

TEXT CLASSIFICATION WITH GOOGLE CLOUD PLATFORM

I. PROBLEM STATEMENT:

The goal is to build an end-to-end text classification system that can categorize text documents into custom categories defined by the user. The key requirements are:

- Allow users to easily label text data into custom categories through an interactive interface. This labeled data can then be used for training.
- Utilize Google Cloud services like Natural Language API and Cloud Machine Learning Engine to analyze the text data and train classification models.
- The system should handle the entire workflow from data labeling to model training and evaluation.
- It should be able to iterate and improve the model accuracy by allowing users to re-label data and retrain models.

The problem is to provide an easy-to-use text classification system for custom user-defined categories leveraging Google Cloud's ML capabilities. The system should cover the end-to-end workflow needed for a text classification task.

II. WHAT WE HAVE BUILD:

In the "Text Classification with Google Cloud Platform" project, we construct a powerful text classification solution using Google Cloud Platform's (GCP) diverse set of services. Starting with an introduction to text classification and GCP, we delve into services like the Natural Language API, AutoML Natural Language, AI Platform, and BigQuery. We learn to perform sentiment analysis, entity recognition, and more with the Natural Language API. AutoML Natural Language enables the creation of custom models tailored to specific needs, while AI Platform offers scalability and flexibility for model development and deployment. BigQuery assists in preprocessing and analyzing text data. The project guides us through a step-by-step workflow for implementing text classification, with real-world case studies showcasing its practical applications. Best practices, challenges, and future trends in text classification are addressed, providing a comprehensive understanding of the topic, and empowering us to build effective text classification solutions on GCP.

III. HOW WE HAVE ADDRESSED THE ISSUES USING CLOUD TECHNOLOGY:

Here is how I would address the text classification problem using Google Cloud Platform:

- o Use Google Cloud Storage to store the text data files as well as intermediate files like labeled data CSVs, trained models etc. This provides a scalable and reliable storage layer.
- o Use Cloud Natural Language API to analyze the text data and auto-suggest potential labels based on content detection. This makes the labeling process faster.
- o Build an interactive Python terminal-based tool for labeling the text data based on NL API suggested labels. Store labeled data as CSVs in Cloud Storage.
- o Use Cloud Machine Learning Engine to train TensorFlow models on the labeled data stored in Cloud Storage. Take advantage of distributed training on GPUs for fast model training.
- o Evaluate models on a hold-out test set. If model accuracy is not satisfactory, enable re-labeling of data using the terminal-based tool.
- o Retrain models on newly labeled data. Repeat until desired accuracy is reached.
- o Provide an easy-to-use Python interface to interact with all the cloud services and orchestrate the end-to-end workflow.
- o Leverage the scalability and reliability of Google Cloud to build an end-to-end text classification pipeline that can handle large amounts of data and iterations.

IV. TOOLS AND TECHNOLOGIES USED:

1. Frontend:

- The frontend is the user-facing part of the system, responsible for presenting the project's features and functionality to users. In this project, the frontend could include:
- **User Interface (UI):** The graphical interface where users interact with the system. This might involve a web-based dashboard or application.
- **Labeling Interface:** A user-friendly interface where users can assign labels to data or provide feedback on the system's predictions.
- **Training Interface:** An area where users can initiate model training or retraining, potentially visualizing the progress and performance metrics.

2. Backend:

- The backend is the behind-the-scenes infrastructure that handles data processing, storage, machine learning model training, and prediction services. In this project, the backend includes:
- **Server-side Logic:** The logic and algorithms that process user requests, manage data flow, and interact with GCP services.
- **Data Storage:** The backend manages the storage of text data, labeled datasets, and machine learning models, likely utilizing Google Cloud Storage for efficient and scalable storage.
- **Machine Learning Engine Integration:** The backend orchestrates the interaction with Cloud Machine Learning Engine for distributed training of TensorFlow models.
- **Compute Engine Integration:** Integration with Compute Engine for deploying and managing machine learning training jobs and prediction services.

3. Communication Between Frontend and Backend:

- **Python**, as the primary programming language, is likely used to build interfaces on both the frontend and backend. Communication channels, such as APIs (Application Programming Interfaces), facilitate the interaction between the user interface and the underlying GCP services. Users interact with the frontend, triggering actions that result in requests being sent to the backend, where the necessary processing and interactions with GCP services occur.

In summary, the collective integration of Google Cloud Platform and its associated technologies—Google Cloud Storage, Cloud Natural Language API, Cloud Machine Learning Engine, Compute Engine, and Python—forms a powerful ecosystem. This ecosystem not only accommodates the storage and analysis of textual data but also enables efficient training and deployment of machine learning models. Through a user-friendly interface, it empowers users to label and train/retrain models effectively, making it a comprehensive solution for building a scalable and sophisticated text classification system.

V. SECURITY MEASURES:

Implementing robust security measures is crucial for any project, especially those involving sensitive data and cloud services. Here are some key security considerations and measures for the text classification project on Google Cloud Platform:

1. Identity and Access Management (IAM):

- Implement least privilege principles, granting only the necessary permissions to users and services.
- Regularly review and audit IAM roles to ensure they align with current project needs.
- Utilize service accounts with the principle of least privilege.

