THE SUPER SEO GAME

Webinar

Powering up your Data Studio reports with a BigQuery NLP Pipeline

oncrawl

GAMEPLAY & RULES

- Earn points by signing up, attending, and participating
- Unlock new levels, earn badges and check our leaderboard
- Use #SuperSEOGame to continue the conversation
- Have fun!

SINGLE PLAYER of the day

JR Oakes

Senior Director, Technical SEO Research

LOCOMOTIVE

oncrawl

We will miss you Russ!

We will go on a journey to understand BigQuery ML.

BigQuery ML allows for the hosting of Bert language models, advanced forecasting, classification tasks, and many other use cases, all easily accessible with standard SQL queries. This talk will cover the various use-cases for BigQuery ML, with examples and code. You will learn how this can make your Data Studio reporting more actionable and understandable.

What I hope to accomplish

- Introduce BigQuery ML
- Provide a few examples to play with
- Become familiar with some building blocks
- Encourage you to try, fail, succeed, and share so we can all grow

PyTorch > TensorFlow

{

"@type": "Question",

"name": **What is BigQuery ML?**",

"acceptedAnswer": {

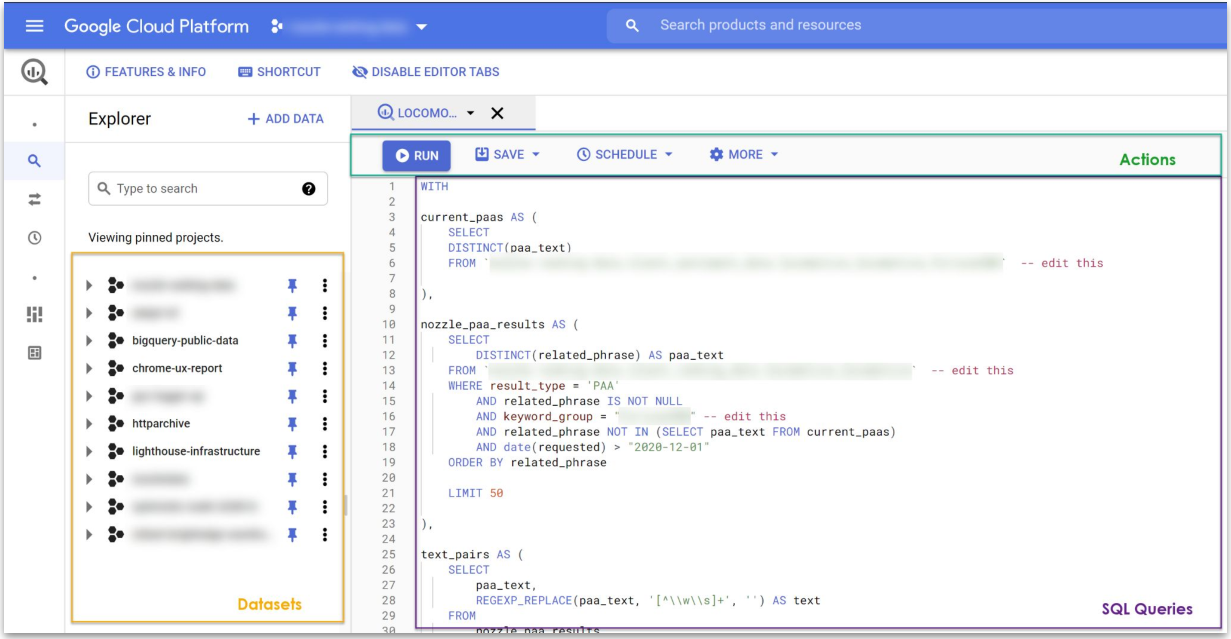
"@type": "Answer",

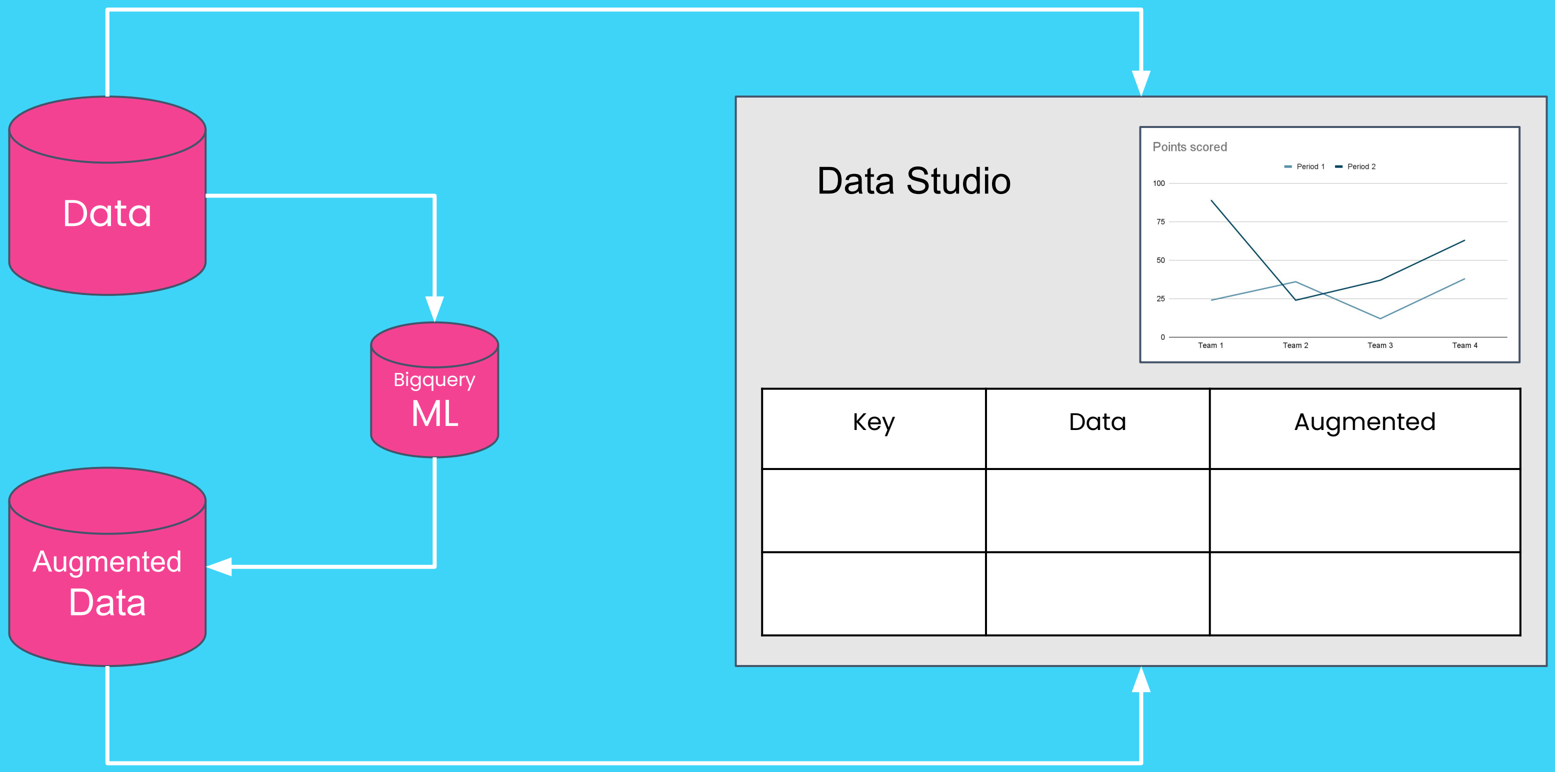
"text": "BigQuery ML lets you create and execute machine learning models in BigQuery using standard SQL queries. BigQuery ML democratizes machine learning by letting SQL practitioners build models using existing SQL tools and skills. **BigQuery ML increases development speed by eliminating the need to move data**."

