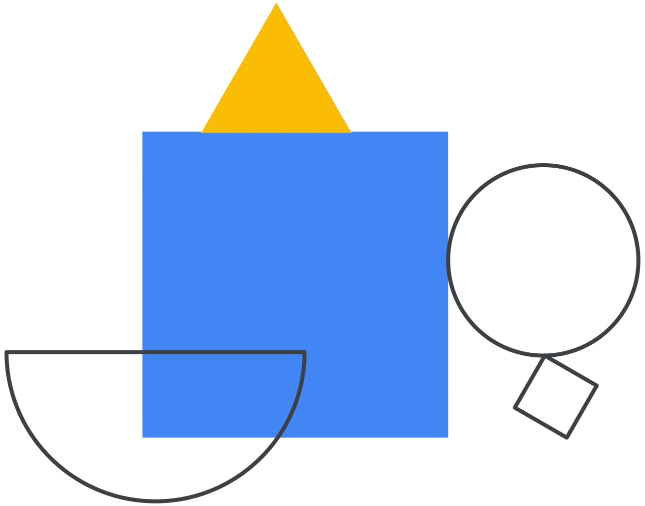
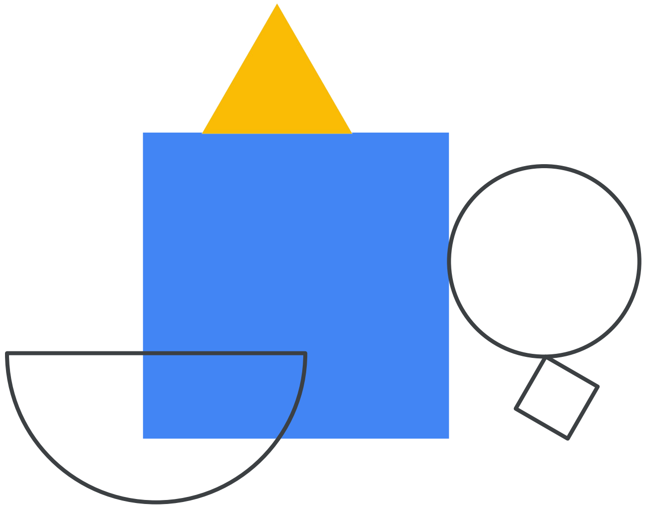
03

Task-specific Solutions

Lab Startup:
Text Classification with Model Garden



Complete Task 1.



Task-specific Solutions

Task-specific solutions

Vision

Process images, video, and documents to detect, extract, classify, and enrich.

Language

Process natural language for sentiment, entities, and translation.