2. Data Encryption:

- Enable encryption at rest for stored data in Google Cloud Storage to protect against unauthorized access.
- Utilize Google Cloud Key Management Service (KMS) for managing and controlling encryption keys.
- Enable HTTPS for secure communication between the frontend and backend.

3. Network Security:

- Leverage Virtual Private Cloud (VPC) networks to isolate and control the flow of network traffic.
- Implement firewall rules to control incoming and outgoing traffic to and from resources.
- Utilize Identity-Aware Proxy (IAP) for controlling access to your applications based on user identity.

4. Data Loss Prevention (DLP):

- Implement Google Cloud's Data Loss Prevention API to automatically detect and redact sensitive information in text data.
- Regularly scan and audit stored data for potential leaks of sensitive information.

5. Monitoring and Logging:

- Set up logging and monitoring using Google Cloud's Stackdriver to detect and respond to security incidents.
- Implement alerts for suspicious activities or unauthorized access.

6. Authentication and Authorization:

- Implement strong authentication mechanisms for both the frontend and backend.
- Use Identity Providers (IDPs) and OAuth 2.0 for secure user authentication.
- Employ granular authorization controls to restrict access to specific features and data.

7. Regular Audits and Reviews:

- Conduct regular security audits of your infrastructure, codebase, and configurations.
- Perform code reviews to identify and address potential security vulnerabilities.

8. Incident Response Plan:

- Develop and document an incident response plan outlining procedures for identifying, containing, eradicating, recovering, and learning from security incidents.
- Regularly conduct tabletop exercises to test and improve the incident response plan.

VI. SCALABILITY AND RELIABILITY:

1. Scalability:

Google Cloud Storage:

- Leverage the scalability of Google Cloud Storage to handle increasing amounts of text data, labeled datasets, and model files.

Cloud Machine Learning Engine:

- Utilize the distributed training capabilities of Cloud Machine Learning Engine to scale the training of TensorFlow models based on the size of datasets and computational needs.

Compute Engine:

- Configure Compute Engine instances for elasticity, allowing them to automatically scale based on demand, especially during peak times of model training or prediction services.

AutoML Natural Language:

- If using AutoML Natural Language, take advantage of its automatic model training capabilities, reducing the manual effort required for scaling.

Load Balancing:

- Implement load balancing mechanisms for the frontend to evenly distribute user requests, ensuring a responsive and scalable user interface.

Kubernetes Engine (if applicable):

- If containerized, consider using Google Kubernetes Engine (GKE) for container orchestration, allowing for efficient scaling of containerized applications.

2. Reliability:

Data Redundancy:

- Implement data redundancy strategies within Google Cloud Storage to ensure data availability and prevent data loss.

Data Backups:

- Regularly backup critical data, including labeled datasets and trained models, to mitigate the impact of accidental data loss or corruption.

Multi-Region Deployment:

- Deploy critical components and services across multiple regions to enhance reliability and minimize the impact of regional outages.

Service Level Agreements (SLAs):

- Familiarize yourself with GCP's SLAs for services used in the project, ensuring that service availability meets project requirements.

Monitoring and Alerts:

- Set up proactive monitoring using Stackdriver to detect potential issues and implement alerts for immediate notification of any anomalies.

Automated Scaling Policies:

- Implement automated scaling policies for Compute Engine instances, ensuring that resources dynamically adjust to varying workloads.

Error Handling and Retry Mechanisms:

- Implement robust error handling and retry mechanisms in both frontend and backend components to gracefully handle intermittent failures.

Regular Performance Testing:

- Conduct regular performance testing to identify potential bottlenecks and optimize the system for increased reliability and responsiveness.

VII. PROJECT EXECUTION:

1. Create a Google Cloud Platform (GCP) project.
2. Activate and set your project in cloud shell.
 - Click Activate Cloud Shell at the top of the Google Cloud console.
 - When you are connected, you are already authenticated, and the project is set to your PROJECT_ID. The output contains a line that declares the Project_ID for this session.
 - Your Cloud Platform project in this session is set to "PROJECT_ID."
 - First, run the following command to ensure that the Cloud Shell is using the correct GCP project (replacing <project-name> with the name of project gcloud config set project <project-name>)
3. APIs & services > Credentials in the Cloud Console

The screenshot shows the Google Cloud console's 'Credentials' page. The left sidebar has 'APIs & Services' expanded, with 'Credentials' selected. The main area has a 'CREATE CREDENTIALS' button and a warning: 'Remember to configure the OAuth consent screen with information about your application.' Below this are three sections: 'API Keys' with one key named 'API key 1', 'OAuth 2.0 Client IDs' which is empty, and 'Service Accounts' with two accounts: 'Compute Engine default service account' and 'my-account'.

