



## Session Report

LLM でさらなる進化を実現  
BigQuery が叶える新たなデータ分析体験

# BigQuery と PaLM で進化する データ分析

Google Cloud  
カスタマー エンジニア  
データ アナリティクス スペシャリスト  
村上 祐磨

## セッションレポート概要

技術の進化によって、現在では BigQuery のデータに最先端の AI を簡単に適用できるようになりました。今回は、BigQuery 搭載の機械学習機能を紹介したうえで、Vertex AI の大規模言語モデル (LLM) を BigQuery から直接呼び出し、エンベディングやベクトル検索など高度なユースケースで手軽に活用する方法について紹介します。

## プレゼンター紹介



Google Cloud  
カスタマー エンジニア  
データ アナリティクス スペシャリスト  
村上 祐磨

メーカー R&D にて機械学習モデルの開発・運用業務を担当後、Google Cloud に入社。入社後はデータ分析分野のカスタマー エンジニアとして、BigQuery をはじめとするデータ分析ソリューションでお客様のビジネスを変革する支援を行っている。

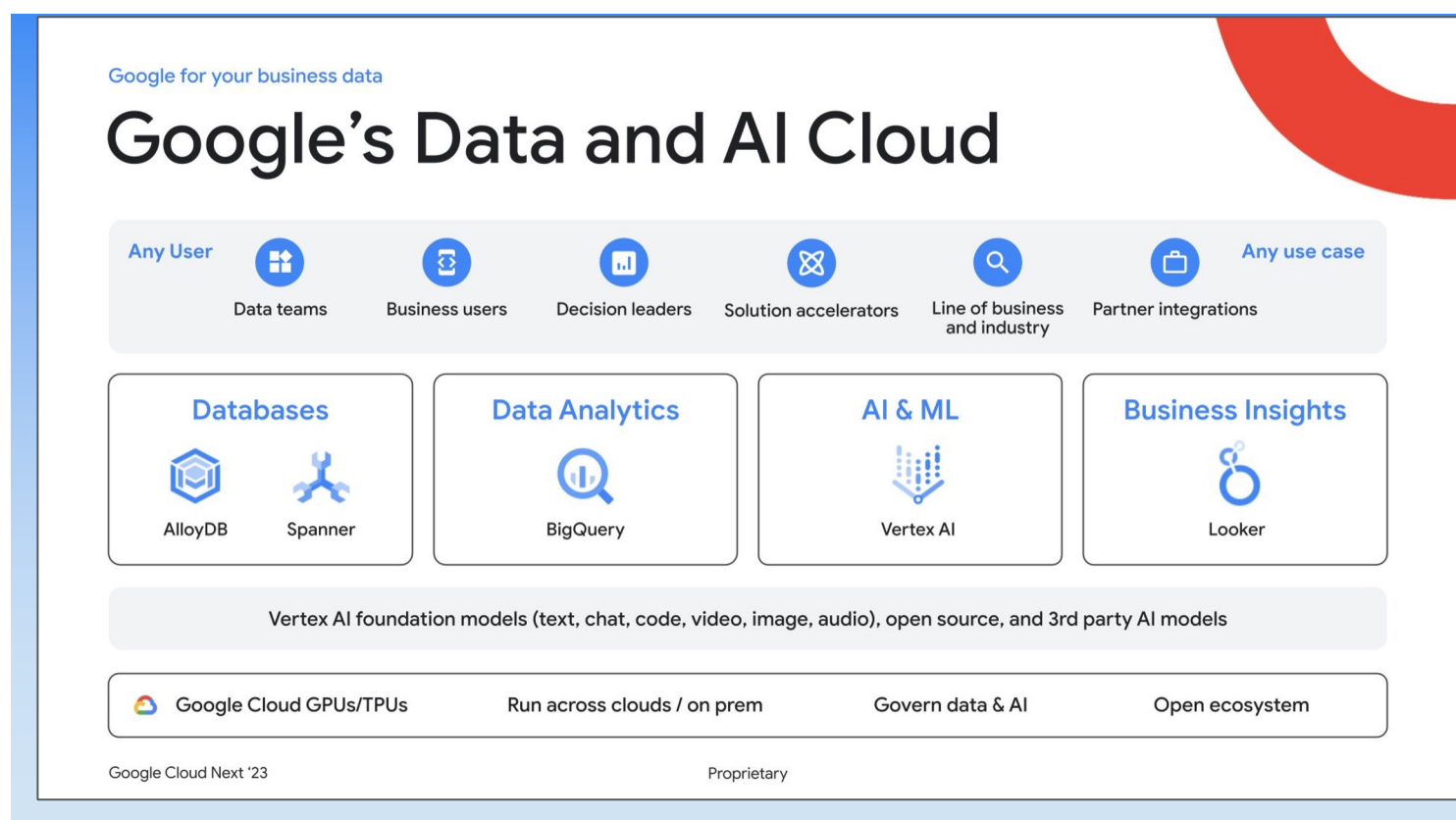
## 目次

- Google Cloud の各ソリューションに組み込まれる機械学習機能 3
- Colab Enterprise と接続した「BigQuery Studio」とは 4
- BigQuery DataFrames とは 5
- LLM で進化する BigQuery の可能性 6
- BigQuery と Retrieval Augmented Generation (RAG) 10
- BigQuery 上で RAG を実行可能 13

## Google Cloud の各ソリューションに組み込まれる機械学習機能

機械学習の機能は、Google Cloud のソリューション群でさまざまなところに統合されています。機械学習の統合開発ソリューションである「Vertex AI」はもちろん「AlloyDB」や「Cloud Spanner」でも、データに対して Vertex AI のモデルを適用し、推論結果をデータベースに回してアプリケーションで活用することが可能です。