Tabular

Classification and regression solutions for structured data

Models-as-a-Service

Vision

Video Intelligence

Natural Language

Translate

Speech-to-Text

Text-to-Speech

Vertex AI Vision

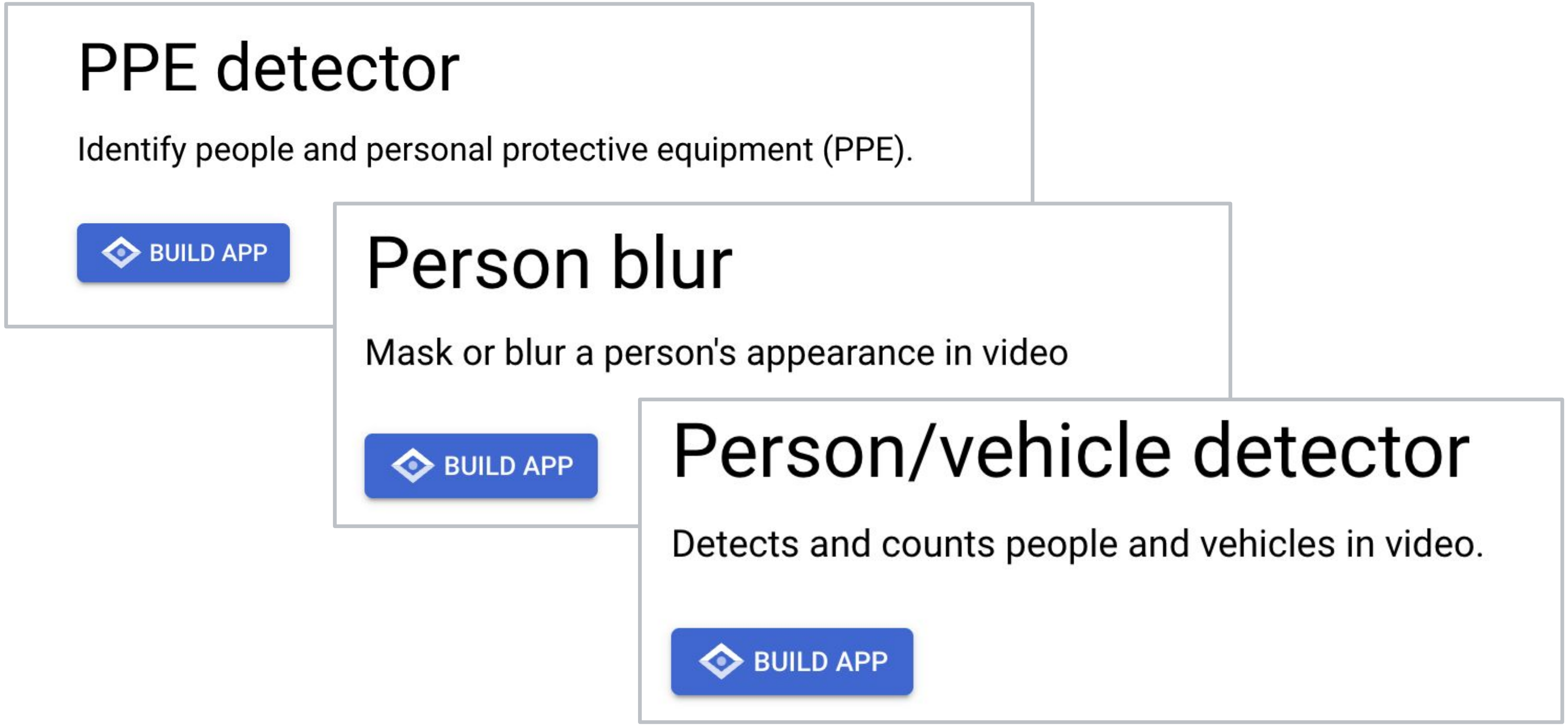
Dialogflow

Document AI

Contact Center AI

Product Discovery

App Platform


PPE detector

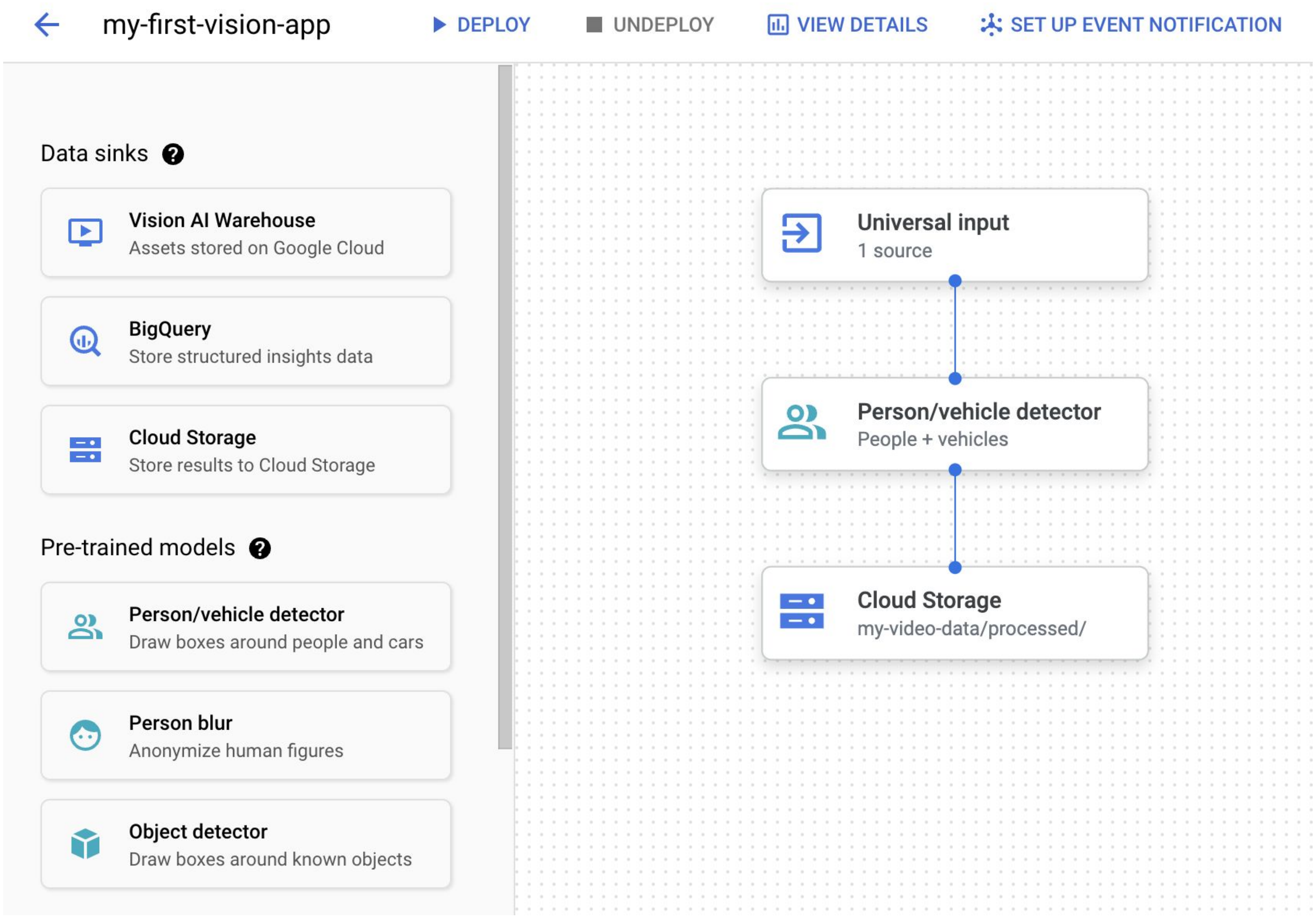
Identify people and personal protective equipment (PPE).

Person blur

Mask or blur a person's appearance in video

Person/vehicle detector

Detects and counts people and vehicles in video.



Data sinks

- **Vision AI Warehouse** Assets stored on Google Cloud