- Then click create credentials and in the drop-down menu, select API key then Next, copy the key you just generated and click close.
- Now that you have an API key, save it to an environment variable to avoid having to insert the value of your API key in each request. You can do this in Cloud Shell. Be sure to replace <your_api_key> with the key you just copied.
 - **export API_KEY=<YOUR_API_KEY>**
 -

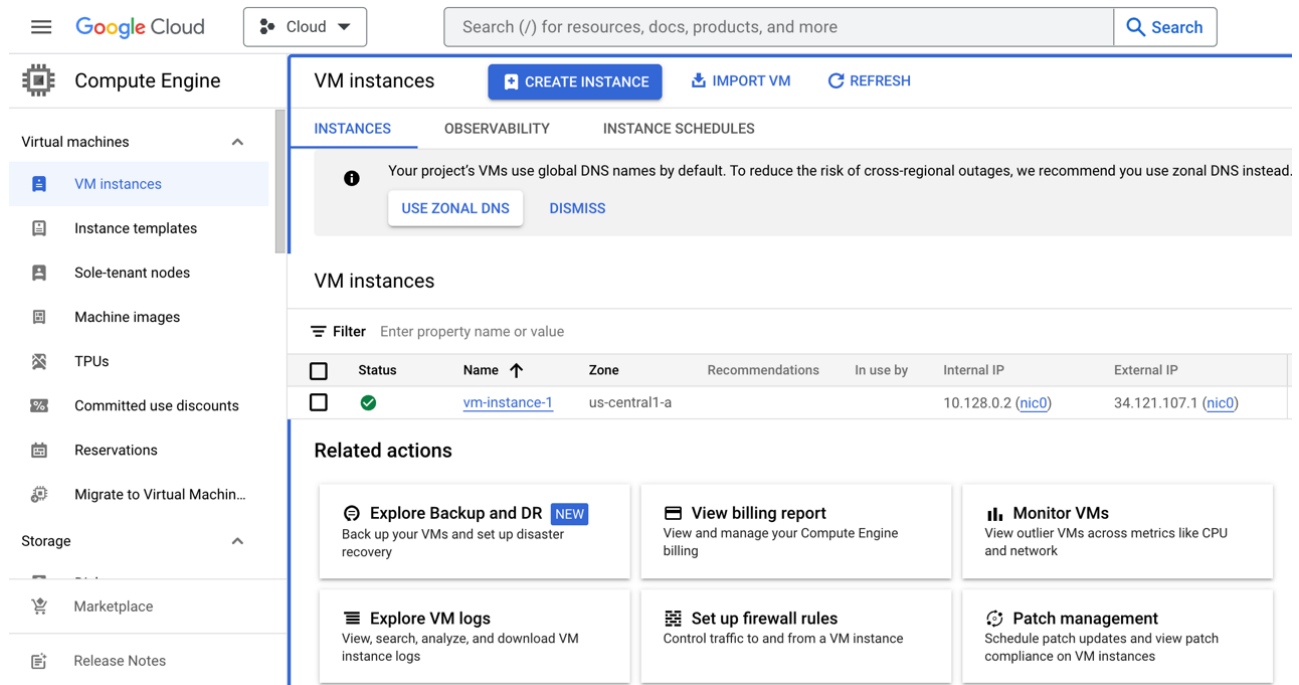
4. Enable the Cloud Natural Language API

- Expand the Navigation menu at the top left of the screen.
- Select APIs & Services > Enabled APIs and Services.
- Click Enable APIs and Services.
- Then, search for language in the search box.
- Click Cloud Natural Language API:
- If the API is not enabled, you'll see the Enable button.
- Click Enable to enable the Cloud Natural Language API.

The screenshot shows the 'Cloud Natural Language API' product details page. The top has a blue header with 'Product details' and a blue icon. The main content area shows the API name, a 'Google Enterprise API' link, a description 'Provides natural language understanding technologies, such as sentiment analysis, entity...', and buttons for 'MANAGE' and 'TRY THIS API'. A green checkmark indicates the API is enabled.

5. To perform next steps please connect to the instance provisioned for you via ssh.

- Open the Navigation menu and select Compute Engine > VM Instances
- Create a VM Instance named as Linux-Instance.
- You should see a provisioned Linux-instance.
- Click on the SSH button. You be brought to an interactive shell.
- In the command line, enter in the following, replacing <YOUR_API_KEY> with the key you just copied.
 - `export API_KEY=<YOUR_API_KEY>`



Google Cloud

Cloud

Search (/) for resources, docs, products, and more

Search

Compute Engine

Virtual machines

- VM instances
- Instance templates
- Sole-tenant nodes
- Machine images
- TPUs
- Committed use discounts
- Reservations
- Migrate to Virtual Machin...

Storage

- Marketplace
- Release Notes

VM instances

CREATE INSTANCE IMPORT VM REFRESH

INSTANCES OBSERVABILITY INSTANCE SCHEDULES

Your project's VMs use global DNS names by default. To reduce the risk of cross-regional outages, we recommend you use zonal DNS instead.

USE ZONAL DNS DISMISS

VM instances

Filter Enter property name or value

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP
<input checked="" type="checkbox"/>	vm-instance-1	us-central1-a			10.128.0.2 (nic0)	34.121.107.1 (nic0)

Related actions

- Explore Backup and DR NEW
Back up your VMs and set up disaster recovery
- View billing report
View and manage your Compute Engine billing
- Monitor VMs
View outlier VMs across metrics like CPU and network
- Explore VM logs
View, search, analyze, and download VM instance logs
- Set up firewall rules
Control traffic to and from a VM instance
- Patch management
Schedule patch updates and view patch compliance on VM instances



ssh.cloud.google.com/v2/ssh/projects/cloud-408201/zones/us-central1-a/instances/vm-instance-1?authuser=5&hl=en_US&pro...

ssh.cloud.google.com/v2/ssh/projects/cloud-408201/zones/us-central1-a/instances/vm-instance-1?authuser=5&hl=en_US&pr...