}

}

oncrawl

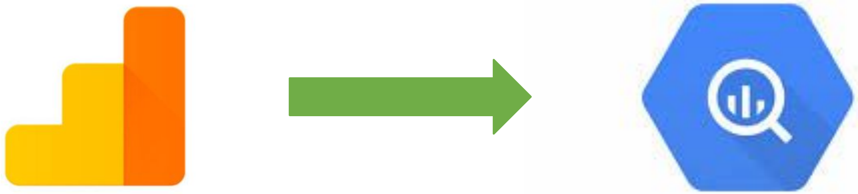

Google Cloud Platform
Search products and resources
FEATURES & INFO
SHORTCUT
DISABLE EDITOR TABS
Explorer
+ ADD DATA
Type to search
Viewing pinned projects.
bigquery-public-data
chrome-ux-report
httparchive
lighthouse-infrastructure
Datasets
LOCOMO...
RUN
SAVE
SCHEDULE
MORE
Actions
WITH
current_paas AS (
SELECT
DISTINCT(paa_text)
FROM ` -- edit this
),
nozzle_paa_results AS (
SELECT
DISTINCT(related_phrase) AS paa_text
FROM ` -- edit this
WHERE result_type = 'PAA'
AND related_phrase IS NOT NULL
AND keyword_group = "" -- edit this
AND related_phrase NOT IN (SELECT paa_text FROM current_paas)
AND date(requested) > "2020-12-01"
ORDER BY related_phrase
LIMIT 50
),
text_pairs AS (
SELECT
paa_text,
REGEXP_REPLACE(paa_text, '[^\\w\\s]+', '') AS text
FROM
SQL Queries


Data
Bigquery
ML
Augmented
Data
Data Studio
Points scored
Period 1
Period 2
100
75
50
25
0
Team 1
Team 2
Team 3
Team 4
Key
Data
Augmented
oncrawl

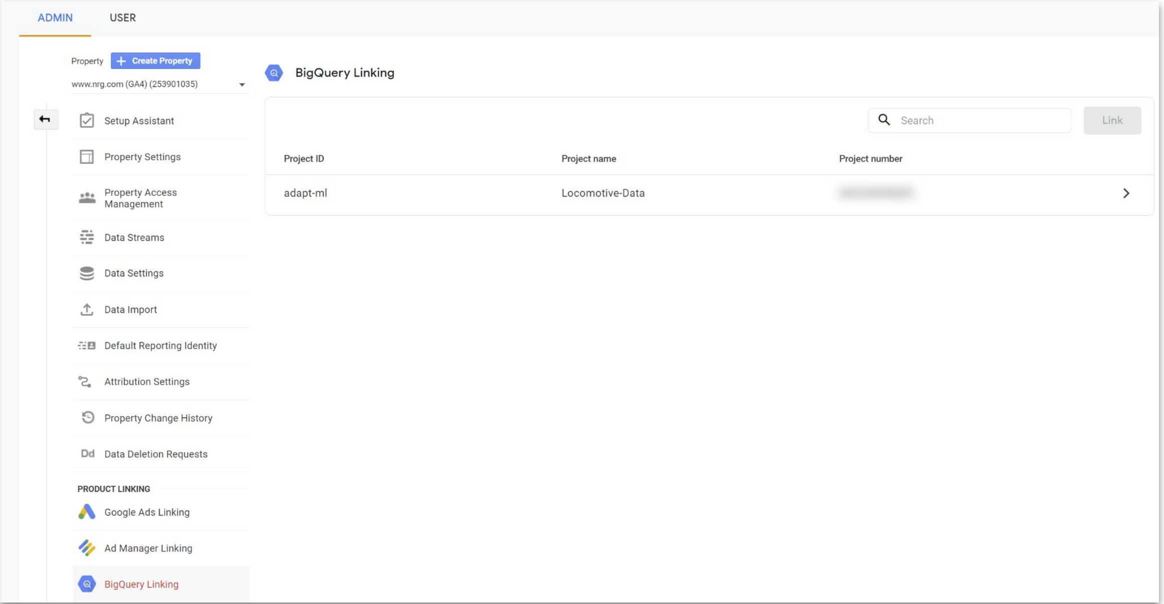
Type of Models

- Classification
- Clustering
- Product Recommendation
- Forecasting
- Anomaly Detection
- Custom TensorFlow Models

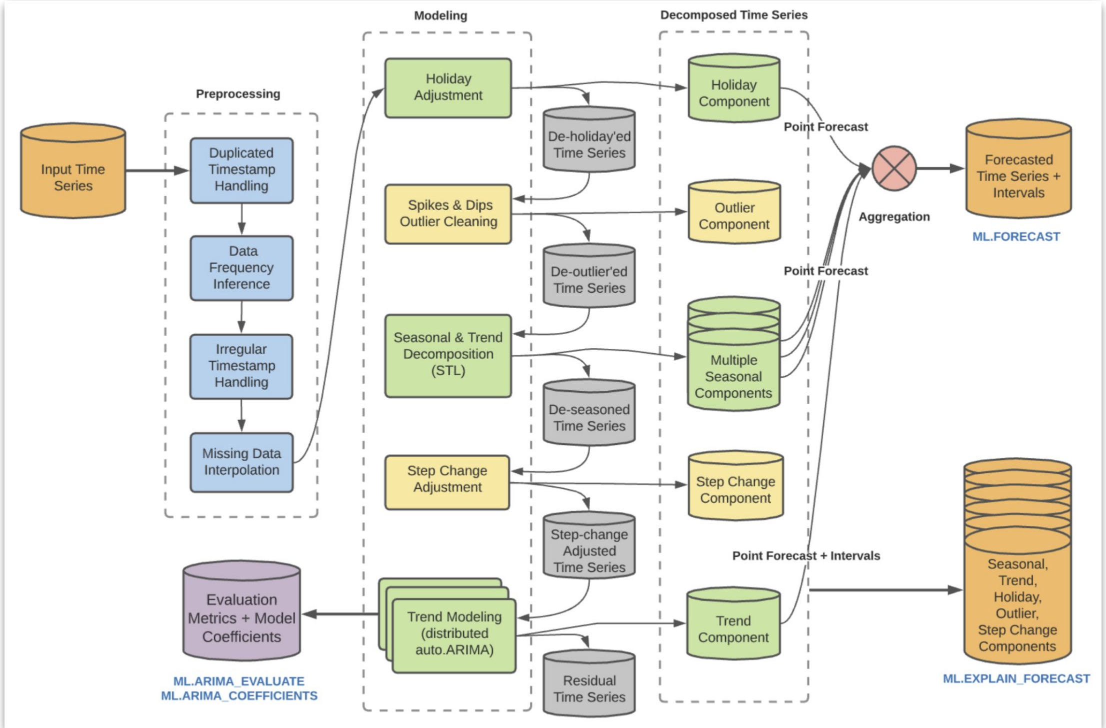
Forecasting



Google Analytics 4 has a direct connect to BigQuery



BigQuery ML handles all the difficult work



Build Arima model from GA4 data

```sql
-- Build ARIMA model
CREATE OR REPLACE MODEL `forecasting.forecast_client_name`
  OPTIONS(
    MODEL_TYPE='ARIMA_PLUS',
    TIME_SERIES_TIMESTAMP_COL='date',
    TIME_SERIES_DATA_COL='sessions',
    HOLIDAY_REGION='US'
  ) AS
  SELECT
    date,
    sessions,

  FROM
    (
      WITH raw_ga_4 AS (

        SELECT
        * except(row)
        FROM (
        SELECT
          -- extracts date from source table
          parse_date('%Y%m%d',regexp_extract(_table_suffix,'[0-9]+')) as table_date,
          -- flag to indicate if source table is `events_intraday_`
          case when _table_suffix like '%intraday%' then true else false end as is_intraday,
          *,
          row_number() over (partition by user_pseudo_id, event_name, event_timestamp order by event_timestamp) as row
        FROM
          `adapt-ml.analytics_XXXXXXXXXXXXXX.events_*`

          )
        WHERE
        row = 1

        ),

      pageviews AS (
        SELECT
          parse_date("%Y%m%d", event_date) event_date,
          event_timestamp,
          user_pseudo_id,
          user_first_touch_timestamp,
          device.category as device_category,
          device.language as device_language,
          device.web_info.browser as device_browser,
          geo.continent as geo_continent
```