- **BigQuery** Store structured insights data
- **Cloud Storage** Store results to Cloud Storage

Pre-trained models

- **Person/vehicle detector** Draw boxes around people and cars
- **Person blur** Anonymize human figures
- **Object detector** Draw boxes around known objects

Universal input 1 source

Person/vehicle detector People + vehicles

Cloud Storage my-video-data/processed/

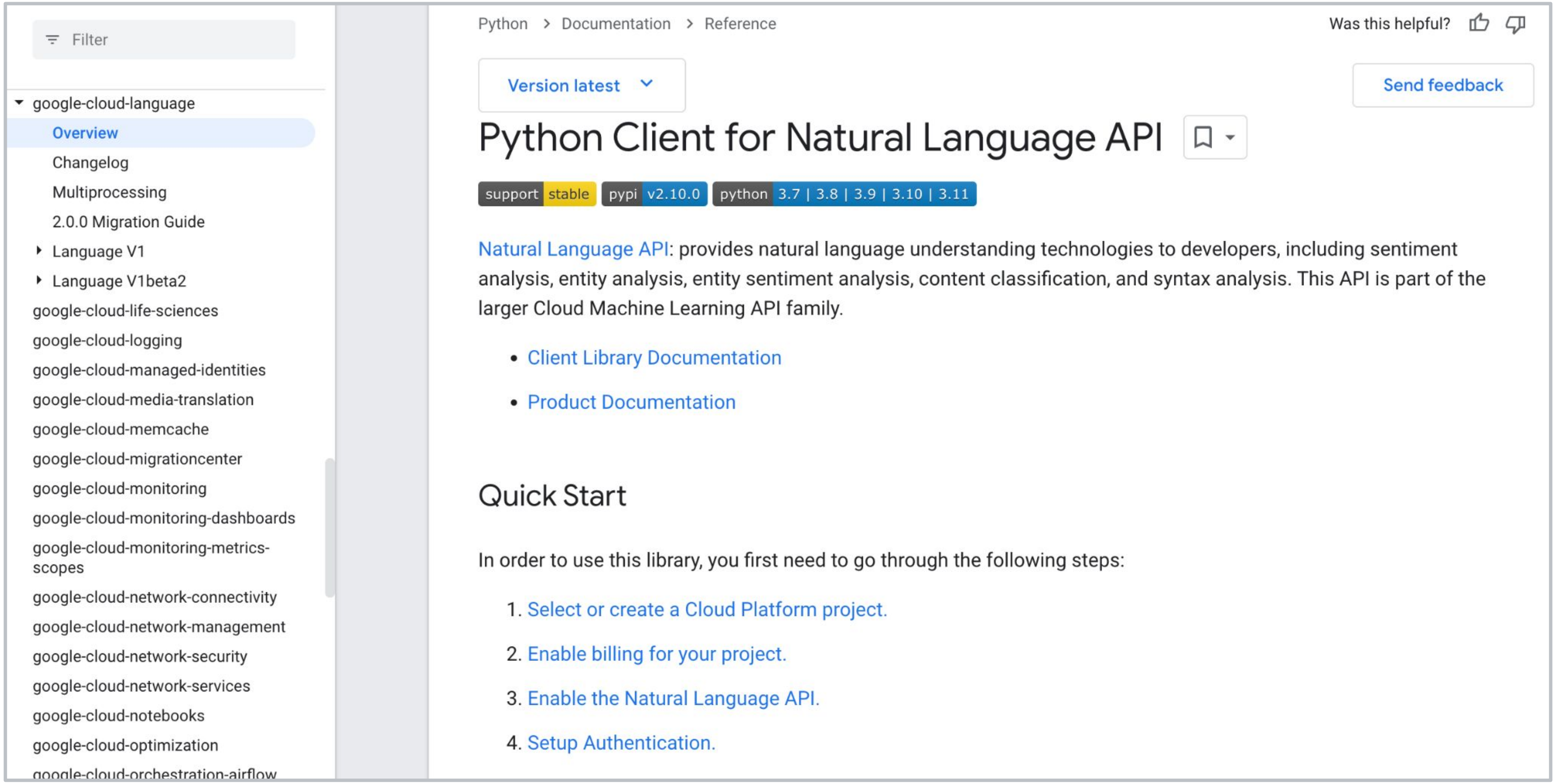
REST APIs

```
curl "https://language.googleapis.com/v1/documents:analyzeSentiment?key=${API_KEY}" \
  -s -X POST \
  -H "Content-Type: application/json" \
  --data-binary @request.json
```