SSH-in-browser

UPLOAD FILE DOWNLOAD FILE

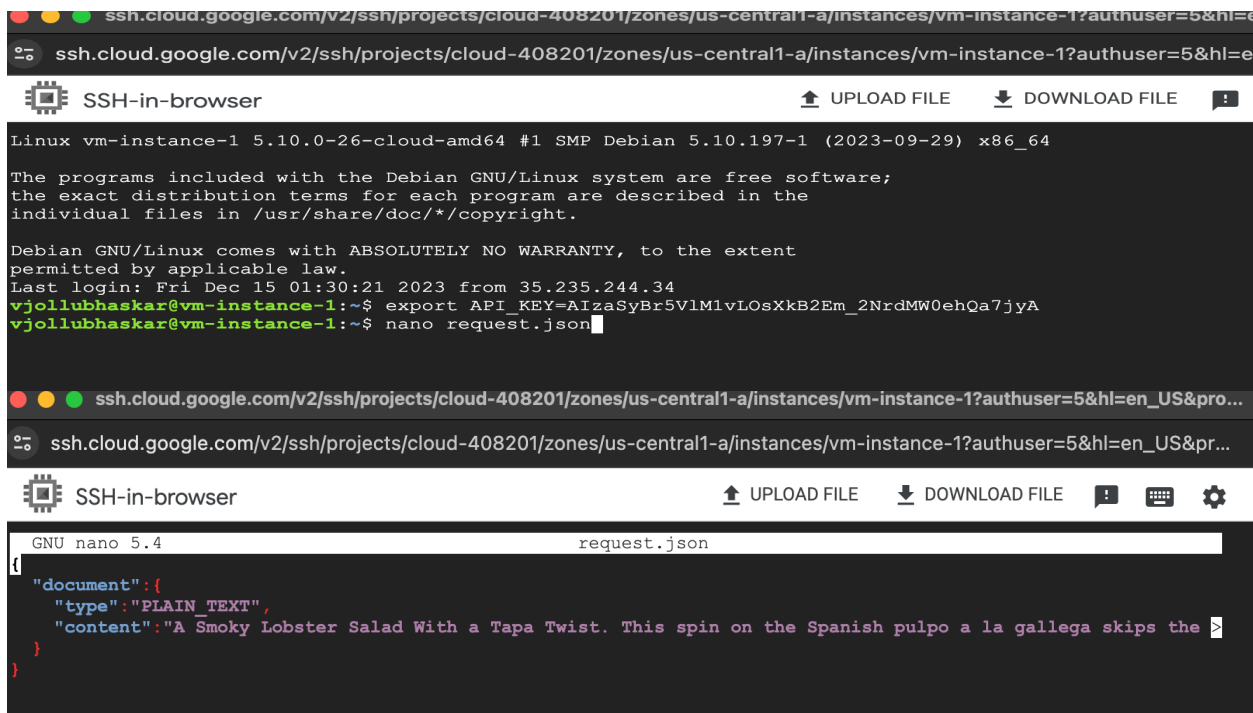
```
Linux vm-instance-1 5.10.0-26-cloud-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Dec 15 01:30:21 2023 from 35.235.244.34
vjollubhaskar@vm-instance-1:~$ export API_KEY=AIzaSyBr5VlM1vLOsXkB2Em_2NrdMW0ehQa7jyA
vjollubhaskar@vm-instance-1:~$
```


6. Classify a news Article.

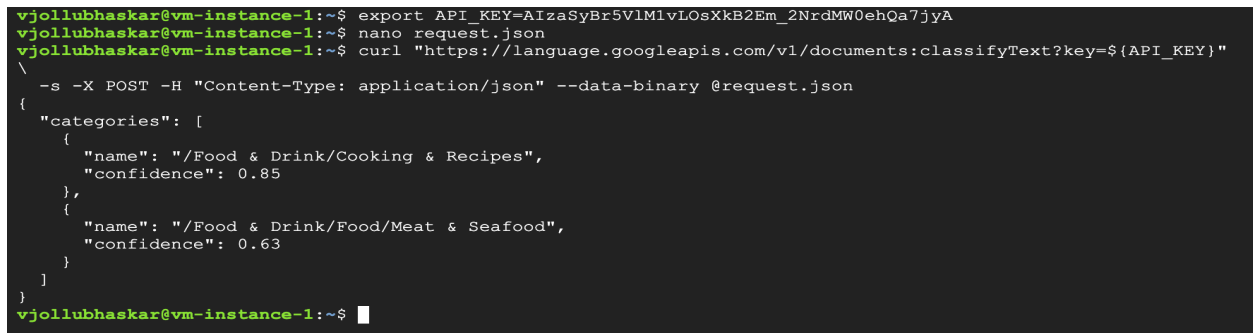
- Using the Natural Language API's classify Text method, you can sort text data into categories with a single API call. This method returns a list of content categories that apply to a text document.
- You'll start by classifying a single article, and then see how you can use this method to make sense of a large news corpus.
- To start, take this headline and description from a New York Times article in the food section.
 - *A Smoky Lobster Salad with a Tapa Twist. This spin on the Spanish pulpo a la gallega skips the octopus, but keeps the sea salt, olive oil, pimentón and boiled potatoes.*
- Create a file named request.json and add the code found below. You can create the file using one of your preferred command lines editors.



```
ssh.cloud.google.com/v2/ssh/projects/cloud-408201/zones/us-central1-a/instances/vm-instance-1?authuser=5&hl=en
SSH-in-browser
Linux vm-instance-1 5.10.0-26-cloud-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Dec 15 01:30:21 2023 from 35.235.244.34
vjollubhaskar@vm-instance-1:~$ export API_KEY=AiZaSyBr5VlM1vLOsXkB2Em_2NrdMW0ehQa7jyA
vjollubhaskar@vm-instance-1:~$ nano request.json

ssh.cloud.google.com/v2/ssh/projects/cloud-408201/zones/us-central1-a/instances/vm-instance-1?authuser=5&hl=en_US&pro...
SSH-in-browser
GNU nano 5.4 request.json
{
  "document": {
    "type": "PLAIN_TEXT",
    "content": "A Smoky Lobster Salad With a Tapa Twist. This spin on the Spanish pulpo a la gallega skips the
  }
}
```

