentiment',
                               'outputFormat': 'json',
                               'timeout': 1000,
                           })

        overallSentiment = 0
        try:
            for s in res["sentences"]:
                if (s["sentiment"] == "Negative"):
                    overallSentiment -= int(float(s["sentimentValue"]))
                elif (s["sentiment"] == "Positive"):
                    overallSentiment += int(float(s["sentimentValue"]))
            sentiments.append(overallSentiment)
        except:
            sentiments.append(0)
    return sentiments
```

In [36]:
```
sentiments = getSentiments(texts)
```

In [305]:
```
d = {'file': filenames, 'sentiments': sentiments, 'text':texts}
df = pd.DataFrame(data=d)
df['pos sent'] = [1 if val >= 0 else 0 for val in df['sentiments']]
df['is tabloid'] = [1 if file.startswith("Tabloid") else 0 for file in df['file']]
df['is misleading'] = [1 if file.startswith("Misleading") else 0 for file in df['file']]
df['is bad'] = [1 if file.startswith("Tabloid") or file.startswith("Misleading") else 0 for file in df['fi
le']]
df['type'] = [file[:file.index(" ")] for file in df['file']]
df
```

Language Sentiment Analysis: Derive Scores

In [345]:
```
def getBigrams(category):
    badtexts = df.loc[df[category] == 1]["text"]
    words = nltk.word_tokenize(" ".join(badtexts.tolist()))
    cleanerWords = [i for i in words if i.isalpha() and len(i) > 2]
    bigrams = nltk.bigrams(cleanerWords)
  #  print(Counter(bigrams).most_common()) #read more, more sponsored, simple trick, discover how, stay yo
ung, hollywoods elite
    return Counter(bigrams)

def getKeywords(category):
    stop = stopwords.words('english')
    badtexts = df.loc[df[category] == 1]["text"]
    words = nltk.word_tokenize(" ".join(badtexts.tolist()))
    cleanerWords = [i for i in words if i.isalpha() and len(i) > 2 and i not in stop]
  #  print(Counter(cleanerWords).most_common()) #read, sponsored, reveals, trick, seconds
    return Counter(cleanerWords)
```

In [348]:
```
def language_score(language, texts):
    filterWords = ["nonsecure", "view", "chrome", "file", "edit", "help", "bookmarks", "tab", "creatives",
 "inventory", "preview", "window"]
    scores = []
    for msg in texts:
        score = 0
        for word in language:
            wordFreq = language[word]
            if type(word) == tuple:
                word = ' '.join(word)
            if word in filterWords or word[:word.find(" ")] in filterWords:
                continue
            if msg.find(word) != -1:
                score += wordFreq
        scores.append(score)
    return scores
```

Language Sentiment Analysis: Deploy scikit-learn & Test!

In [470]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

X = df[['pos sent', 'tabloid word score', 'tabloid bigram score', 'misleading word score', 'misleading bigram score']]
Y = df[['type']]
```

In [471]:
```
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
kfold = model_selection.KFold(n_splits=50, random_state=17)
model = RandomForestClassifier(n_estimators=10)
results = model_selection.cross_val_score(model, X, Y.values.ravel(), cv=kfold)
print(results.mean())
```

```
0.8271428571428571
```

In [472]:
```
#####THIS IS OUR FINAL MODEL#####

from sklearn import svm

model = svm.SVC(kernel='linear', C=1, gamma=1)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y.values.ravel(), test_size=0.5)
model.fit(X_train, Y_train)
model.score(X_test, Y_test)
```

Out[472]: 0.885

Key Learnings and Takeaways

- An absence of text is usually a good way to determine REAL ads

- Two different forms of Malicious content can still successfully be developed
  - Misleading
  - Tabloid

Image Classification

- Overview of Steps
  - Retrain last layer of pretrained CNN called Mobile Net – Directly availably from Google

- Libraries Used
  - Tensorflow

Image Classification: Step 1

Download Tensorflow from the Pre-Trained Repository

```
Chennys-MacBook-Pro:/ Chenny$ cd ~/tensorflow-for-poets-2/
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ ls
CONTRIBUTING.md LICENSE         README.md       android         ios             scripts         tf_files
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ cd tf_files/
Chennys-MacBook-Pro:tf_files Chenny$ ls
bottlenecks                 models                      retrained_labels.txt
hackathon_training_images   retrained_graph.pb          training_summaries
Chennys-MacBook-Pro:tf_files Chenny$
```