また、BI ツールとして使用可能な「Looker」では、機械学習の推論結果をビジネス アプリケーションに組み込む使い方ができるようになっています。



### 統合された機械学習機能

「BigQuery」にも機械学習機能が統合されており、BigQuery ML では SQL での機械学習モデル開発、Vertex AI のモデル接続などが可能です。しかし、BigQuery を使った機械学習モデルの開発に際しては、実際には SQL だけでなく複数のツールを組み合わせるケースがほとんどでしょう。

そのため、各ツールに習熟したり複数のプログラミング言語を使い分けたり、またデータ転送したりしなければなりません。時間と手間がかかるだけでなく、データ転送を行うとなればコンプライアンスの懸念もあります。そこで、今回登場したのが「BigQuery Studio」です。

## Colab Enterprise と接続した「BigQuery Studio」とは

BigQuery Studio は、データ分析や機械学習開発のために作られたコラボレーティブなワークスペースです。開発環境と言い換えてもよいかもしれません。最大の特徴は「Colab Enterprise」とのインテグレーションです。Colab Enterprise は、Google が提供するエンタープライズ向けの Jupyter Notebook 環境です。インスタンスの管理不要で、必要に応じてランタイムに接続して使用できます。

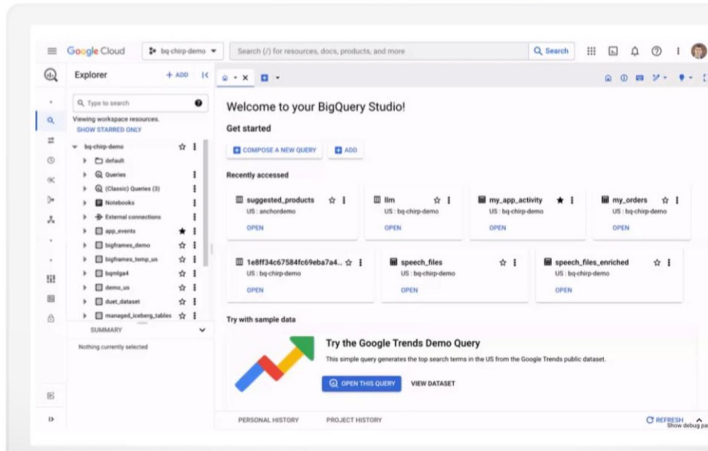
ここにセキュリティやガバナンスといった機能を統合し、エンタープライズ向けに使用できるようにしたのが Colab Enterprise です。Colab Enterprise を BigQuery から使えるようになったことで、SQL のほかにノートブックも同じ環境上で記述できるようになりました。