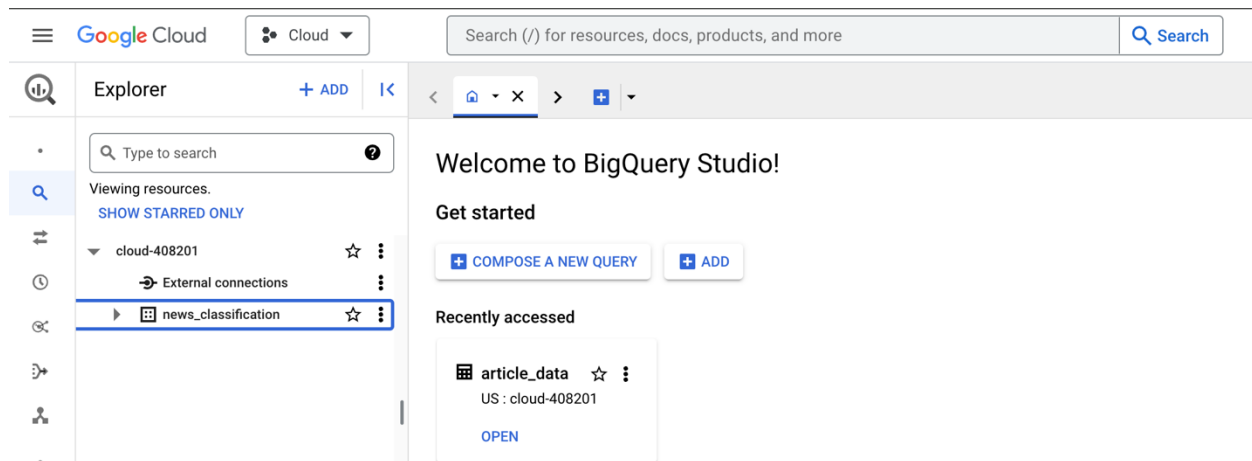
Now you can send this text to the Natural Language API's classify Text method with the following curl command:



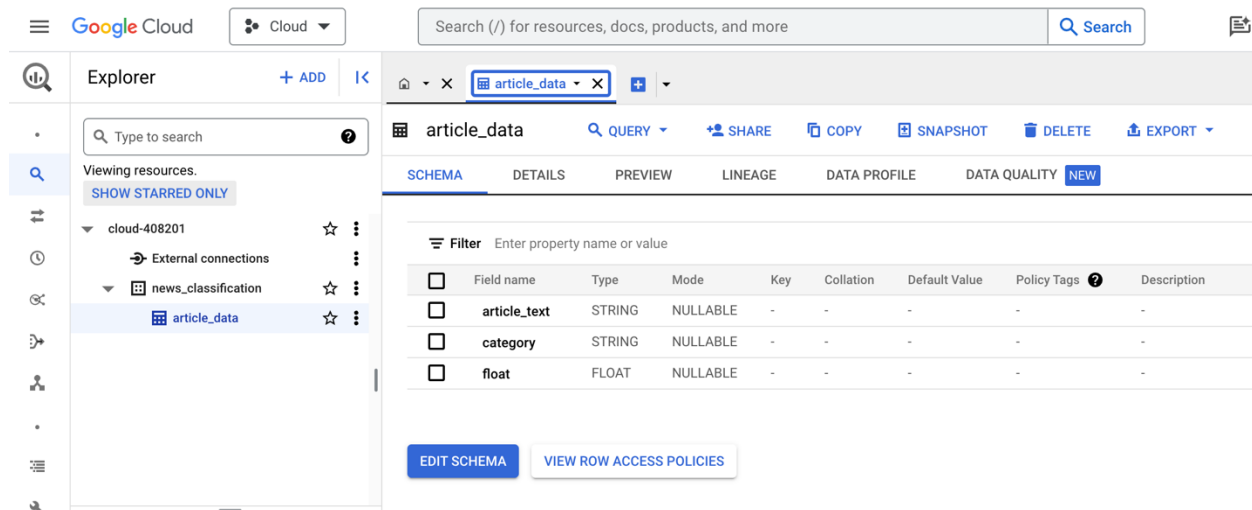
```
vjollubhaskar@vm-instance-1:~$ export API_KEY=AiZaSyBr5VlM1vLOsXkB2Em_2NrdMW0ehQa7jyA
vjollubhaskar@vm-instance-1:~$ nano request.json
vjollubhaskar@vm-instance-1:~$ curl "https://language.googleapis.com/v1/documents:classifyText?key=${API_KEY}"
-s -X POST -H "Content-Type: application/json" --data-binary @request.json
{
  "categories": [
    {
      "name": "/Food & Drink/Cooking & Recipes",
      "confidence": 0.85
    },
    {
      "name": "/Food & Drink/Food/Meat & Seafood",
      "confidence": 0.63
    }
  ]
}
vjollubhaskar@vm-instance-1:~$
```