Client libraries

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": content}

response = client.analyze_sentiment(request={"document": document})
sentiment = response.document_sentiment
```

Task-specific Solutions



Filter

google-cloud-language
- Overview
- Changelog
- Multiprocessing
- 2.0.0 Migration Guide
- Language V1
- Language V1beta2

google-cloud-life-sciences
google-cloud-logging
google-cloud-managed-identities
google-cloud-media-translation
google-cloud-memcache
google-cloud-migrationcenter
google-cloud-monitoring
google-cloud-monitoring-dashboards
google-cloud-monitoring-metrics-scopes
google-cloud-network-connectivity
google-cloud-network-management
google-cloud-network-security
google-cloud-network-services
google-cloud-notebooks
google-cloud-optimization
google-cloud-orchestration-airflow

Python > Documentation > Reference

Was this helpful?

Version latest

Send feedback

Python Client for Natural Language API

support stable | pypi v2.10.0 | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11

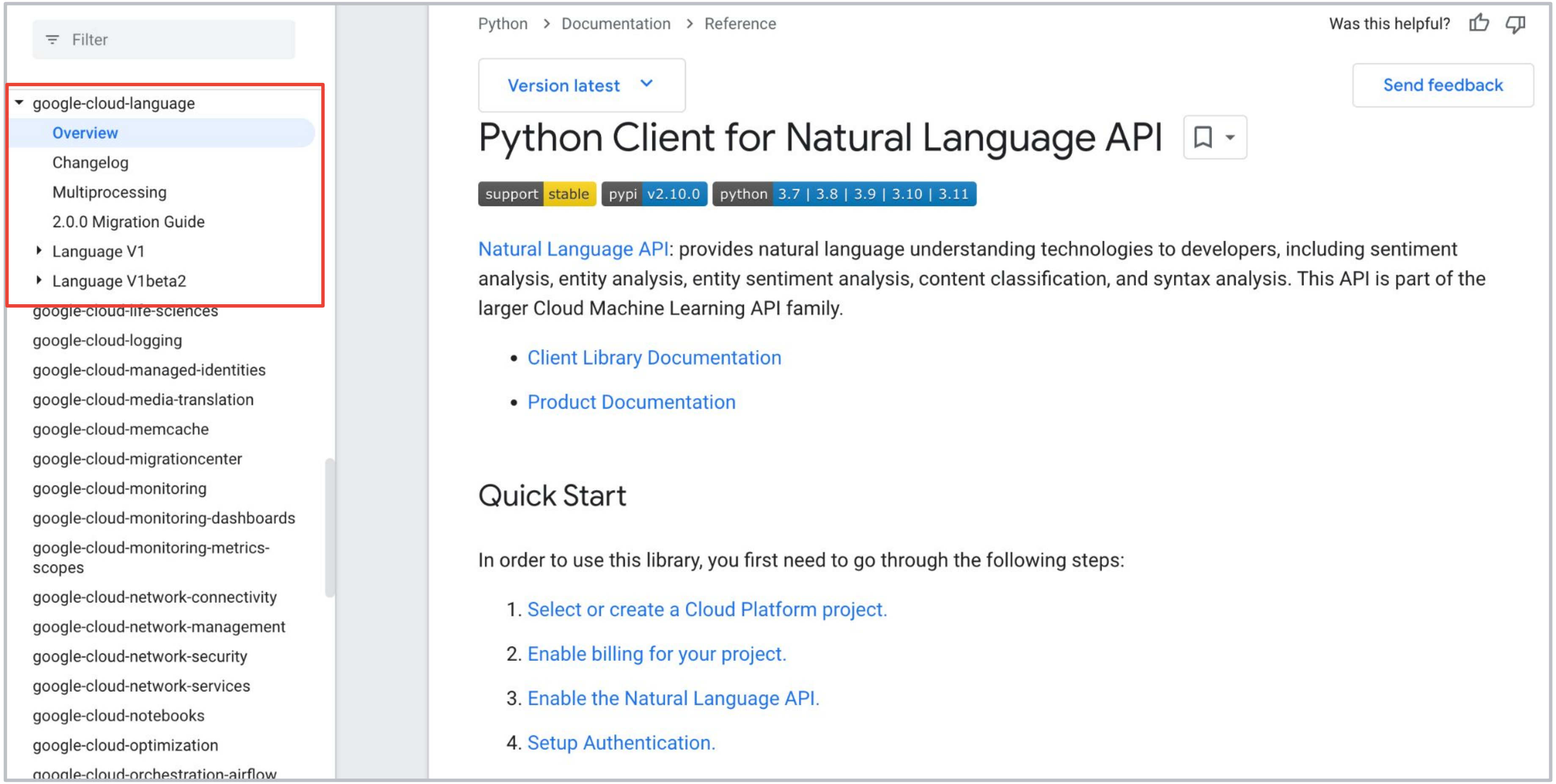
Natural Language API: provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

- Client Library Documentation
- Product Documentation

Quick Start

In order to use this library, you first need to go through the following steps:

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Enable the Natural Language API.
4. Setup Authentication.



Filter
google-cloud-language
Overview
Changelog
Multiprocessing
2.0.0 Migration Guide
Language V1
Language V1beta2
google-cloud-life-sciences
google-cloud-logging
google-cloud-managed-identities
google-cloud-media-translation
google-cloud-memcache
google-cloud-migrationcenter
google-cloud-monitoring
google-cloud-monitoring-dashboards
google-cloud-monitoring-metrics-scopes
google-cloud-network-connectivity
google-cloud-network-management
google-cloud-network-security
google-cloud-network-services
google-cloud-notebooks
google-cloud-optimization
google-cloud-orchestration-airflow
Python > Documentation > Reference
Was this helpful?
Version latest
Send feedback
support stable
pypi v2.10.0
python 3.7 | 3.8 | 3.9 | 3.10 | 3.11

Python Client for Natural Language API

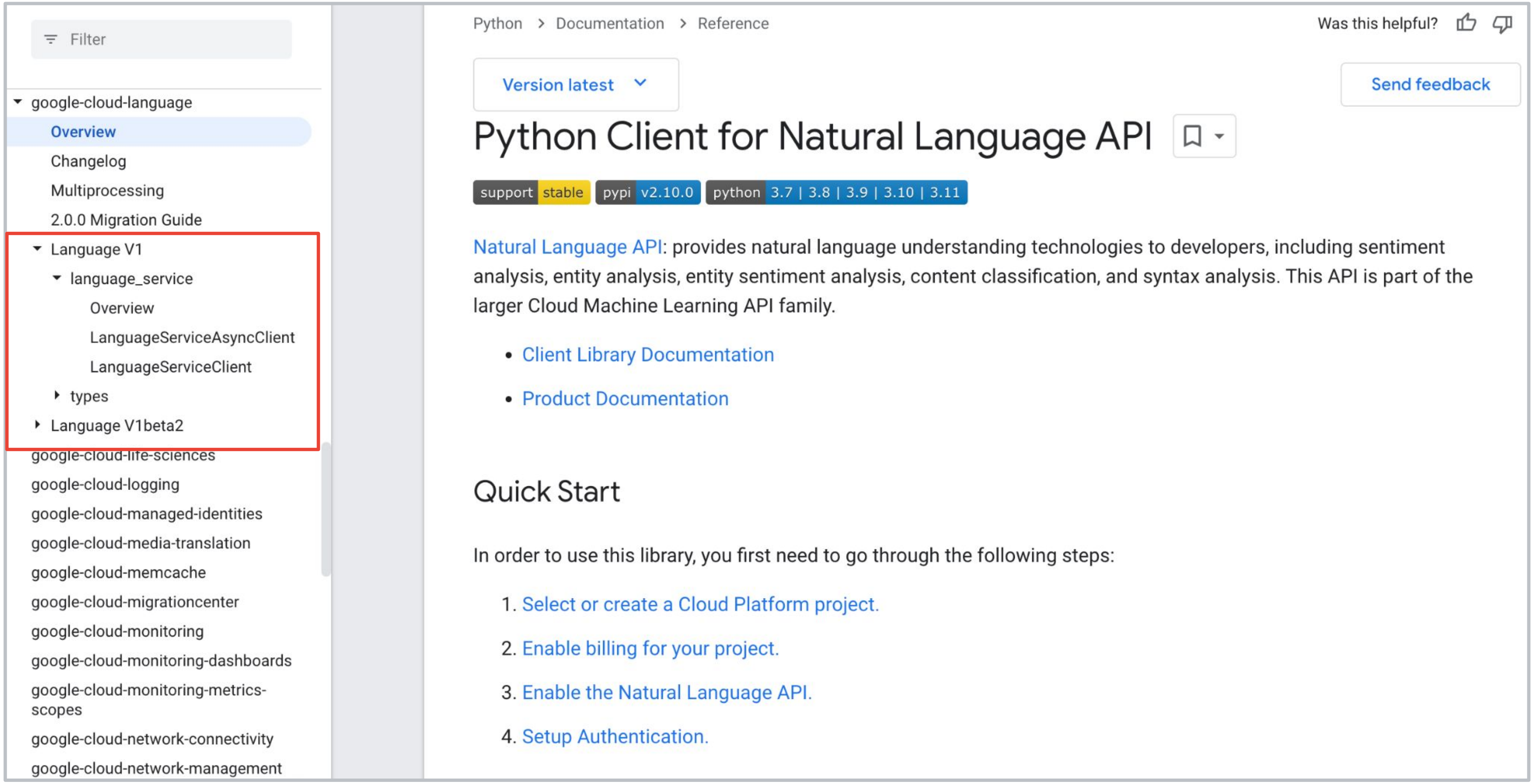
Natural Language API: provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

- Client Library Documentation
- Product Documentation