Unified Preview

## BigQuery Studio: コラボレーティブなデータ分析 / 機械学習のワークスペース

全てのデータ プラクティショナーのデータ分析や AI のワークフローを加速

- **Colab Enterprise と BigQuery** が統合されデータアナリスト、データサイエンティスト、エンジニアがコラボレーションする場として提供
- **自動化されたデータプロファイリング、データ品質、リネージ**を利用可能
- **AI によるチャット、コードアシスタント**により生産性を最大化
- **BigQuery DataFrames** により、馴染みのある Pandas / scikit-learn の書き心地で BigQuery のパワーを活かした分析と機械学習を実行



Google Cloud Next '23 Proprietary

### 新たな機械学習のワークスペース

これまで Dataplex で提供されていたデータ プロファイリング、データ品質、データリネージなどの機能をコンソールからも参照できるようになりました。さらにコード生成やデバッグ、リファクタリングなどを AI アシスタントが補助してくれます。

## BigQuery DataFrames とは

特に注目したい機能が「BigQuery DataFrames」です。BigQuery DataFrames は「bigframes」という Python のパッケージで提供されます。Pandas や scikit-learn などと同じ書き心地でコードを書くことができ、裏側で BigQuery の処理がオフロードされます。極めて巨大なデータセットでも Python の基本知識だけでデータ分析が可能です。

### BigQuery DataFrames で始まる次世代の分析体験

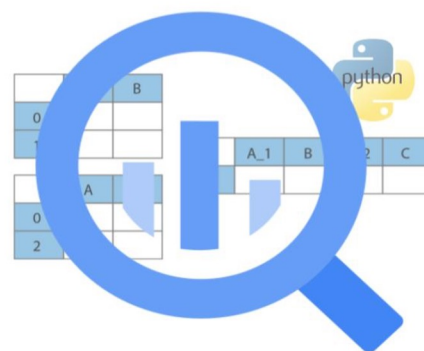
Python パッケージ “bigframes” で、BigQuery のパワーを pandas や scikit-learn のインターフェースから呼び出せる

#### bigframes.pandas

- データ import (read) / export (write)
- データ変換処理
- Notebook 上での柔軟な可視化 (matplotlib, seaborn)
- 任意の Python 関数を BigQuery 上でスケール実行

#### bigframes.ml

- scikit-learn のインターフェースで BQML を実行
- Vertex API を呼び出し



Google Cloud Next '23

Proprietary

BigQuery のパワーを、Python を書き心地で呼び出せるように

パッケージは「bigframes.pandas」と「bigframes.ml」の2つを提供しています。bigframes.pandas が Pandas に、bigframes.ml が scikit-learn に似た API を提供するものになっています。どちらを使用しても裏側では BigQuery が実行されます。

これまでのノートブック開発では、ノートブック上のメモリに乗らない巨大なデータを分析したい場合、一般的なライブラリでは取り扱いにくく、ローカルディスクにキャッシュとして吐き出したり、分散クラスターを作ったりする必要がありました。しかし、専門知識を求められる上に、データサイズに合わせて実装を切り替える手間もかかっていました。

これを bigframes.pandas に書き換えるだけで、ロジックそのままに実際の処理は BigQuery にオフロードされる使い方が可能です。特別な知識は要らず、ローカルマシンのメモリを気にする必要はありません。

BigQuery Studio を用いたイメージとして、講演の[アーカイブ動画](#)にて、入手したあるデータに対してリネージやプロファイルなどの概要をつかんだ上で分析するデモを紹介していますので、ぜひご覧ください。bigframes.pandas をインポートし、BigQuery のテーブルを読み込んだ上で、あとは pandas の使い勝手でデータを処理し分析する流れをご覧ください。

## LLM で進化する BigQuery の可能性

ここまで BigQuery Studio を紹介してきましたが、ここからは、大規模言語モデル (LLM) を統合したことで、BigQuery の可能性がどのように変わるのかを解説します。大きなアップデートとして、Google が開発した LLM である PaLM を BigQuery から呼び出せるようになりました。Vertex AI 上にホストされている PaLM に BigQuery 上のデータをプロンプトとして入力し、その結果を得られるようになりました。

呼び出すための具体的な手順を、次から解説します。

## PaLM を呼び出すには

PaLM を呼び出す際には、アクセス権限をコントロールするため、事前にコネクションを作成する必要があります。権限の設定が終われば、呼び出す手順は下図に示す 2 つです。