7. Create a Big Query table for categorized text data:

- Before sending the text to the Natural Language API, you need a place to store the text and category for each article.
- Navigate to Navigation menu > Big Query in the Console
- Click Done.
- To create a dataset, click on the **View actions** icon next to your Project ID and select **Create dataset**.



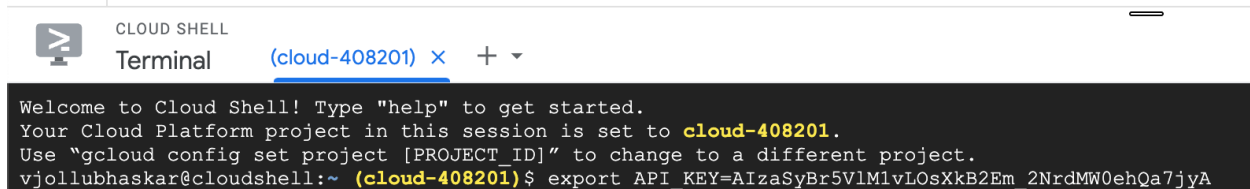
- Name the dataset and create a dataset.
- To create a table, click on the View actions icon next to the dataset table and select Create Table



- Click Create Table.
- The table is empty right now. In the next step you'll read the articles from Cloud Storage, send them to the Natural Language API for classification, and store the result in BigQuery.

8. Classify news data and store the result in BigQuery.

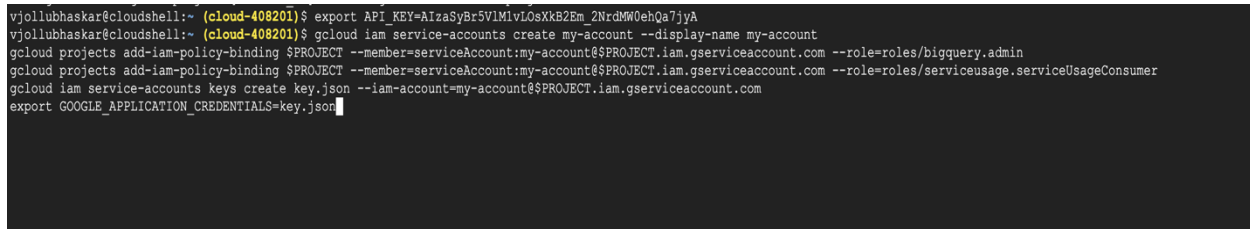
- To perform next steps please connect to the Cloud Shell. If prompted click continue.
- Before writing a script to send the news data to the Natural Language API, you need to create a service account. This will be used to authenticate to the Natural Language API and BigQuery from a Python script.
- Export your Project ID as an environment variable:
- `export PROJECT="Project ID"`



```
CLOUD SHELL
Terminal (cloud-408201) x + v

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cloud-408201.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
vjollubhaskar@cloudshell:~ (cloud-408201)$ export API_KEY=AIZA5yBr5VlM1vLOsXkB2Em_2NrdMW0ehQa7jyA
```

- Run the following commands to create a service account.



```
vjollubhaskar@cloudshell:~ (cloud-408201)$ export API_KEY=AIZA5yBr5VlM1vLOsXkB2Em_2NrdMW0ehQa7jyA
vjollubhaskar@cloudshell:~ (cloud-408201)$ gcloud iam service-accounts create my-account --display-name my-account
gcloud projects add-iam-policy-binding $PROJECT --member=serviceAccount:my-account@$PROJECT.iam.gserviceaccount.com --role=roles/bigquery.admin
gcloud projects add-iam-policy-binding $PROJECT --member=serviceAccount:my-account@$PROJECT.iam.gserviceaccount.com --role=roles/serviceusage.serviceUsageConsumer
gcloud iam service-accounts keys create key.json --iam-account=my-account@$PROJECT.iam.gserviceaccount.com
export GOOGLE_APPLICATION_CREDENTIALS=key.json
```

```
gcloud iam service-accounts create my-account --display-name my-account
gcloud projects add-iam-policy-binding $PROJECT --member=serviceAccount:my-account@$PROJECT.iam.gserviceaccount.com --role=roles/bigquery.admin
gcloud projects add-iam-policy-binding $PROJECT --member=serviceAccount:my-account@$PROJECT.iam.gserviceaccount.com --role=roles/serviceusage.serviceUsageConsumer
gcloud iam service-accounts keys create key.json --iam-account=my-account@$PROJECT.iam.gserviceaccount.com
export GOOGLE_APPLICATION_CREDENTIALS=key.json
```