Initialize Training of CNN on Newly Created Image

Image Classification: Step 2

Allows for Training Result Analysis in Real Time throughout process

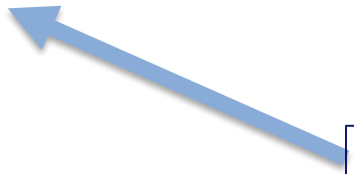
```
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ 2018-06-20 11:34:53.645892: I tensorflow/core/pla
tform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not c
ompiled to use: AVX2 FMA
TensorBoard 1.7.0 at http://Chennys-MacBook-Pro.local:6006 (Press CTRL+C to quit)
W0620 11:35:29.535485 Reloader plugin_event_multiplexer.py:203] Deleting accumulator 'mobilenet_0.50
_224/train'
W0620 11:35:29.536097 Reloader plugin_event_multiplexer.py:203] Deleting accumulator 'mobilenet_0.50
_224/validation'
```

Image Classification: Step 3 – Start Training!

```
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ IMAGE_SIZE=224
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ ARCHITECTURE="mobilenet_0.50_${IMAGE_SIZE}"
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ python -m scripts.retrain \
>   --bottleneck_dir=tf_files/bottlenecks \
>   --how_many_training_steps=500 \
>   --model_dir=tf_files/models/ \
>   --summaries_dir=tf_files/training_summaries/"${ARCHITECTURE}" \
>   --output_graph=tf_files/retrained_graph.pb \
>   --output_labels=tf_files/retrained_labels.txt \
>   --architecture="${ARCHITECTURE}" \
>   --image_dir=tf_files/hackathon_training_images/
INFO:tensorflow:Looking for images in 'Good Advertisements'
INFO:tensorflow:Looking for images in 'Misleading Claims'
INFO:tensorflow:Looking for images in 'Tabloid Advertorial'
WARNING:tensorflow:WARNING: Folder has less than 20 images, which may cause issues.
2018-06-20 11:36:09.544234: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
INFO:tensorflow:100 bottleneck files created.
INFO:tensorflow:200 bottleneck files created.
WARNING:tensorflow:From /Users/Chenny/tensorflow-for-poets-2/scripts/retrain.py:790: softmax_cross_entropy_with_logits (from tensorflow.python.ops.nn_ops) is deprecated and will be removed in a futu
re version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See tf.nn.softmax_cross_entropy_with_logits_v2.

INFO:tensorflow:2018-06-20 11:36:10.366231: Step 0: Train accuracy = 80.0%
INFO:tensorflow:2018-06-20 11:36:10.366512: Step 0: Cross entropy = 0.451494
INFO:tensorflow:2018-06-20 11:36:10.534022: Step 0: Validation accuracy = 81.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:10.850277: Step 10: Train accuracy = 60.0%
INFO:tensorflow:2018-06-20 11:36:10.850410: Step 10: Cross entropy = 4.833435
INFO:tensorflow:2018-06-20 11:36:10.880348: Step 10: Validation accuracy = 60.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:11.193087: Step 20: Train accuracy = 80.0%
INFO:tensorflow:2018-06-20 11:36:11.193227: Step 20: Cross entropy = 0.587889
INFO:tensorflow:2018-06-20 11:36:11.223784: Step 20: Validation accuracy = 50.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:11.538036: Step 30: Train accuracy = 75.0%
INFO:tensorflow:2018-06-20 11:36:11.538175: Step 30: Cross entropy = 0.727076
INFO:tensorflow:2018-06-20 11:36:11.568081: Step 30: Validation accuracy = 37.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:11.881056: Step 40: Train accuracy = 77.0%
INFO:tensorflow:2018-06-20 11:36:11.881197: Step 40: Cross entropy = 0.785316
INFO:tensorflow:2018-06-20 11:36:11.911201: Step 40: Validation accuracy = 53.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:12.226277: Step 50: Train accuracy = 81.0%
INFO:tensorflow:2018-06-20 11:36:12.226418: Step 50: Cross entropy = 0.217396
INFO:tensorflow:2018-06-20 11:36:12.257380: Step 50: Validation accuracy = 41.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:12.570065: Step 60: Train accuracy = 93.0%
INFO:tensorflow:2018-06-20 11:36:12.570206: Step 60: Cross entropy = 0.290409
INFO:tensorflow:2018-06-20 11:36:12.602743: Step 60: Validation accuracy = 48.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:12.915675: Step 70: Train accuracy = 97.0%
INFO:tensorflow:2018-06-20 11:36:12.915826: Step 70: Cross entropy = 0.117552
INFO:tensorflow:2018-06-20 11:36:12.945868: Step 70: Validation accuracy = 54.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:13.259343: Step 80: Train accuracy = 98.0%
INFO:tensorflow:2018-06-20 11:36:13.259481: Step 80: Cross entropy = 0.038130
INFO:tensorflow:2018-06-20 11:36:13.289796: Step 80: Validation accuracy = 54.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:13.601989: Step 90: Train accuracy = 96.0%
INFO:tensorflow:2018-06-20 11:36:13.602126: Step 90: Cross entropy = 0.103011
INFO:tensorflow:2018-06-20 11:36:13.632751: Step 90: Validation accuracy = 55.0% (N=100)
INFO:tensorflow:2018-06-20 11:36:13.945141: Step 100: Train accuracy = 98.0%
INFO:tensorflow:2018-06-20 11:36:13.945278: Step 100: Cross entropy = 0.063705
INFO:tensorflow:2018-06-20 11:36:13.975170: Step 100: Validation accuracy = 45.0% (N=100)
```