[Code](#)

Output 30-day prediction

```sql
-- Forecast 30 days

SELECT
  *
FROM
  ML.EXPLAIN_FORECAST(MODEL forecasting.forecast_client_name,
                      STRUCT(30 AS horizon, 0.9 AS confidence_level))
;
```

[Code]

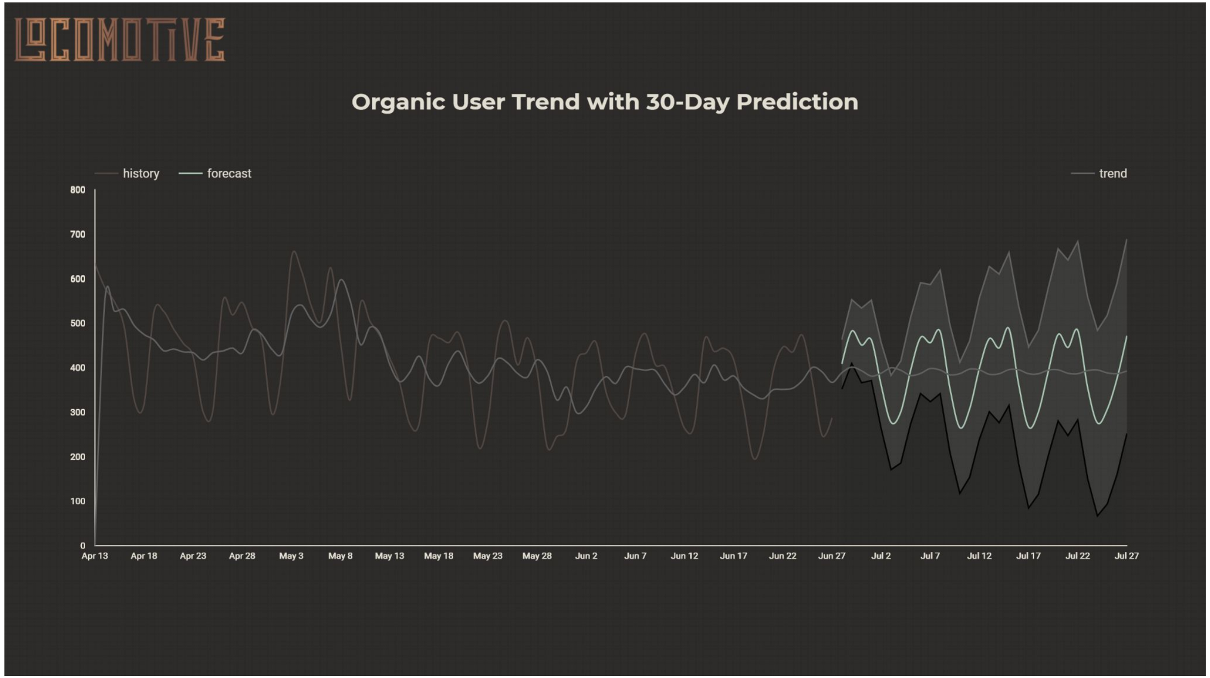
Save as scheduled query

+ CREATE SCHEDULED QUERY

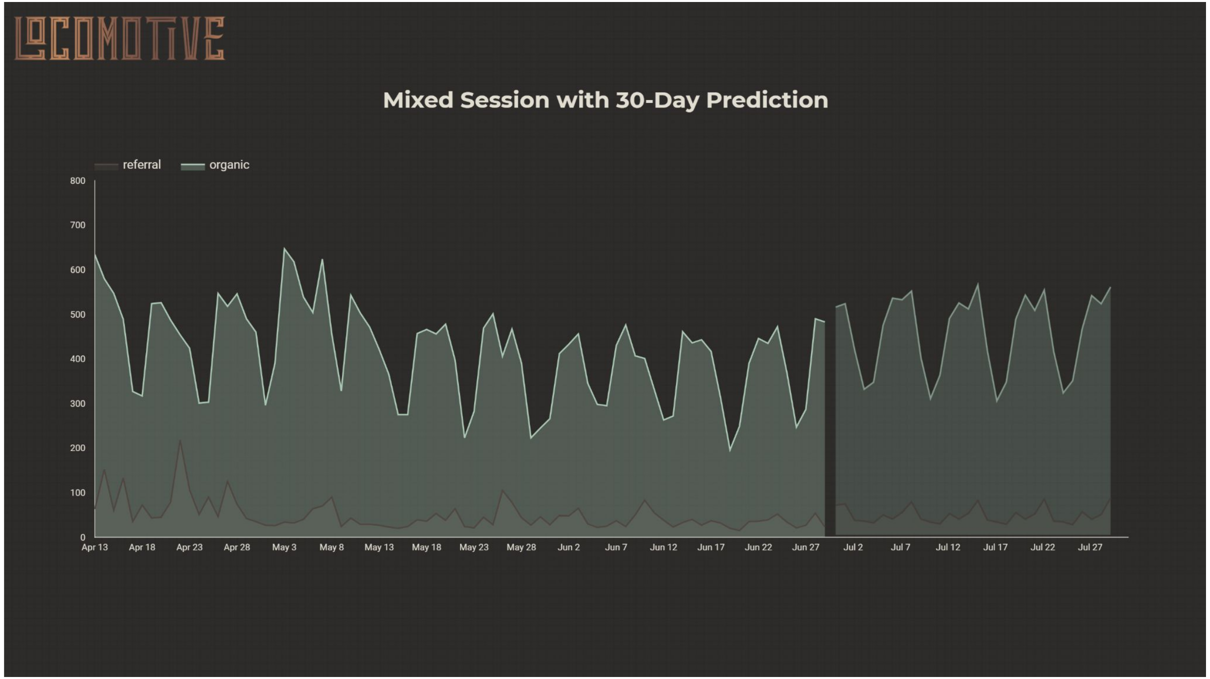
SCHEMA DETAILS PREVIEW

Row	time_series_timestamp	time_series_type	time_series_data	time_series_adjusted_data	standard_error	confidence_level	prediction_interval_lower_bound	prediction_interval_upper_bound
77	2021-06-28 00:00:00 UTC	forecast	408.02564333716197	408.02564333716197	33.927575290793456	0.9	352.2802235205583	463.7710643222681
78	2021-06-29 00:00:00 UTC	forecast	481.83813312835343	481.83813312835343	43.606292749833258	0.9	410.18989047899231	553.48637577771456
79	2021-06-30 00:00:00 UTC	forecast	450.53942578912228	450.53942578912228	51.133864089163353	0.9	366.5228476572003 2	534.55600392104429
80	2021-07-01 00:00:00 UTC	forecast	461.93267314470484	461.93267314470484	55.070473176451507	0.9	371.44796609570631	552.41738019370337
81	2021-07-02 00:00:00 UTC	forecast	359.71005261841543	359.71005261841543	58.834275553663055	0.9	263.04115037644846	456.37895486038241
82	2021-07-03 00:00:00 UTC	forecast	276.99586490642906	276.99586490642906	64.568571622877 4	0.9	170.90510608639016	383.08662372646796
83	2021-07-04 00:00:00 UTC	forecast	300.74493894698992	300.74493894698992	69.88491952423017	0.9	185.91904213597832	415.57083575800152
84	2021-07-05 00:00:00 UTC	forecast	394.22570317017295	394.22570317017295	73.029034155257534	0.9	274.2338022392143	514.21760410113154
85	2021-07-06 00:00:00 UTC	forecast	467.128395216567	467.128395216567	75.912480307133279	0.9	342.39878702988784	591.85800340324624
86	2021-07-07 00:00:00 UTC	forecast	455.67242432549006	455.67242432549006	80.209064882939089	0.9	323.88322187298064	587.46162677799953
87	2021-07-08 00:00:00 UTC	forecast	480.99854671897128	480.99854671897128	84.528087240889576	0.9	342.11288312756733	619.88421031037524
88	2021-07-09 00:00:00 UTC	forecast	352.46624341120895	352.46624341120895	87.307913322620479	0.9	209.01312771445166	495.91935910796622
89	2021-07-10 00:00:00 UTC	forecast	265.01871206670927	265.01871206670927	89.756723083775711	0.9	117.54202763926585	412.49539649415271