Quick Start

In order to use this library, you first need to go through the following steps:

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Enable the Natural Language API.
4. Setup Authentication.



Filter

- google-cloud-language
  - Overview
  - Changelog
  - Multiprocessing
  - 2.0.0 Migration Guide
  - Language V1
    - language_service
      - Overview
      - LanguageServiceAsyncClient
      - LanguageServiceClient
    - types
  - Language V1beta2
- google-cloud-life-sciences
- google-cloud-logging
- google-cloud-managed-identities
- google-cloud-media-translation
- google-cloud-memcache
- google-cloud-migrationcenter
- google-cloud-monitoring
- google-cloud-monitoring-dashboards
- google-cloud-monitoring-metrics-scopes
- google-cloud-network-connectivity
- google-cloud-network-management

Python > Documentation > Reference

Was this helpful?

Version latest

Send feedback

Python Client for Natural Language API

support stable | pypi v2.10.0 | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11

Natural Language API: provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

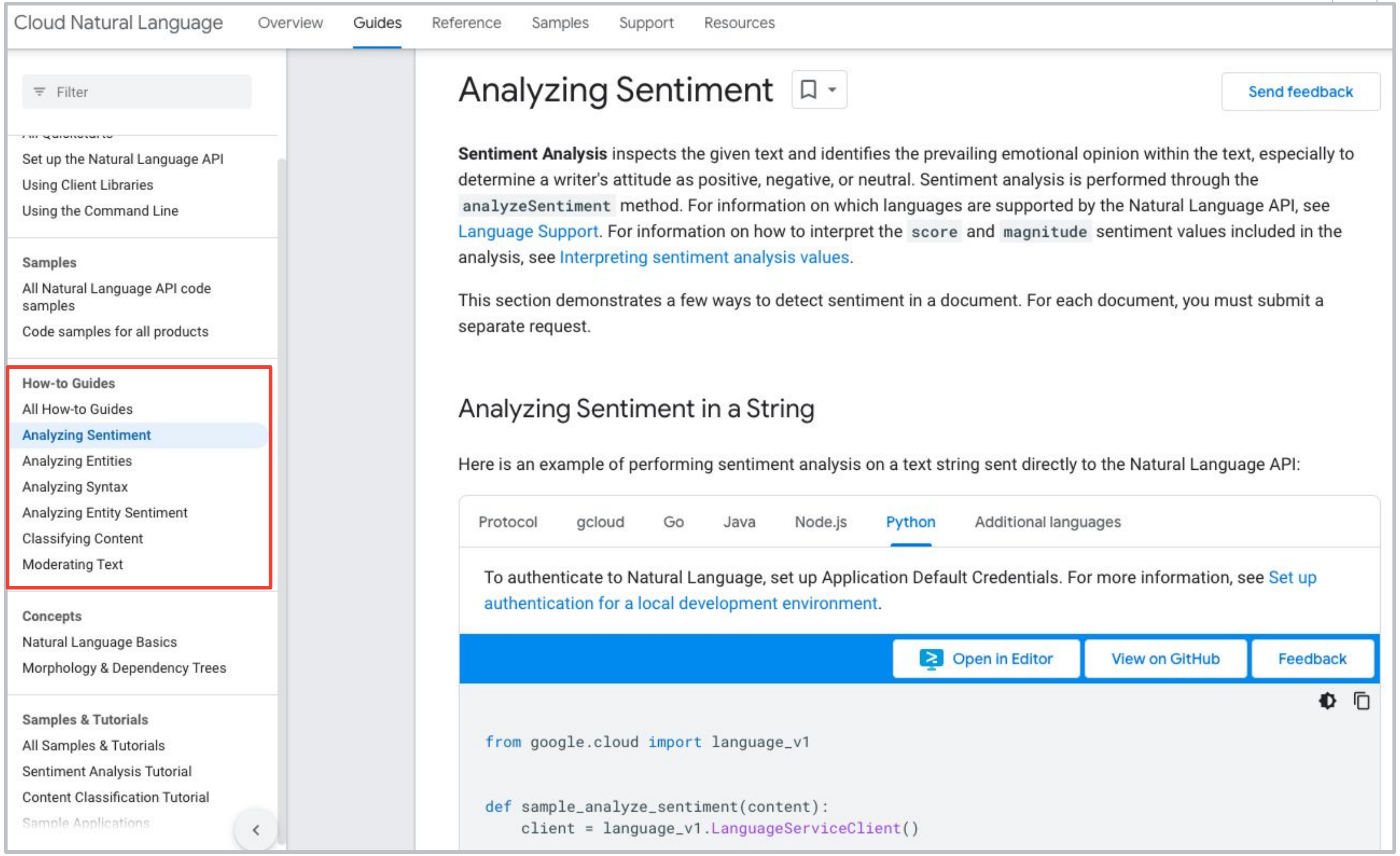
- Client Library Documentation
- Product Documentation

Quick Start

In order to use this library, you first need to go through the following steps:

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Enable the Natural Language API.
4. Setup Authentication.

Google Cloud



Analyzing Sentiment

Sentiment Analysis inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a writer's attitude as positive, negative, or neutral. Sentiment analysis is performed through the `analyzeSentiment` method. For information on which languages are supported by the Natural Language API, see Language Support. For information on how to interpret the `score` and `magnitude` sentiment values included in the analysis, see Interpreting sentiment analysis values.

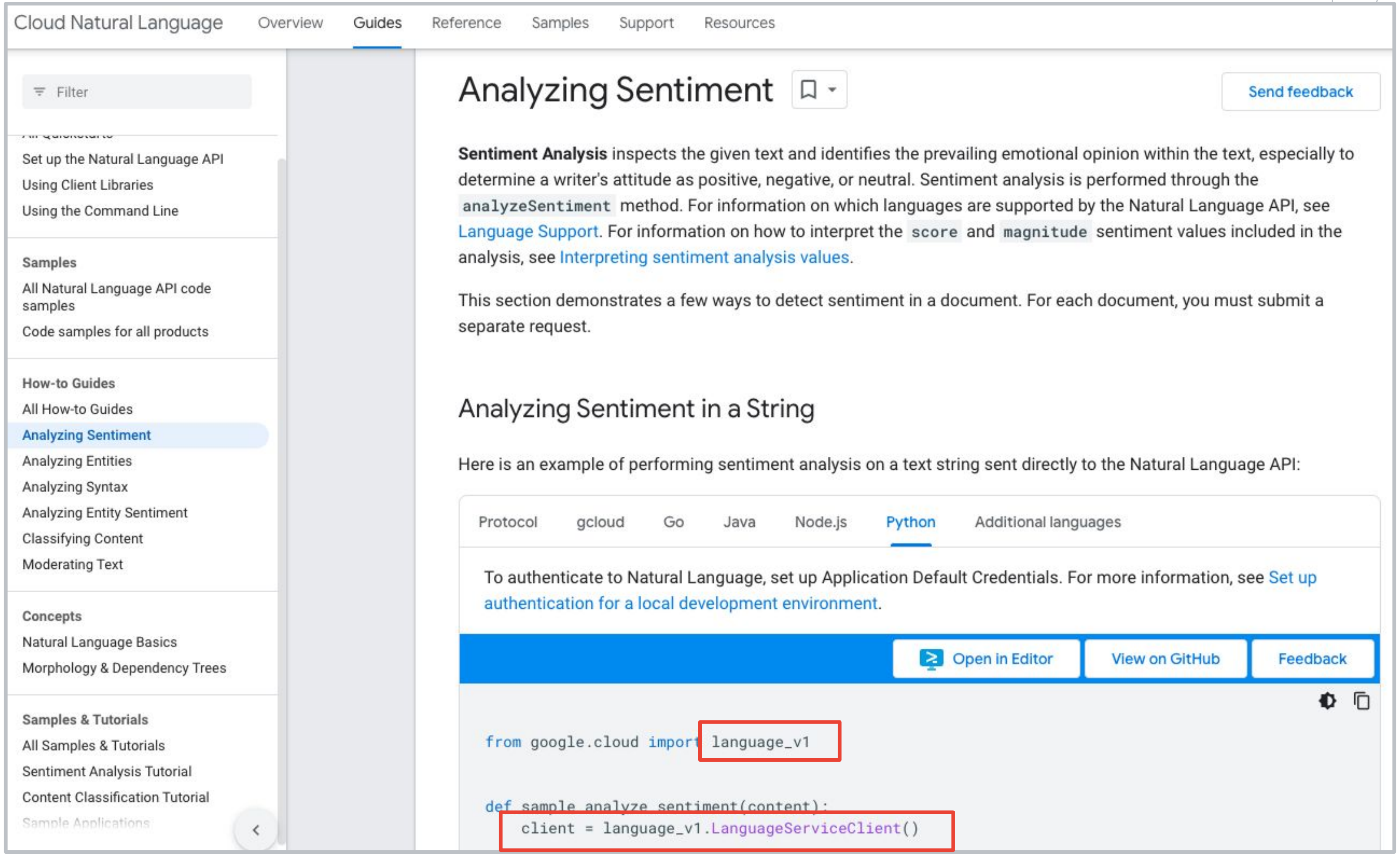
This section demonstrates a few ways to detect sentiment in a document. For each document, you must submit a separate request.

Analyzing Sentiment in a String

Here is an example of performing sentiment analysis on a text string sent directly to the Natural Language API:

To authenticate to Natural Language, set up Application Default Credentials. For more information, see Set up authentication for a local development environment.