Now you're ready to send the text data to the Natural Language API

- Write a Python script using the Python module for Google Cloud
- You can accomplish the same thing from any language, there are many different cloud client libraries.
- Create a file called classify-text.py and copy the following code into it. You can either create the file using one of your preferred command lines editors.

```
vjollubhaskar@cloudshell:~ (cloud-408201)$ nano classify-text.py
```

- Create a python file using nano File name.py.
- Now save the code in the .py file which is created.

```
GNU nano 5.4 classify-text.py
from google.cloud import storage, language, bigquery

# Set up your GCS, NL, and BigQuery clients
storage_client = storage.Client()
nl_client = language.LanguageServiceClient()
bq_client = bigquery.Client(project='Project ID')

dataset_ref = bq_client.dataset('news_classification_dataset')
dataset = bigquery.Dataset(dataset_ref)
table_ref = dataset.table('article_data')
table = bq_client.get_table(table_ref)

# Send article text to the NL API's classifyText method
def classify_text(article):
    response = nl_client.classify_text(
        document=language.Document(
            content=article,
            type_=language.Document.Type.PLAIN_TEXT
        )
    )
    return response

rows_for_bq = []
files = storage_client.bucket('qwiklabs-test-bucket-gsp063').list_blobs()
print("Got article files from GCS, sending them to the NL API (this will take ~2 minutes)...")

# Send files to the NL API and save the result to send to BigQuery
for file in files:
    # Send article text to the NL API's classifyText method
    def classify_text(article):
        response = nl_client.classify_text(
            document=language.Document(
                content=article,
                type_=language.Document.Type.PLAIN_TEXT
            )
        )
        return response

    rows_for_bq = []
    files = storage_client.bucket('qwiklabs-test-bucket-gsp063').list_blobs()
    print("Got article files from GCS, sending them to the NL API (this will take ~2 minutes)...")

    # Send files to the NL API and save the result to send to BigQuery
    for file in files:
        if file.name.endswith('.txt'):
            article_text = file.download_as_bytes()
            nl_response = classify_text(article_text)
            if len(nl_response.categories) > 0:
                rows_for_bq.append((str(article_text), nl_response.categories[0].name, nl_response.categories[0].confidence))

    print("Writing NL API article data to BigQuery...")
    # Write article text + category data to BQ
    errors = bq_client.insert_rows(table, rows_for_bq)
    assert errors == []

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo     M-A Set Mark  M-] To Bracket M-;
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo     M-G Copy      ^C Where Was  M-'
```

- Run the following script

```
python3 classify-text.py
```

```
vjollubhaskar@cloudshell:~ (cloud-408201) $ nano classify-text.py
vjollubhaskar@cloudshell:~ (cloud-408201) $ python3 classify-text.py
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/dist-packages/google/auth/_default.py", line 120, in load_credentials_from_file
    info = json.load(file_obj)
  File "/usr/lib/python3.9/json/__init__.py", line 293, in load
    return loads(fp.read()),
  File "/usr/lib/python3.9/json/__init__.py", line 346, in loads
    return _default_decoder.decode(s)
  File "/usr/lib/python3.9/json/decoder.py", line 337, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/lib/python3.9/json/decoder.py", line 355, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/home/vjollubhaskar/classify-text.py", line 4, in <module>
    storage_client = storage.Client()
  File "/usr/local/lib/python3.9/dist-packages/google/cloud/storage/client.py", line 172, in __init__
    super(Client, self).__init__(
  File "/usr/local/lib/python3.9/dist-packages/google/cloud/client/_init_.py", line 321, in __init__
    Client.__init__(
  File "/usr/local/lib/python3.9/dist-packages/google/cloud/client/_init_.py", line 178, in __init__
    credentials, _ = google.auth.default(scopes=scopes)
  File "/usr/local/lib/python3.9/dist-packages/google/auth/_default.py", line 657, in default
    credentials, project_id = checker()
  File "/usr/local/lib/python3.9/dist-packages/google/auth/_default.py", line 650, in <lambda>
    lambda: _get_explicit_environ_credentials(quota_project_id=quota_project_id),
  File "/usr/local/lib/python3.9/dist-packages/google/auth/_default.py", line 270, in _get_explicit_environ_credentials
    credentials, project_id = load_credentials_from_file(
  File "/usr/local/lib/python3.9/dist-packages/google/auth/_default.py", line 125, in load_credentials_from_file
    raise new_exc from caught_exc
google.auth.exceptions.DefaultCredentialsError: ('File key.json is not a valid json file.', JSONDecodeError('Expecting value: line 1 column 1 (char 0)'))
vjollubhaskar@cloudshell:~ (cloud-408201) $
```

- The script takes about two minutes to complete, so while it's running read about what's happening.
- You're using the Google Cloud Python client library to access Cloud Storage, the Natural Language API, and BigQuery. First, a client is created for each service, then references are created to the BigQuery table. Files are a reference to each of the BBC dataset files in the public bucket. The files are looked at, the articles are downloaded as strings, then each one is sent to the Natural Language API in the `classify_text` function. For all articles where the Natural Language API returns a category, the article and its category data are saved to a `rows_for_bq` list. When classifying each article is done, the data is inserted into BigQuery using `insert_rows()`
- In BigQuery, navigate to the `article_data` table in the Explorer tab and click Query > In new tab to query the table:

The screenshot shows the Google Cloud BigQuery Explorer interface. On the left, the 'Explorer' panel shows the project 'cloud-408201' and the dataset 'news_classification' with the table 'article_data' selected. The main panel displays the 'article_data' table with a 'QUERY' button and options to 'In new tab' or 'In split tab'. Below this, the 'SCHEMA' tab shows the table's structure:

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
article_text	STRING	NULLABLE	-	-	-	-	-
category	STRING	NULLABLE	-	-	-	-	-
float	FLOAT	NULLABLE	-	-	-	-	-

```
SELECT * FROM `Project
ID.news_classification_dataset.article_data`
```



- Run the query to see data.
- Scroll to the right to see the category column.
- The category column has the name of the first category the Natural Language API returned for the article, and confidence is a value between 0 and 1 indicating how confident the API is that it categorized the article correctly.

9. Analyze Categorized news data in Big Query

- In the BigQuery console, click + **Compose New Query**
- Enter the following query

```
SELECT
  category,
  COUNT(*) c
FROM
  `Project ID.news_classification_dataset.article_data`
GROUP BY
  category
ORDER BY
  c DESC
```

- Now click **Run**:

Query results			 SAVE RESULTS	 EXPLORE DATA ▼
Query complete (0.4 sec elapsed, 0 B processed)				
Job information <u>Results</u> JSON Execution details				
Row	category	c		
1	/News/Politics	20		
2	/Arts & Entertainment/Movies	19		
3	/Sensitive Subjects	14		
4	/Business & Industrial	14		
5	/Sports/Individual Sports	13		
6	/Sports/Individual Sports/Track & Field	11		
7	/Sports	11		

The screenshot displays the Google Cloud BigQuery console interface. At the top, there's a browser-like header with tabs for 'usa_names' and '*Untitled'. Below this is a toolbar with buttons for 'RUN', 'SAVE', 'DOWNLOAD', 'SHARE', 'SCHEDULE', 'MORE', and a status indicator 'Query complete'. The main area contains a SQL query editor with the following code:

```
1 SELECT
2   name,
3   SUM(number) AS total
4 FROM
5   `bigquery-public-data.usa_names.usa_1910_2013`
6 GROUP BY
7   name
8 ORDER BY
9   total DESC
10 LIMIT
11  10;
```

Below the query editor, the 'Query results' section is active, showing a table with 10 rows. The table has columns 'name' and 'total'. The results are as follows:

Row	name	total
1	James	4942431
2	John	4834422
3	Robert	4718787
4	Michael	4297230
5	William	3822209
6	Mary	3737679
7	David	3549801
8	Richard	2531924
9	Joseph	2472917

VIII. CHALLENGES AND CONSIDERATIONS:

Navigating text classification on Google Cloud Platform comes with its set of challenges, each demanding careful consideration for successful implementation. Imbalanced datasets pose a common hurdle, where certain classes may have significantly more instances than others. Addressing this challenge requires strategies such as oversampling or under sampling to ensure fair representation during model training. Language nuances and context add complexity, as the meaning of words can vary based on context, making it crucial to consider the intricacies of language for accurate classification. Additionally, achieving continuous model improvement is an ongoing consideration. Text data evolves, and models must adapt to changing linguistic patterns and user behavior. Regular model evaluation and retraining, coupled with staying informed about emerging language trends, are essential for maintaining the effectiveness of text classification models over time. Effectively tackling these challenges contributes to the development of robust and adaptive text classification solutions on GCP.

IX. FUTURE TRENDS:

The future of text classification on Google Cloud Platform is poised to witness dynamic trends that align with the evolving landscape of artificial intelligence. Integration with other AI technologies, such as image recognition, is set to play a pivotal role. Combining text and visual data for a more comprehensive understanding enables enhanced context-aware models, contributing to more sophisticated and accurate classifications. Advancements in multilingual text classification are also on the horizon, reflecting the global nature of data. Expect innovations that break down language barriers, allowing models to comprehend and classify text in multiple languages more effectively. Furthermore, a shift towards industry-specific applications and solutions is anticipated. As text classification continues to prove its value across diverse sectors, we can anticipate more tailored approaches and solutions catering to the unique needs of specific industries, further expanding the practical applications of text classification on GCP. Stay tuned as these trends shape the future of text classification in innovative and impactful ways.

X. CONCLUSION:

In conclusion, text classification stands as a pivotal tool in modern data analytics, offering indispensable benefits across various domains. The ability to automatically categorize and understand textual information fuels enhanced customer experiences, efficient information retrieval, and automation of critical tasks. Google Cloud Platform (GCP) plays a central role in simplifying and enhancing text classification projects, providing a comprehensive suite of services like the Natural Language API, AutoML Natural Language, AI Platform, and BigQuery. These services collectively empower users to tackle the complexities of text classification, from sentiment analysis to custom model creation and scalable deployment.

As we've explored the step-by-step workflow, best practices, challenges, and future trends, the importance of GCP's integration becomes evident. The platform not only streamlines the implementation of text classification but also opens avenues for innovation, customization, and scalability. In closing, I encourage further exploration and implementation of text classification on GCP. Dive into the possibilities, leverage the robust tools and services available, and continue to stay abreast of evolving trends. With GCP's support, the journey into text classification promises to be not only insightful but also transformative in the realm of data-driven decision-making.