90	2021-07-11 00:00:00 UTC	forecast	307.22089149638043	307.22089149638043	93.251286852647 8	0.9	154.0023900724263 2	460.43939292033451

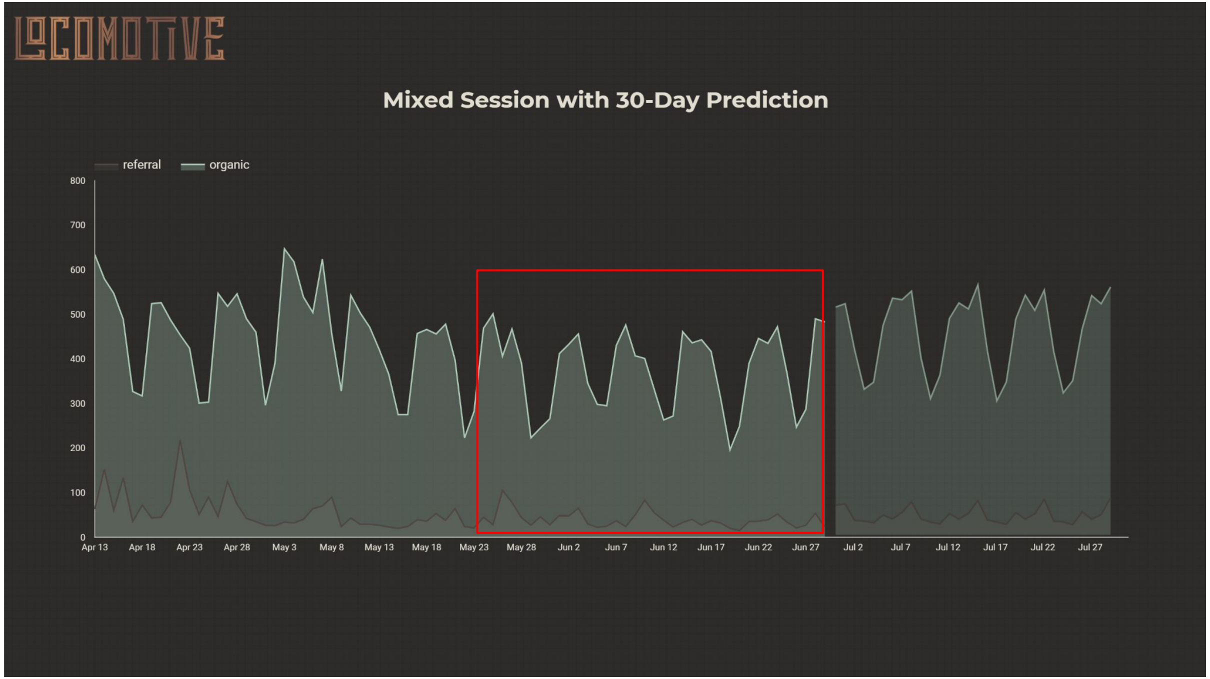
Add predictive visuals to your Data Studio reports



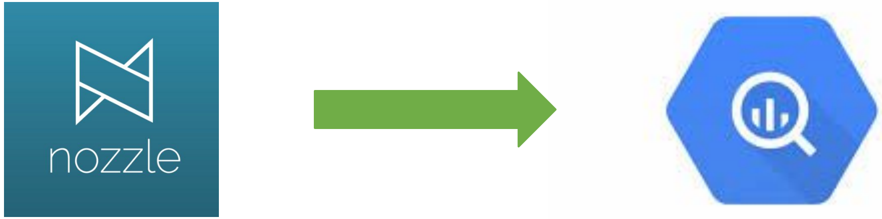
Add predictive visuals to your Data Studio reports



Predict historical to look for missed predictions



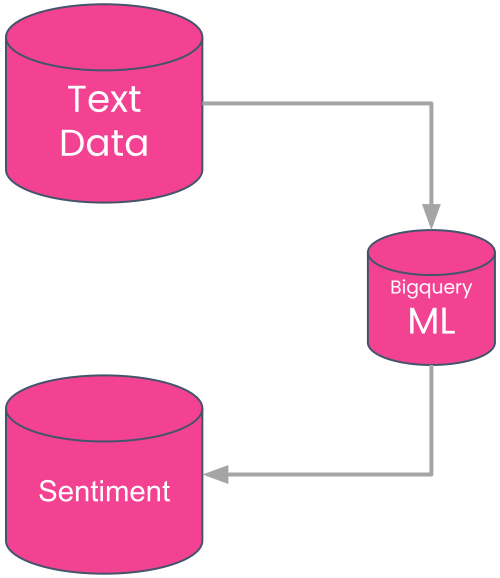
Classification


nozzle

Nozzle gives you REALLY granular SERP data

Row	requested	adwords_search_volume	adwords_cpc	phrase	device	engine	language	result_url_domain	result_type	related_phrase
1	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	collegeconfidential.com	Organic	null
2	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	collegeconfidential.com	Sitelink	null
3	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	collegeconfidential.com	Sitelink	null
4	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	collegeconfidential.com	Sitelink	null
5	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	collegeconfidential.com	Sitelink	null
6	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	google.com	Sitelink	null
7	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	reddit.com	Organic	null
8	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	reddit.com	Sitelink	null
9	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	reddit.com	Sitelink	null
10	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	reddit.com	Sitelink	null
11	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	reddit.com	Sitelink	null
12	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	google.com	Sitelink	null
13	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	quora.com	Organic	null
14	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	quora.com	Sitelink	null
15	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	quora.com	Sitelink	null
16	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	quora.com	Sitelink	null
17	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	quora.com	Sitelink	null
18	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	google.com	Sitelink	null
19	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	null	Video	null
20	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	youtube.com	Video	null
21	2020-07-15 00:00:00 UTC	260	0.00106433	columbia waitlist	desktop	google	English	youtube.com	Video	null