```
from google.cloud import language_v1


def sample_analyze_sentiment(content):
    client = language_v1.LanguageServiceClient()
```



Set up the Natural Language API
Using Client Libraries
Using the Command Line

Samples
All Natural Language API code samples
Code samples for all products

How-to Guides
All How-to Guides
Analyzing Sentiment
Analyzing Entities
Analyzing Syntax
Analyzing Entity Sentiment
Classifying Content
Moderating Text

Concepts
Natural Language Basics
Morphology & Dependency Trees

Samples & Tutorials
All Samples & Tutorials
Sentiment Analysis Tutorial
Content Classification Tutorial
Sample Applications

Analyzing Sentiment

Sentiment Analysis inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a writer's attitude as positive, negative, or neutral. Sentiment analysis is performed through the `analyzeSentiment` method. For information on which languages are supported by the Natural Language API, see Language Support. For information on how to interpret the `score` and `magnitude` sentiment values included in the analysis, see Interpreting sentiment analysis values.

This section demonstrates a few ways to detect sentiment in a document. For each document, you must submit a separate request.

Analyzing Sentiment in a String

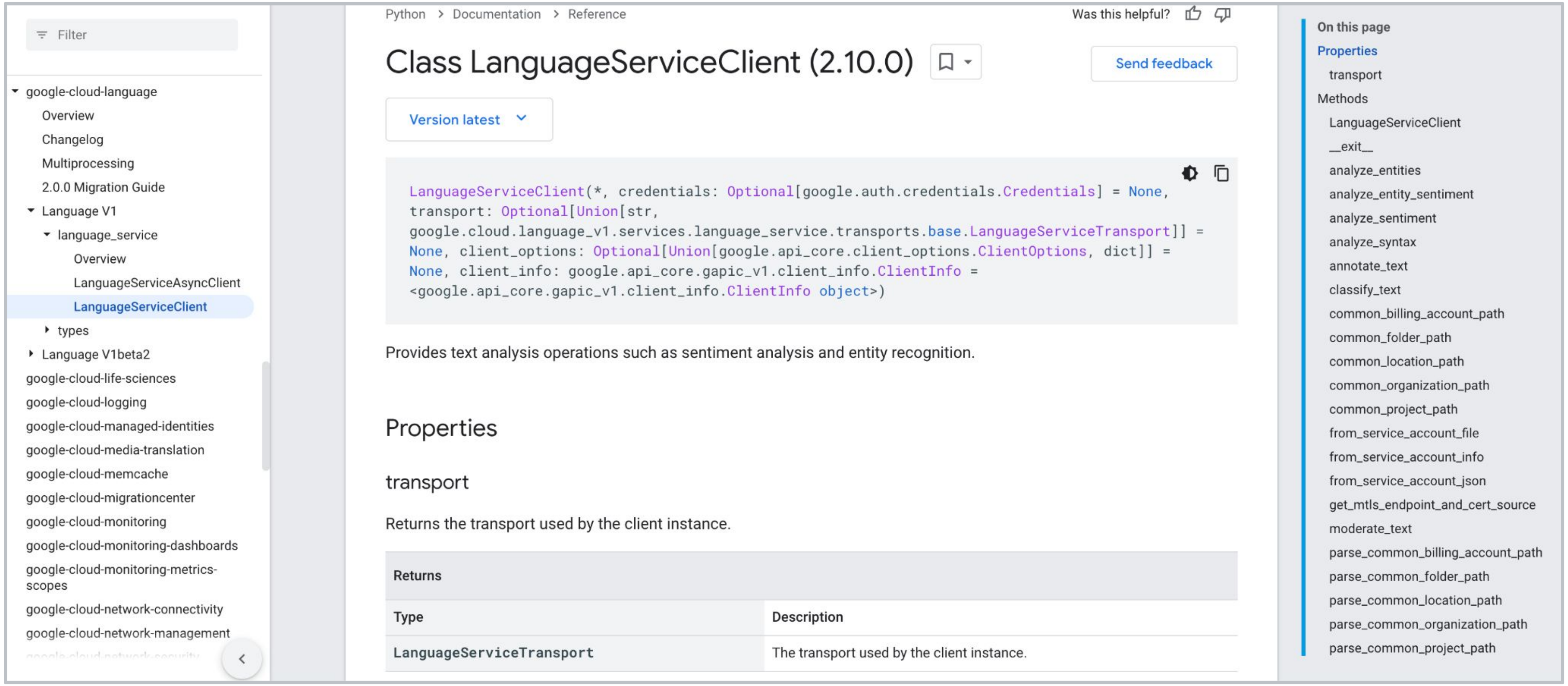
Here is an example of performing sentiment analysis on a text string sent directly to the Natural Language API:

To authenticate to Natural Language, set up Application Default Credentials. For more information, see Set up authentication for a local development environment.

```
from google.cloud import language_v1

def sample_analyze_sentiment(content):
    client = language_v1.LanguageServiceClient()
```

Google Cloud

Client



Filter
google-cloud-language
Overview
Changelog
Multiprocessing
2.0.0 Migration Guide
Language V1
language_service
Overview
LanguageServiceAsyncClient
LanguageServiceClient
types
Language V1beta2
google-cloud-life-sciences
google-cloud-logging
google-cloud-managed-identities
google-cloud-media-translation
google-cloud-memcache
google-cloud-migrationcenter
google-cloud-monitoring
google-cloud-monitoring-dashboards
google-cloud-monitoring-metrics-scopes
google-cloud-network-connectivity
google-cloud-network-management
Python > Documentation > Reference
Was this helpful?
Version latest
Send feedback

Class LanguageServiceClient (2.10.0)

```
LanguageServiceClient(*, credentials: Optional[google.auth.credentials.Credentials] = None,
transport: Optional[Union[str,
google.cloud.language_v1.services.language_service.transports.base.LanguageServiceTransport]] =
None, client_options: Optional[Union[google.api_core.client_options.ClientOptions, dict]] =
None, client_info: google.api_core.gapic_v1.client_info.ClientInfo =
<google.api_core.gapic_v1.client_info.ClientInfo object>)
```

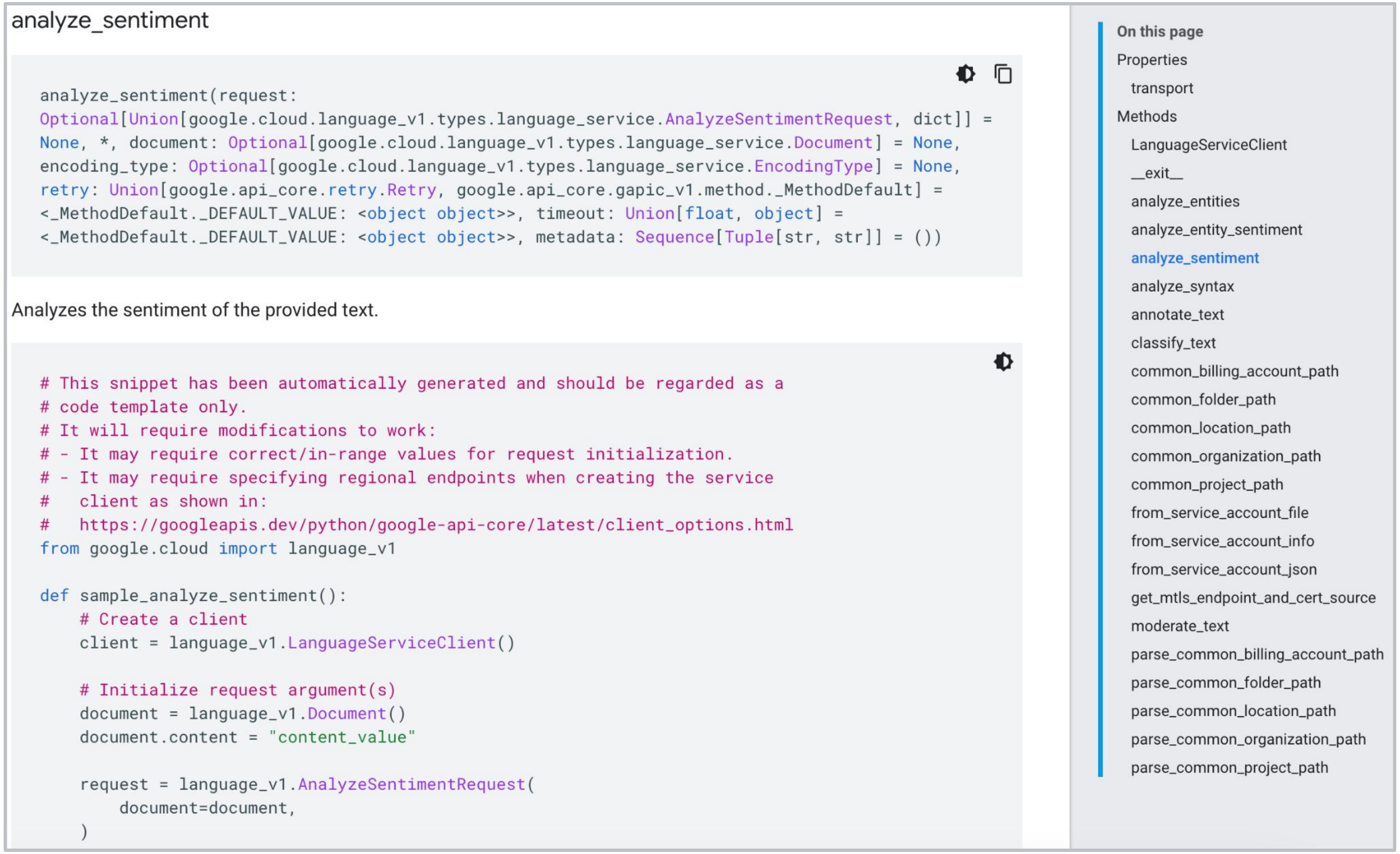
Provides text analysis operations such as sentiment analysis and entity recognition.

Properties

transport

Returns the transport used by the client instance.

Returns	
Type	**Description**
`LanguageServiceTransport`	The transport used by the client instance.



analyze_sentiment