## PaLM を呼び出すには

- 1 事前に作成したコネクションを利用し LLM モデルを登録する
- 2 ML.GENERATE\_TEXT() で推論実施 右の例では city の属する 国名を生成させようとしている

```
CREATE MODEL my_project.my_company.llm_model
  REMOTE WITH CONNECTION
  my_project.us.remote_connection_name
  OPTIONS (remote_service_type =
  'CLOUD_AI_LARGE_LANGUAGE_MODEL_V1')
```

```
SELECT * FROM
  ML.GENERATE_TEXT (
    MODEL 'my_company.llm_model',
    (SELECT CONCAT ("Give the country name
  for city: ", city) AS prompt
  FROM example_table),
  STRUCT ( 0.2 AS temperature,
  1024 AS max_output_tokens,
  0.8 AS top_p,
  40 AS top_k))
```

Google Cloud Next '23 Proprietary

### PaLMを呼び立つ 2 つの手順

まず 1 つ目の手順はモデルの登録です。Vertex AI 上でホストされているモデルは、多種多様なバリエーション、バージョンが存在します。いずれか 1 つを選び、名前を付けて保存することで、特定のモデルを後から同じように呼び出せます。

次に、ML.GENERATE\_TEXT 関数を使います。これで PaLM に指示を投げられます。ML.GENERATE\_TEXT 関数は 3 つの引数を取り、第 1 引数は先ほど保存したモデル名です。

第 2 引数は上図の「SELECT CONCAT」で始まる部分で、プロンプトを行ごとに渡します。

「Give the country name for city」の City に Tokyo や Las Vegas といった都市名が入るので、結果として、その都市が位置する国、例えば Tokyo なら Japan が返ってくることが期待されます。このようにプロンプトを SQL で生成して PaLM に実行させることが可能です。

第 3 引数はモデルに渡すパラメータになっており、実用上は temperature や max\_output\_tokens などを編集することが多くなります。

このプロンプトを実行したものが下図です。

## PaLM を呼び出すには

行	ml_generate_text_result	ml_generate_text_status	prompt
1	<pre>{   "predictions": {     "citationMetadata": {       "citations": []     },     "content": " United States",     "safetyAttributes": {       "blocked": false,       "categories": [],       "scores": {}     }   } }</pre>		Give the country name for city: Las Vegas
2	<pre>{   "predictions": {     "citationMetadata": {       "citations": []     },     "content": " United Kingdom",     "safetyAttributes": {       "blocked": false,       "categories": ["Toxic"],       "scores": {         "Toxic": 0.1       }     }   } }</pre>		Give the country name for city: London
3	<pre>{   "predictions": {     "citationMetadata": {       "citations": []     },     "content": " United States",     "safetyAttributes": {       "blocked": false,       "categories": ["Legal"],       "scores": {         "Legal": 0.1       }     }   } }</pre>		Give the country name for city: San Francisco
4	<pre>{   "predictions": {     "citationMetadata": {       "citations": []     },     "content": " Japan",     "safetyAttributes": {       "blocked": false,       "categories": ["Toxic"],       "scores": {         "Toxic": 0.1       }     }   } }</pre>		Give the country name for city: Tokyo

Google Cloud Next '23 Proprietary

プロンプト実行後

3列目が PaLM に入力されたプロンプトです。「ラスベガス、ロンドン、サンフランシスコ、東京はそれぞれどこの国か」を聞いています。その答えが1列目に入っており、contentの部分に正しい答えが入っています。このように所有データに対し、LLMを適用してさまざまな処理が可能です。



ここまでは SQL での話でしたが、普段 LLM 開発で使っている Python から先ほどの LLM を呼び出したいと思うでしょう。今度は Bigframes をフル活用して BigQuery と LLM のパワーを、Python から実行する方法をデモで作成しています。こちら講演の[アーカイブ動画](#)をぜひご覧ください。

BigQuery での LLM の使い道は、コンテンツ生成や分類など多岐にわたる利用方法が考えられるので、ビジネスニーズに合わせてお使いください。

## BigQuery での LLM の使い道

- **コンテンツ生成**- パーソナライズされたメール文面を行動履歴から生成
- **分類**- ブログの文章からカテゴリを自動判定
- **感情分析**- コメント・レビューの傾向生成
- **特徴量生成**- 都市名を含むテキストに、その都市の属する国情報も付加
- **要約**- 長文のコメントやレビューの概要を表示
- **エンティティ抽出**- 記事中の特徴的な用語を抽出
- **文章の校正**- 文法や語法の間違いを修正