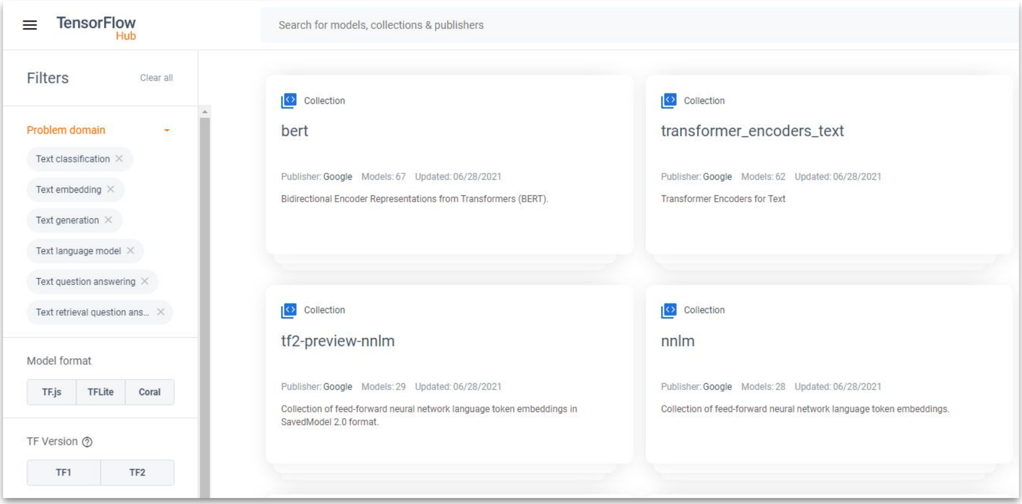
Would be cool to turn it into a brand sentiment tool



The Problem

250 MB

TensorFlow Hub hosts models to turn text into numbers



Even a bunch of BERT models

BERT encoder model	preprocessing for use with it
tensorflow/bert_en_uncased_L-12_H-768_A-12	tensorflow/bert_en_uncased_preprocess
tensorflow/bert_en_uncased_L-24_H-1024_A-16	tensorflow/bert_en_uncased_preprocess
tensorflow/bert_en_wwm_uncased_L-24_H-1024_A-16	tensorflow/bert_en_uncased_preprocess
tensorflow/bert_en_cased_L-12_H-768_A-12	tensorflow/bert_en_cased_preprocess
tensorflow/bert_en_cased_L-24_H-1024_A-16	tensorflow/bert_en_cased_preprocess
tensorflow/bert_en_wwm_cased_L-24_H-1024_A-16	tensorflow/bert_en_cased_preprocess
tensorflow/bert_zh_L-12_H-768_A-12	tensorflow/bert_zh_preprocess
tensorflow/bert_multi_cased_L-12_H-768_A-12	tensorflow/bert_multi_cased_preprocess

I can just build my own model and host on BigQuery

- Matrix Factorization for creating product recommendation systems. You can create product recommendations using historical customer behavior, transactions, and product ratings and then use those recommendations for personalized customer experiences.
- Time series for performing time-series forecasts. You can use this feature to create millions of time series models and use them for forecasting. The model automatically handles anomalies, seasonality, and holidays.
- Boosted Tree for creating XGBoost based classification and regression models.
- Deep Neural Network (DNN) for creating TensorFlow-based Deep Neural Networks for classification and regression models.
- AutoML Tables to create best-in-class models without feature engineering or model selection. AutoML Tables searches through a variety of model architectures to decide the best model.
- TensorFlow model importing. This feature lets you create BigQuery ML models from previously trained TensorFlow models, then perform prediction in BigQuery ML.
- Autoencoder for creating Tensorflow-based BigQuery ML models with the support of sparse data representations. The models can be used in BigQuery ML for tasks such as unsupervised anomaly detection and non-linear dimensionality reduction.

Component Parts of Model

```
def build_classifier_model(train_dataset):
  text_input = tf.keras.layers.Input(shape=[], dtype=tf.string, name='text')
  preprocessing_layer = hub.KerasLayer(PREPROCESSOR_NAME, name='preprocessing', )
  encoder_inputs = preprocessing_layer(text_input)
  encoder = hub.KerasLayer(MODEL_NAME, trainable=True, name='BERT_encoder', )
  outputs = encoder(encoder_inputs)

  net = outputs['pooled_output']

  net = tf.keras.layers.Dense(
            int(PRECLASSIFIER_DIMS),
            kernel_initializer=tf.keras.initializers.TruncatedNormal(stddev=0.002),
            activation="relu",
            name="pre_classifier"
        )(net)

  net = tf.keras.layers.Dropout(DROPOUT)(net)
  net = tf.keras.layers.Dense(2, activation="sigmoid", use_bias=True, name='classifier')(net)

  model = tf.keras.Model(text_input, net, name='sentiment_classification')
```

TF Hub

Classifier Parts

Model

Full end-to-end notebook for training your own sentiment model and uploading to BigQuery

```
Model: "sentiment_classification"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
text (InputLayer)               [(None,)]            0
__________________________________________________________________________________________________
preprocessing (KerasLayer)      {'input_word_ids': ( 0           text[0][0]
__________________________________________________________________________________________________
BERT_encoder (KerasLayer)       {'sequence_output':  11170561    preprocessing[0][0]
                                                                 preprocessing[0][1]
                                                                 preprocessing[0][2]
__________________________________________________________________________________________________
pre_classifier (Dense)          (None, 128)          32896       BERT_encoder[0][5]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           pre_classifier[0][0]
__________________________________________________________________________________________________
classifier (Dense)              (None, 2)            258         dropout_2[0][0]
==================================================================================================
Total params: 11,203,715
Trainable params: 11,203,714
Non-trainable params: 1
```

Code   Article

Training Data Options

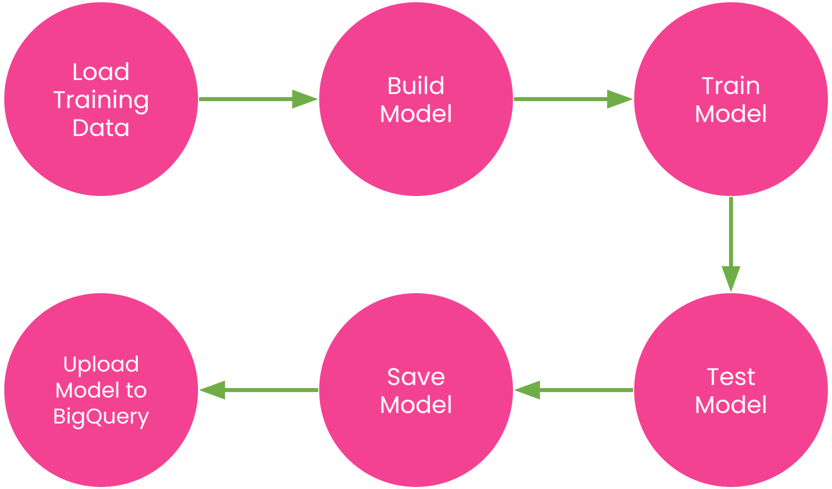
SST-2

The Stanford Sentiment Treebank SST-2 dataset contains 215,154 phrases with fine-grained sentiment labels in the parse trees of 11,855 sentences from movie reviews. Models performances are evaluated either based on a fine-grained (5-way) or binary classification model based on accuracy. (source)

IMDb

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. (source)