```
analyze_sentiment(request:
Optional[Union[google.cloud.language_v1.types.language_service.AnalyzeSentimentRequest, dict]] =
None, *, document: Optional[google.cloud.language_v1.types.language_service.Document] = None,
encoding_type: Optional[google.cloud.language_v1.types.language_service.EncodingType] = None,
retry: Union[google.api_core.retry.Retry, google.api_core.gapic_v1.method._MethodDefault] =
<_MethodDefault._DEFAULT_VALUE: <object object>>, timeout: Union[float, object] =
<_MethodDefault._DEFAULT_VALUE: <object object>>, metadata: Sequence[Tuple[str, str]] = ())
```

Analyzes the sentiment of the provided text.

```
# This snippet has been automatically generated and should be regarded as a
# code template only.
# It will require modifications to work:
# - It may require correct/in-range values for request initialization.
# - It may require specifying regional endpoints when creating the service
#   client as shown in:
#   https://googleapis.dev/python/google-api-core/latest/client_options.html
from google.cloud import language_v1

def sample_analyze_sentiment():
    # Create a client
    client = language_v1.LanguageServiceClient()

    # Initialize request argument(s)
    document = language_v1.Document()
    document.content = "content_value"

    request = language_v1.AnalyzeSentimentRequest(
        document=document,
    )
```

On this page

Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "gcs_content_uri": "gs://my-bucket/my-text-object"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

Python client: Analyze sentiment

```
from google.cloud import language_v1


client = language_v1.LanguageServiceClient()


type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}


response = client.analyze_sentiment(request={"document": document})


sentiment = response.document_sentiment
```

Python client: Analyze sentiment

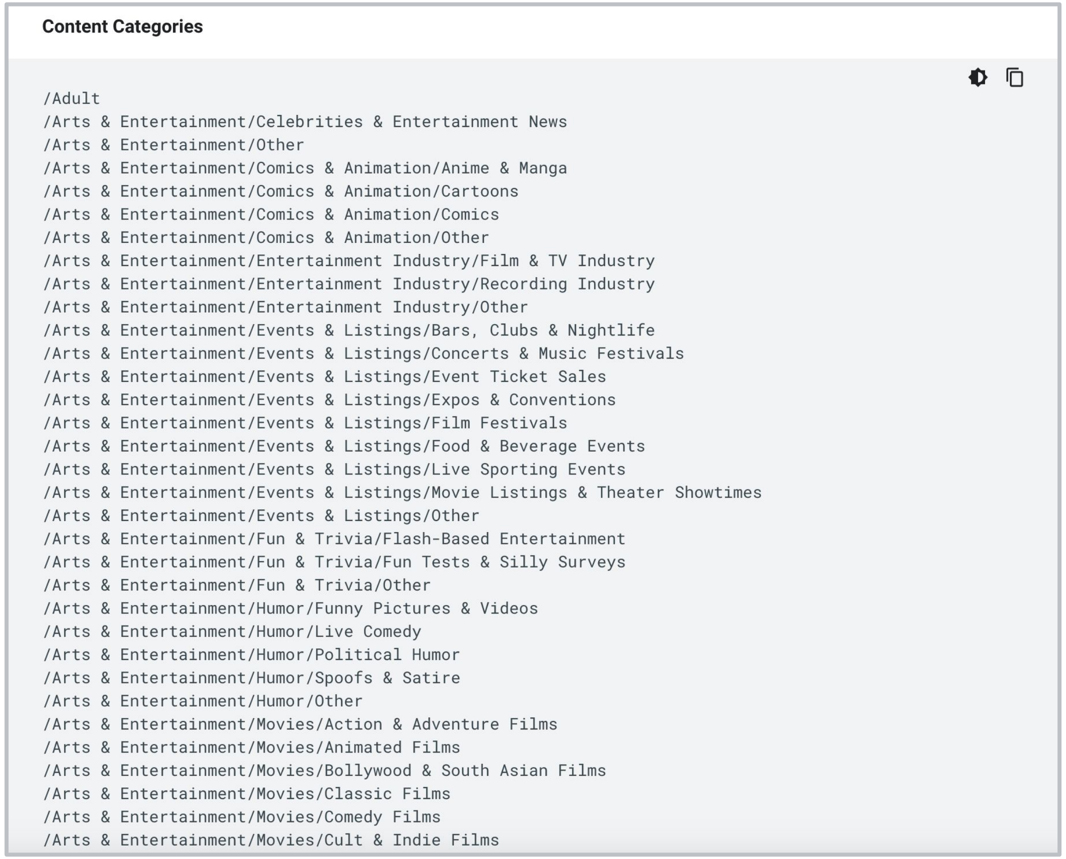
```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

Python client: Analyze sentiment

```python
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

Task-specific Solutions

Natural Language API: Classify Text



Content Categories