ビジネスニーズによってさまざまな使い方があ

## BigQuery と Retrieval Augmented Generation (RAG)

ここからは、LLM と情報検索を組み合わせて、LLM をより実用的に使うための「RAG (Retrieval Augmented Generation)」と呼ばれる技術について説明します。

LLM はもっともらしい回答をしてくれますが、精度については残念ながら十分とは言えません。対照的なのが古典的な情報検索技術であり、ユーザーの質問をそのまま受け取ることはできませんが、精度については既存のドキュメントの中から整合性の高いものを返してくれます。

そこで考えられるのが、LLM と情報検索技術を組み合わせ、双方の長所を活かすアプローチです。これが RAG と呼ばれるアーキテクチャーです。RAG は「埋め込み (Embedding : エンベディング)」と「ベクトル検索」という2つの要素技術を使うと実装できるので、それぞれ説明します。

### ベクトルに変換して元データを埋め込む

ここに「To be, or not to be: that is the question (生きるべきか死ぬべきか、それが問題だ)」というハムレットの一節があります。このデータを必要に応じて検索可能にするにはどうすればいいでしょうか。RAG でよく使われる方法はテキストをベクトル、つまり数値列に変換することです。これをエンベディングと呼んでいます。

もちろん、むやみやたらに数字に変換しても意味がないので、きちんと元の文章の意味を理解した上でベクトルに変換する必要があります。この変換するモデルを Embedding Model などと呼んでいます。なお、Google Cloud ではテキストや画像、動画の Embedding Model を提供しています。この元データの意味を理解して変換するところに、LLM の技術が使われています。

## ベクトルを比較して類似度を検索

テキストを埋め込んでいくとベクトルの格納庫である「ベクトルストレージ（ベクトルストア）」が作れます。このベクトルストレージには、さまざまなフレーズが格納されています。

## ベクトル ストレージ

<p>'To be, or not to be: that is the question'</p>	Model	<p>[0.040746722370386124,0.04314334318 0418015,0.011267253197729588,0.0116 15016497671604,0.04.023249484598636 627,0.018220318481326103,...</p>
<p>All the world's a stage, and all the men and women merely players</p>	Model	<p>[0.040746722370386124,0.04314334318 0418015,0.011267253197729588,0.0116 15016497671604,0.04.023249484598636 627,0.018220318481326103,...</p>
<p>My tricks are not bad, said the cat in the hat.</p>	Model	<p>[0.040746722370386124,0.04314334318 0418015,0.011267253197729588,0.0116 15016497671604,0.04.023249484598636 627,0.018220318481326103,...</p>
<p>I do not like them, Sam-I-Am. I do not like green eggs and ham.</p>	Model	<p>[0.040746722370386124,0.04314334318 0418015,0.011267253197729588,0.0116 15016497671604,0.04.023249484598636 627,0.018220318481326103,...</p>

Google Cloud Next '23
Proprietary

ベクトルの形でフレーズを格納する

例えば、シェイクスピアと絵本作家のドクター・スースのフレーズがある場合、Embedding Model は元の文章の意味を理解した上で埋め込むので、似た作家のベクトルは似た位置に埋め込まれることになります。

ここで新たなフレーズの「You're off to great places! Today is your day. Your mountain is waiting. So...get on your way!」が登場したとします。シェイクスピアかドクター・スースか分からない場合、判定するためにはこれまでと同様にベクトルへ変換します。

すると他のフレーズと数値で比較できるので、新しいベクトルの周辺にある他のベクトルを検索すると、この例ではドクター・スースの方に近く、彼のフレーズだと判定させられます。