Steps in the Colab Notebook



Sentiment model available when needed

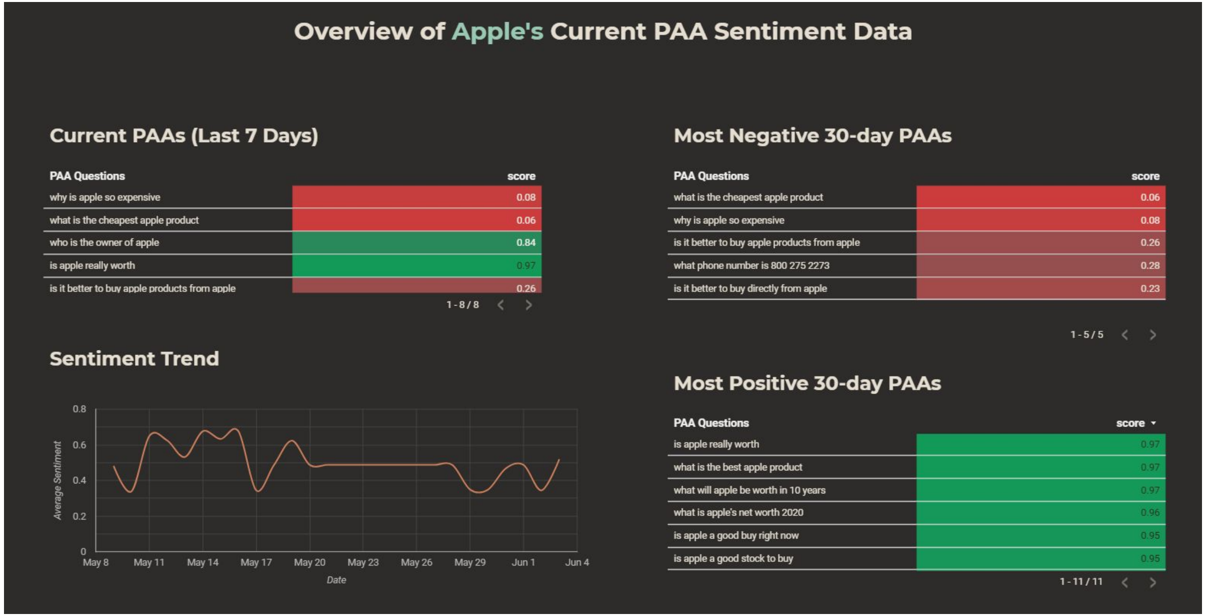
```sql
predictions AS (
    SELECT
    paa_text,
    ROUND(classifier[OFFSET(1)], 3) as score
    FROM ML.PREDICT(MODEL `          .bigquery_ml.sentiment`,
                    (SELECT REGEXP_REPLACE(paa_text, '[^\\w\\s]+', '') AS text FROM nozzle_paa_results))
    JOIN (SELECT paa_text, REGEXP_REPLACE(paa_text, '[^\\w\\s]+', '') AS text FROM nozzle_paa_results) USING (text)
    ORDER BY score ASC


)

SELECT * FROM predictions;
```

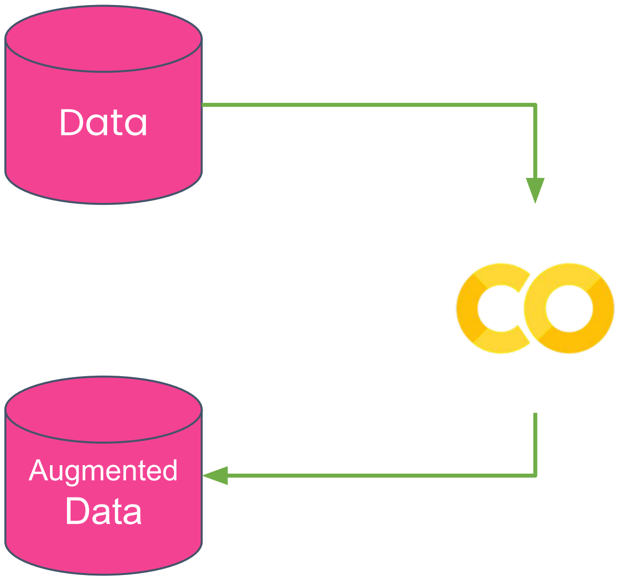
Row	paa_text	score
1	why is ge stock so bad?	0.03
2	why is comcast so bad?	0.031
3	why is allstate so bad?	0.031
4	why was anthem so bad?	0.031
5	why is frontier internet so bad?	0.031
6	why is american express so bad?	0.031
7	why is anthem so bad?	0.031
8	why is centurylink so bad?	0.031
9	is dollar general a bad place to work?	0.031
10	why is apple so bad?	0.032
11	why is anthem game so bad?	0.032
12	is frontier communications stock worthless?	0.032
13	is microsoft bad?	0.032
14	is hp a bad laptop?	0.032
15	why is starbucks so bad?	0.032

Allows for highlighting important trends



Clustering

Some things are difficult in TensorFlow



TFHub converts text into n-dimensonal pooled outputs

```python
self.model = model
self.preprocess = hub.KerasLayer(self.map_model_to_preprocess[model],
                                 name='preprocess' )
self.encoder = hub.KerasLayer(self.map_name_to_model[model],
                              name=model )
```

```
[16] 1  cluster.get_encoded(['export more than 5000 rows google analytics'])['pooled_output']

    <tf.Tensor: shape=(1, 512), dtype=float32, numpy=
    array([[ 4.31210071e-01,  3.72204751e-01,  4.32841897e-01,
             7.99269855e-01,  5.91591954e-01, -4.85236228e-01,
            -5.35845697e-01, -6.24034166e-01,  5.05522072e-01,
             9.96166408e-01, -4.18141454e-01, -8.83240640e-01,
             6.15609944e-01, -2.68304348e-01, -6.27349734e-01,
             8.65133822e-01, -9.96641695e-01,  9.94911909e-01,
             1.09758243e-01, -9.91118193e-01,  8.75520289e-01,
             2.58417249e-01,  3.22697222e-01, -1.72956213e-02,
            -2.35691980e-01,  7.93872168e-04,  2.05544695e-01,
             3.98988545e-01,  5.89602590e-01,  2.64404029e-01,
             1.16417259e-01,  2.54679620e-01,  4.16897982e-01,
             3.29535037e-01, -2.40083560e-01, -9.69020963e-01,
             1.09754995e-01, -3.35454792e-01,  8.95521700e-01,
            -8.41044486e-01, -9.82097983e-02, -4.03535336e-01,
            -2.88857818e-01, -2.32576534e-01, -6.16090238e-01,
             6.53401017e-01,  2.10129172e-01, -1.98760211e-01,
             8.97522628e-01, -5.93671322e-01,  2.25686450e-02,
             9.81816351e-01, -1.60790216e-02,  9.97705579e-01,
```

Umap reduces to 2D while keeping much of the semantic information

```
[17] encoded = cluster.get_encoded(df['Top queries'].tolist()[:100])['pooled_output']
```

```
reduced = cluster.get_reduced(encoded,
                              neighbors=20)
reduced
```

```
array([[ 9.061797 , 13.931413 ],
       [ 7.922755 , 13.170953 ],
       [ 5.070562 , 15.17328  ],
       [ 8.218638 , 15.458736 ],
       [ 7.167097 , 13.543527 ],
       [ 5.4804506, 14.035787 ],
       [ 5.8768635, 14.676909 ],
       [ 4.8549867, 16.21587  ],
       [ 7.919964 , 13.961714 ],
       [ 7.300527 , 13.539764 ],
       [ 7.98836  , 15.114539 ],
       [ 4.740223 , 14.414347 ],
       [ 7.161938 , 16.11412  ],
       [ 6.429485 , 15.984068 ],
       [ 5.1955585, 14.296892 ],
       [ 7.2378354, 13.65459  ],
       [ 6.612344 , 13.720347 ],
```