Image Classification: Step 4 – Result Analysis

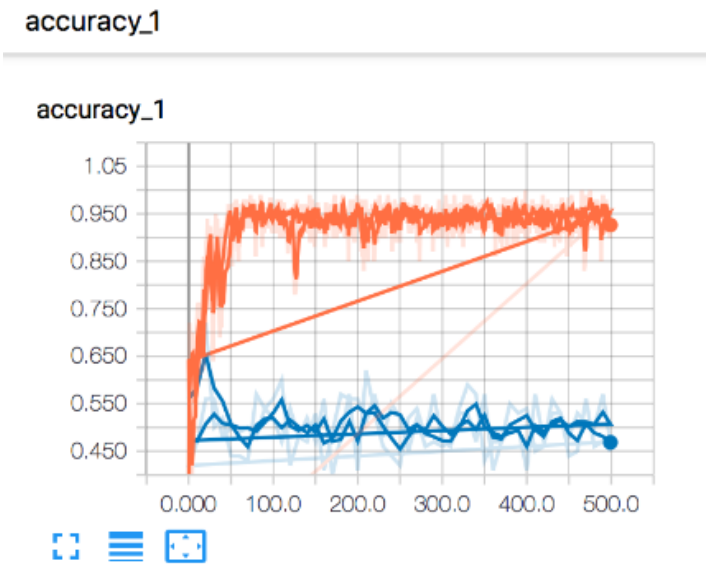
```
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ python -m scripts.label_image \
>     --graph=tf_files/retrained_graph.pb  \
[> python -m scripts.label_ima
usage: label_image.py [-h] [--image IMAGE] [--graph GRAPH] [--labels LABELS]
[                     [--input_height INPUT_HEIGHT]
                      [--input_width INPUT_WIDTH] [--input_mean INPUT_MEAN]
                      [--input_std INPUT_STD] [--input_layer INPUT_LAYER]
                      [--output_layer OUTPUT_LAYER]
Chennys-MacBook-Pro:tensorflow-for-poets-2 Chenny$ python -m scripts.label_image --image=tf_files/hackathon_t
raining_images/Misleading\ Claims/6.jpg
2018-06-20 11:45:39.813889: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructio
ns that this TensorFlow binary was not compiled to use: AVX2 FMA
[
[Evaluation time (1-image): 0.184s
[
[misleading claims (score=0.99995)
[good advertisements (score=0.00004)
[tabloid advertorial (score=0.00000)
```