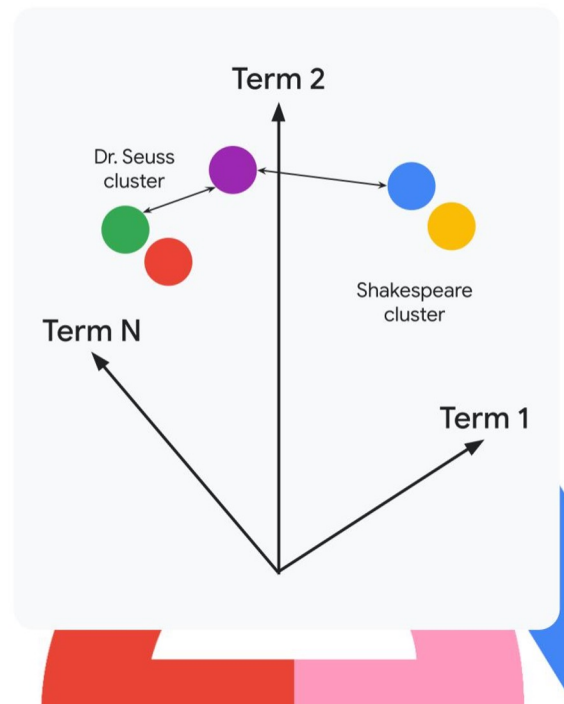
## ベクトル類似度 検索の例

### 検索クエリ

You're off to great places! Today is your day. Your mountain is waiting. So...get on your way!

Model

```
[0.040746722370386124,0.04314334318  
0418015,0.011267253197729588,0.0116  
15016497671604,0.04.023249484598636  
627,0.018220318481326103,...
```



Google Cloud Next '23

Proprietary

作家とベクトルの類似性を示す

これを応用すると、ユーザーの質問に近いデータを検索して返すことも可能です。エンベディングとベクトル検索を用いて見つかったデータを LLM に渡すことで、LLM は実際のデータに基づいて回答生成を行うことになるので、すなわち精度が上がるのが期待できます。このアーキテクチャーが RAG です。

## BigQuery 上で RAG を実行可能

BigQuery でも、エンベディングとベクトル検索をサポートして RAG を実行できるようにしています。エンベディングの利用方法は比較的簡単で、モデルに関して名前を付けて保存した後、「ML.GENERATE\_TEXT\_EMBEDDING」と呼ばれる関数を読むと、エンベディングを取得できます。

検索する前に、あらかじめ検索対象としたいデータ全件に対してエンベディングを取得しておいて、後で使うわけです。その上で「ML.DISTANCE」関数で、あるデータとそれ以外の全データとの距離を測れるので、そこから類似するものを返せます。

具体的にどうやって組み合わせ、RAG を実装するのかについては、講演の[アーカイブ動画](#)でデモをご覧ください。

このデモでは「BigQuery でエンベディングを扱うにはどのような Python のパッケージが良いか」というプロンプトに対して、LLM から適切なパッケージを提案してもらう流れを示しています。Python のパッケージ情報を記載したテーブルを ML.GENERATE\_TEXT\_EMBEDDING 関数で、エンベディング化してベクトルストレージを構築し、プロンプトを適宜拡張しながらベクトル検索を行い、最終的に精度の高い回答が得られるまで方法を示しています。

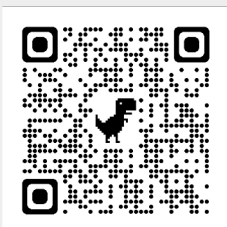
ここまで、さまざまな技術を紹介させていただきました。最後のまとめとして、BigQuery Studio はリネージ機能や Colab Enterprise、bigframes がインテグレーションされた開発環境として、ツールの垣根を越えたデータ分析ができるようになったので、ぜひ体験してみてください。

また、LLM を SQL やノートブックで呼び出して活用できるというメリットはもちろん、ベクトル検索のユースケースでは、クエリの生成や変換や要約まであらゆる分野で LLM のパワーを活用できるようになっています。ぜひ皆様のビジネスでもお役立てください。

## 参照リンク

1. [BigQuery 製品紹介ページ](#)
2. [BigQuery と PaLM で進化するデータ分析 アーカイブ動画視聴ページ](#)

## 製品、サービスに関するお問い合わせ



[goo.gl/CCZL78](https://goo.gl/CCZL78)

Google Cloud の詳細については、上記 URL もしくは QR コードからアクセスしていただくか、同ページ「お問い合わせ」よりお問い合わせください。

© Copyright 2024 Google

Google は、Google LLC の商標です。その他すべての社名および製品名は、それぞれ該当する企業の商標である可能性があります。