HDBScan takes the 2D vectors and creates the right number of clusters

```
labels  = cluster.get_clusters(reduced,
                               min_samples=2,
                               min_cluster_size=5)

labels
```

```
array([ 5, -1,  3, -1,  0,  3,  3,  3, -1,  0, -1,  3,  1,  1,  3,  0,  0,
        5,  3,  3,  6,  1,  2,  3,  0, -1,  0,  3,  3,  0,  3,  1,  5,  3,
        3,  1,  6,  1,  3,  1,  1,  3,  1,  4, -1,  3,  1,  5, -1,  4,  0,
        3,  4,  0,  3, -1,  3, -1,  3,  3,  3,  4,  0,  3,  4, -1,  4,  4,
       -1,  0,  1,  2, -1,  2,  1, -1,  2,  3,  4,  3,  3,  0,  4,  3,  6,
        6,  3,  1,  6,  6,  2,  3,  5,  3,  3, -1,  1,  3,  2, -1])
```

Not having to pre-specify the number of clusters is the key benefit of HDBScan over K-Means

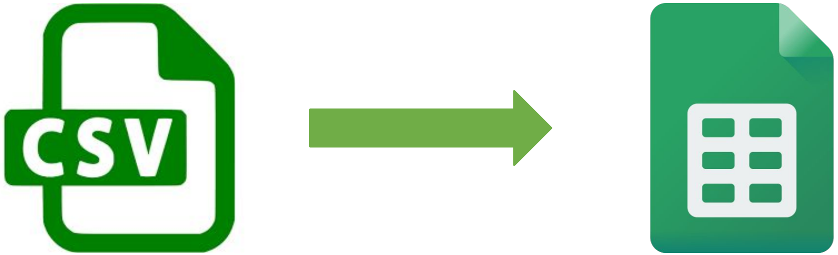
Apply category data back to original data

	Top queries	Clicks	Impressions	CTR	Position	labels	labels_text
0	locomotive agency	240	1173	0.2046	1.81	4	locomotive agency
1	google analytics export more than 5000	67	417	0.1607	1.02	1	google analytics export more than 5000
2	export more than 5000 rows google analytics	67	141	0.4752	1.09	2	export more than 5000 rows google analytics
3	locomotive seo	56	92	0.6087	1.34	0	locomotive seo
4	google analytics export more than 5000 rows	56	90	0.6222	1.00	1	google analytics export more than 5000
5	google analytics export all rows	53	129	0.4109	2.32	5	google analytics export all rows
6	how to export more than 5000 rows in google an...	47	98	0.4796	1.45	5	google analytics export all rows
7	google analytics export full report	13	484	0.0269	4.56	2	export more than 5000 rows google analytics
8	adapt partners	11	190	0.0579	1.90	4	locomotive agency
9	datadog export more than 5000	10	65	0.1538	4.78	5	google analytics export all rows

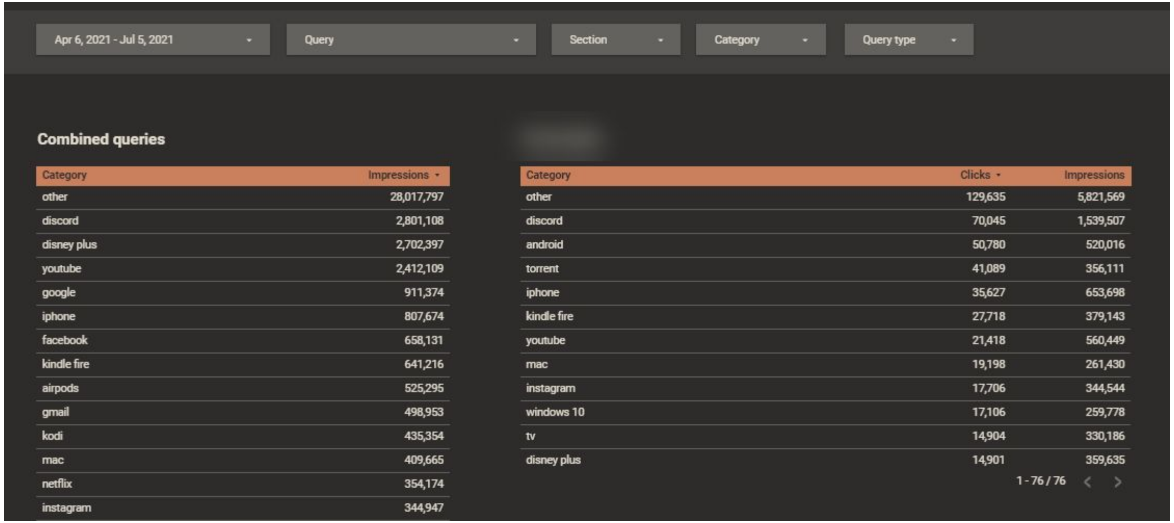
[Code]

~ 2 hours to clean data and adjust category labels

Upload CSV to Google Sheets



Merge category data in Data Studio



Apr 6, 2021 - Jul 5, 2021 | Query | Section | Category | Query type

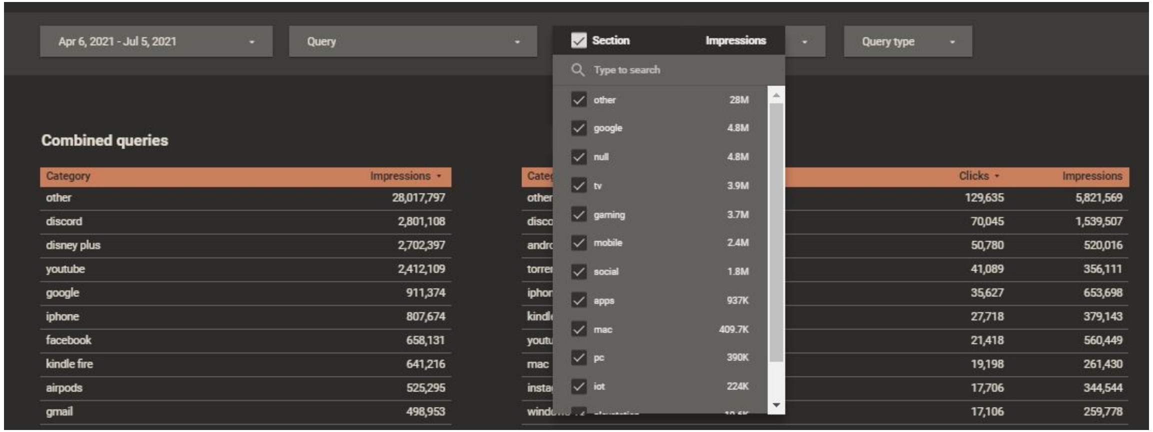
Combined queries

Category	Impressions
other	28,017,797
discord	2,801,108
disney plus	2,702,397
youtube	2,412,109
google	911,374
iphone	807,674
facebook	658,131
kindle fire	641,216
airpods	525,295
gmail	498,953
kodi	435,354
mac	409,665
netflix	354,174
instagram	344,947

Category	Clicks	Impressions
other	129,635	5,821,569
discord	70,045	1,539,507
android	50,780	520,016
torrent	41,089	356,111
iphone	35,627	653,698
kindle fire	27,718	379,143
youtube	21,418	560,449
mac	19,198	261,430
instagram	17,706	344,544
windows 10	17,106	259,778
tv	14,904	330,186
disney plus	14,901	359,635