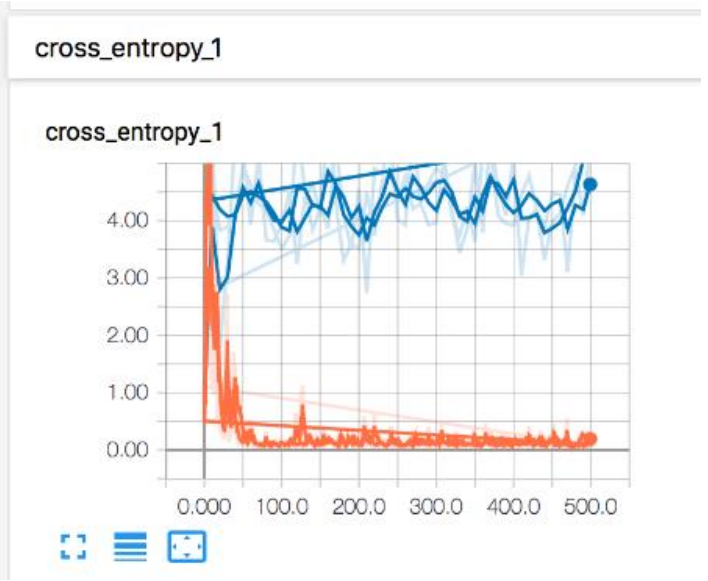


Initial Results for select images

Image Classification: Initial Results

- The **training accuracy** shows the percentage of the images used in the current training batch that were labeled with the correct class.
- **Validation accuracy**: The validation accuracy is the precision (percentage of correctly-labelled images) on a randomly-selected group of images from a different set.
- **Cross entropy** is a loss function that gives a glimpse into how well the learning process is progressing. (Lower numbers are better.)