1 - 76 / 76

Filter all GSC data by Section, Category, and Type



`https?://[^/]+/([^/]+)`

Extension

```sql
-- Euclidean squared distance

CREATE TEMPORARY FUNCTION td(a ARRAY<FLOAT64>, b ARRAY<FLOAT64>, idx INT64) AS (
   (a[OFFSET(idx)] - b[OFFSET(idx)]) * (a[OFFSET(idx)] - b[OFFSET(idx)])
);
CREATE TEMPORARY FUNCTION term_distance(a ARRAY<FLOAT64>, b ARRAY<FLOAT64>) AS ((
   SELECT SQRT(SUM( td(a, b, idx))) FROM UNNEST(GENERATE_ARRAY(0, 19)) idx
));

SELECT
    term_distance(arr1,arr2) as same,
    term_distance(arr3,arr4) as different,

FROM (
SELECT
    (SELECT encoder FROM ML.PREDICT(MODEL `adapt-ml.bigquery_ml.embedding_model`,(SELECT 'the cat is crazy'AS text))) AS arr1,
    (SELECT encoder FROM ML.PREDICT(MODEL `adapt-ml.bigquery_ml.embedding_model`,(SELECT 'the cat is crazy' AS text))) AS arr2,
    (SELECT encoder FROM ML.PREDICT(MODEL `adapt-ml.bigquery_ml.embedding_model`,(SELECT 'to be or not to be'AS text))) AS arr3,
    (SELECT encoder FROM ML.PREDICT(MODEL `adapt-ml.bigquery_ml.embedding_model`,(SELECT 'windows was developed by bill gates' AS text))) AS arr4
    )
```

Query results   SAVE RESULTS   EXPLORE DATA

Query complete (10.9 sec elapsed, 48.6 MB processed)

Job information   Results   JSON   Execution details

Row	same	different
1	0.0	1.7741975061302246

[Code](#)

Clustering in BigQuery

Problem:
No way to separate posts by performance groupings

/blog/post-id-247748378

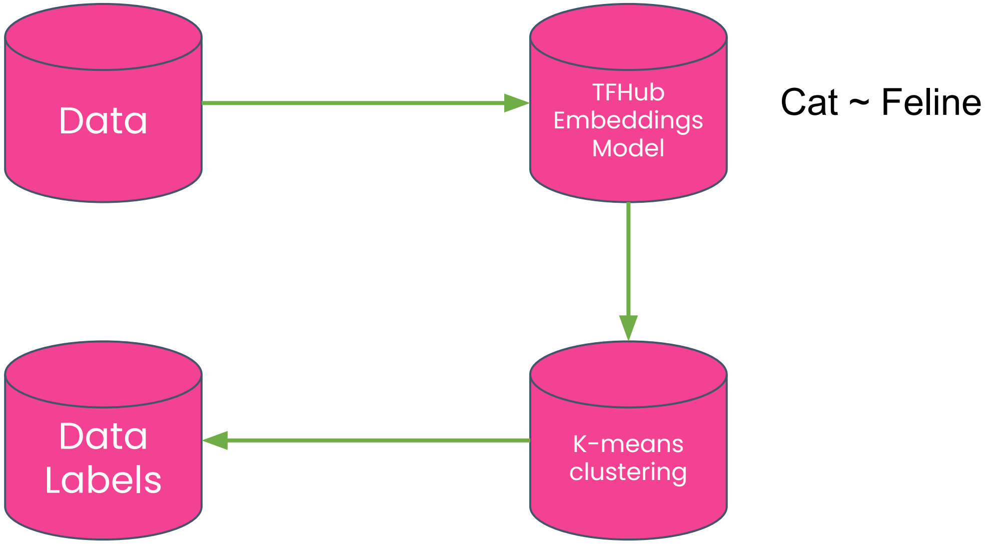
/blog/post-id-258398959

/blog/post-id-189489937

/blog/post-id-387588274

...

Structure


Data
TFHub Embeddings Model
Cat ~ Feline
K-means clustering
Data Labels

Use TensorFlow Hub to create an embedding model

```
tfhub_handle_encoder = map_name_to_handle[MODEL_NAME]
tfhub_handle_preprocess = map_model_to_preprocess[MODEL_NAME]

def build_embedding_model():
  text_input = tf.keras.layers.Input(shape=[], dtype=tf.string, name='text')
  preprocessing_layer = hub.KerasLayer(tfhub_handle_preprocess, name='preprocess', )
  encoder_inputs = preprocessing_layer(text_input)
  encoder = hub.KerasLayer(tfhub_handle_encoder, trainable=True, name='encoder', )
  outputs = encoder(encoder_inputs)
  net = outputs['pooled_output']
  model = tf.keras.Model(text_input, net)

  model.summary()

  return model


model = build_embedding_model()
```

Upload to BigQuery ML

```
client = bigquery.Client(project=PROJECT_ID, location="US")
dataset = client.create_dataset('bigquery_ml', exists_ok=True)

```

Initialize model with BigQuery

```
%%bigquery

CREATE OR REPLACE MODEL `[illegible].bigquery_ml.embedding_model`
OPTIONS (MODEL_TYPE='TENSORFLOW',  MODEL_PATH='gs://[illegible]/embedding_model/*')


```

Create K-Means model with embeddings

```
CREATE OR REPLACE MODEL `bigquery_ml.title_test`
OPTIONS(model_type='kmeans',
        num_clusters = 5,
        DISTANCE_TYPE = 'cosine',
        kmeans_init_method = 'KMEANS++') AS
(

    WITH
    raw_ga_4 AS (

        SELECT
        * except(row)
        FROM (
        SELECT
            -- extracts date from source table
            parse_date('%Y%m%d',regexp_extract(_table_suffix,'[0-9]+')) as table_date,
            -- flag to indicate if source table is `events_intraday_`
            case when _table_suffix like '%intraday%' then true else false end as is_intr
            *,
            row_number() over (partition by user_pseudo_id, event_name, event_timestamp o
        FROM
            `adapt-ml.analytics_266065389.events_*`

            )
        WHERE
        row = 1

        ),
```

Code

oncrawl

Save labeled data to new table

```
-- bigquery_ml.title_test
SELECT text, CENTROID_ID, FROM
ML.PREDICT(MODEL `bigquery_ml.title_test`,
(
    SELECT * FROM arrays
))
WHERE CENTROID_ID = 5
```

Query results   SAVE RESULTS   EXPLORE DATA

Query complete (15.9 sec elapsed, 215.1 MB processed)

Job information   Results   JSON   Execution details

Row	text	CENTROID_ID
1	Immersive Learning Engagement opportunity Inperson	5
2	[illegible] is an Assistant Professor of Communication and an advisor to [illegible]	5
3	Delta Kappa Gamma Provide Scholarship to WPU Students [illegible]	5
4	[illegible] Liberal Arts in [illegible]	5
5	School of Professional Studies NonTraditional 2021	5
6	Accelerated RN to BSN Nursing Program [illegible]	5
7	Accelerated online degree programs at [illegible]	5
8	Academic Majors Degrees Programs of Study [illegible]	5
9	Accelerated Bachelors and Licensure in Education Programs	5

Areas left to explore

- Product Recommendation
- Anomaly Detection in Traffic and Rankings
- SERP Title Sentiment
- Query entity extraction
- Question answering

Please Share your Code!

Grab your controller, it's time for Q&A!

oncrawl