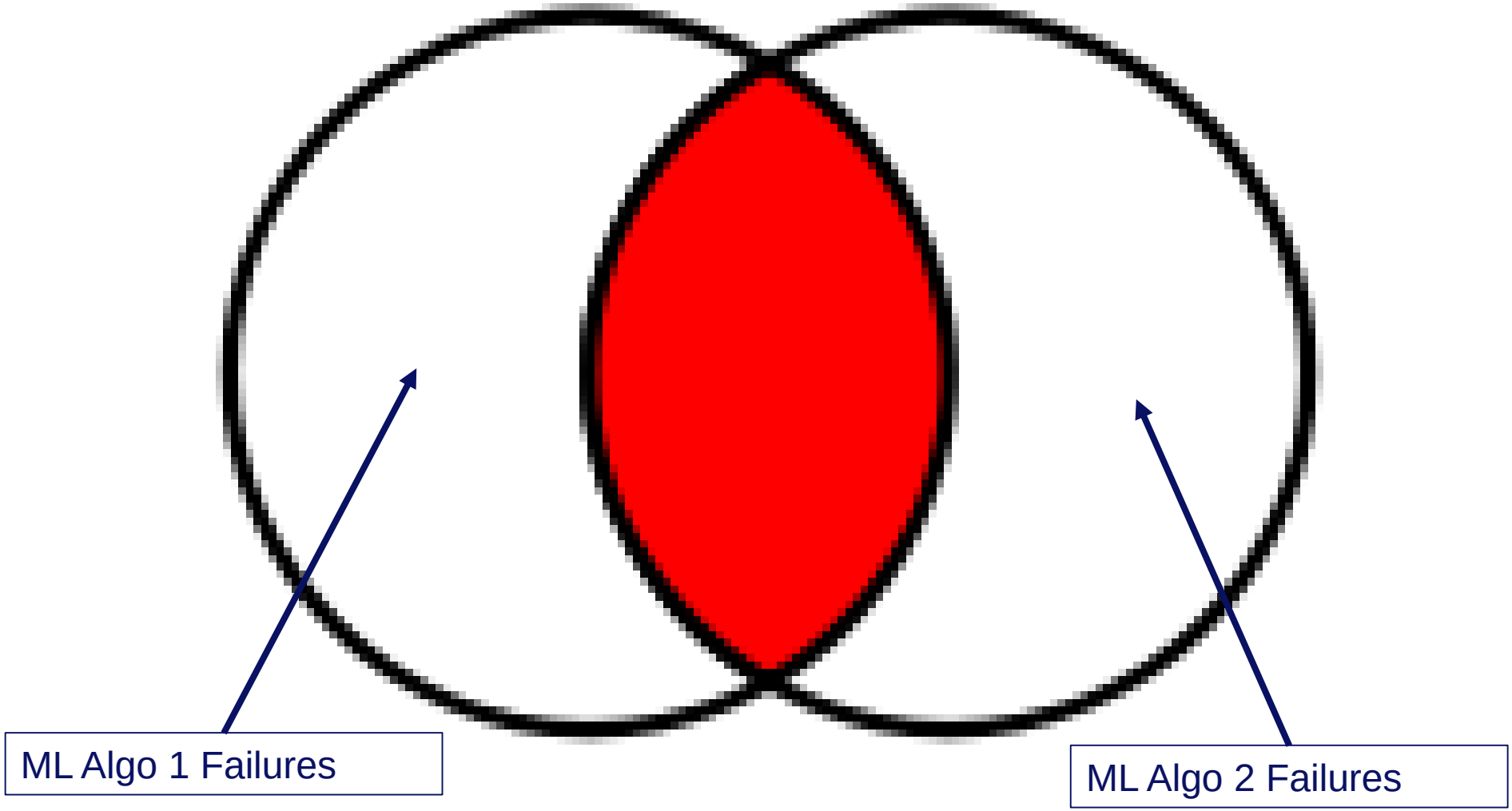




Agenda

- Project Introduction
- Details of completed steps
- **Next Steps for Early Bird**

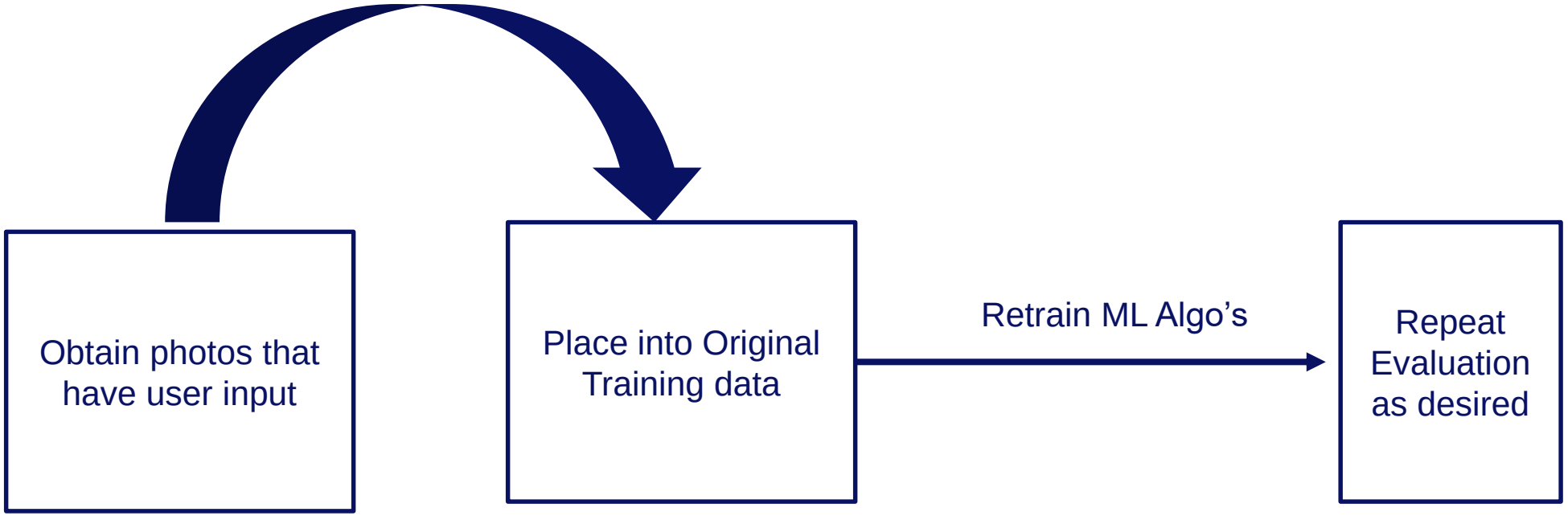
Core Next Step: Create sample for clients to Evaluate



Core Next Step: Create sample for clients to Evaluate

- Uses Random Sample - Why?
  - Key step: We do not want to introduce any biases based upon the metrics that we create. Sample should be statistically significant
- Insert these photos into the original dataset with the NEW marked MC content status, and retrain
- Test the algorithm on a new sample and allow analysis of the outputs

Core Next Step: Graphic



Summary of Accomplished Work

- Created Methodology for finding large amounts of Quality Advertisements
- Successfully Trained two independent ML framework
- Developed Extensive Next Step and Evaluation Steps