```
/Adult
/Arts & Entertainment/Celebrities & Entertainment News
/Arts & Entertainment/Other
/Arts & Entertainment/Comics & Animation/Anime & Manga
/Arts & Entertainment/Comics & Animation/Cartoons
/Arts & Entertainment/Comics & Animation/Comics
/Arts & Entertainment/Comics & Animation/Other
/Arts & Entertainment/Entertainment Industry/Film & TV Industry
/Arts & Entertainment/Entertainment Industry/Recording Industry
/Arts & Entertainment/Entertainment Industry/Other
/Arts & Entertainment/Events & Listings/Bars, Clubs & Nightlife
/Arts & Entertainment/Events & Listings/Concerts & Music Festivals
/Arts & Entertainment/Events & Listings/Event Ticket Sales
/Arts & Entertainment/Events & Listings/Expos & Conventions
/Arts & Entertainment/Events & Listings/Film Festivals
/Arts & Entertainment/Events & Listings/Food & Beverage Events
/Arts & Entertainment/Events & Listings/Live Sporting Events
/Arts & Entertainment/Events & Listings/Movie Listings & Theater Showtimes
/Arts & Entertainment/Events & Listings/Other
/Arts & Entertainment/Fun & Trivia/Flash-Based Entertainment
/Arts & Entertainment/Fun & Trivia/Fun Tests & Silly Surveys
/Arts & Entertainment/Fun & Trivia/Other
/Arts & Entertainment/Humor/Funny Pictures & Videos
/Arts & Entertainment/Humor/Live Comedy
/Arts & Entertainment/Humor/Political Humor
/Arts & Entertainment/Humor/Spoofs & Satire
/Arts & Entertainment/Humor/Other
/Arts & Entertainment/Movies/Action & Adventure Films
/Arts & Entertainment/Movies/Animated Films
/Arts & Entertainment/Movies/Bollywood & South Asian Films
/Arts & Entertainment/Movies/Classic Films
/Arts & Entertainment/Movies/Comedy Films
/Arts & Entertainment/Movies/Cult & Indie Films
```

Natural Language API: Classify Text

My waiter, Robert, provided an excellent experience.

The lobster eggs benedict was delicious! My compliments to the chef.

I would have liked to see the market price for the lobster before purchase, though. Please add that online!

Natural Language API: Classify Text

Food/Meat & Seafood	0.92
Food/Breakfast Foods	0.84
Hospitality Industry/Food Service	0.78

AutoML Text

My waiter, Robert, provided an excellent experience.

The lobster eggs benedict was delicious! My compliments to the chef.

I would have liked to see the market price for the lobster before purchase, though. Please add that online!

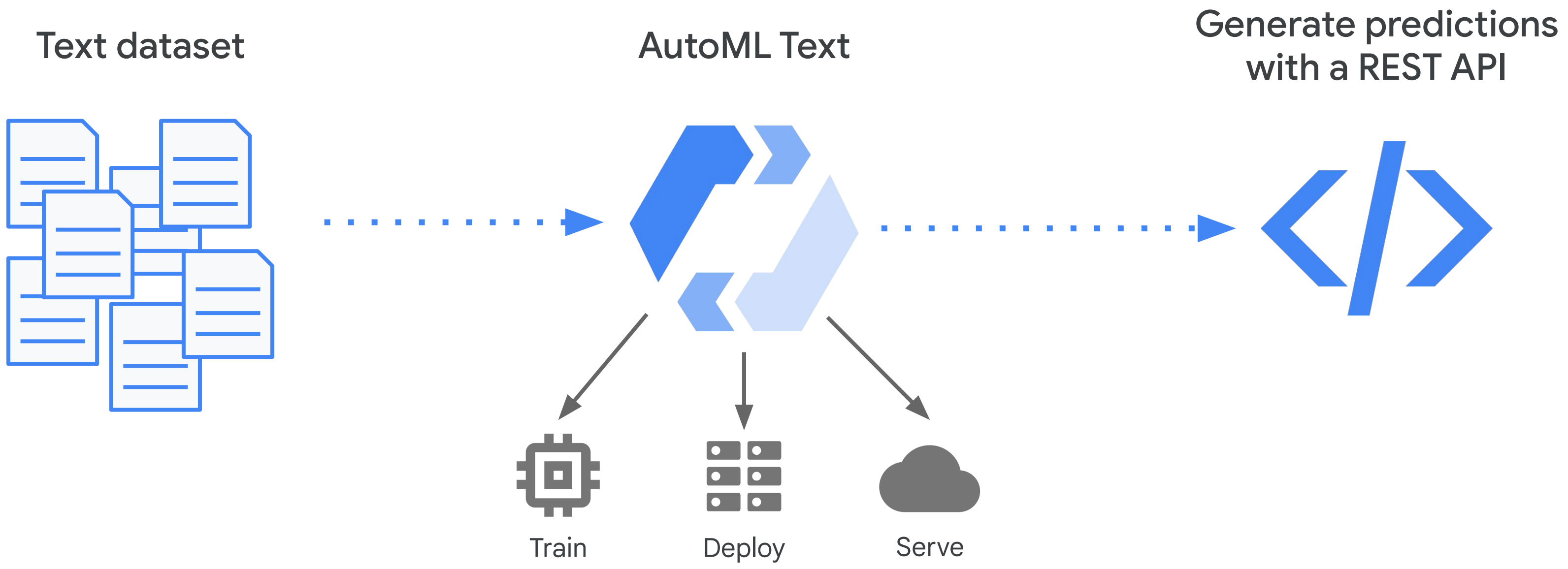
Natural Language API: Classify Text

Food/Meat & Seafood	0.92
Food/Breakfast Foods	0.84
Hospitality Industry/Food Service	0.78

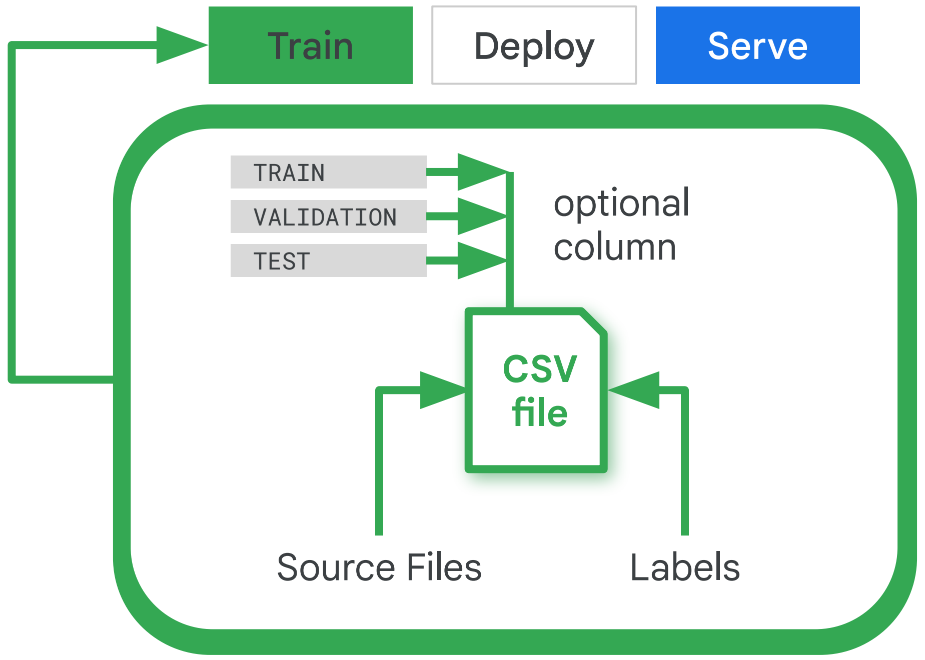
AutoML Text

Great Service	0.95
Happy Customer	0.92
Suggestion	0.8

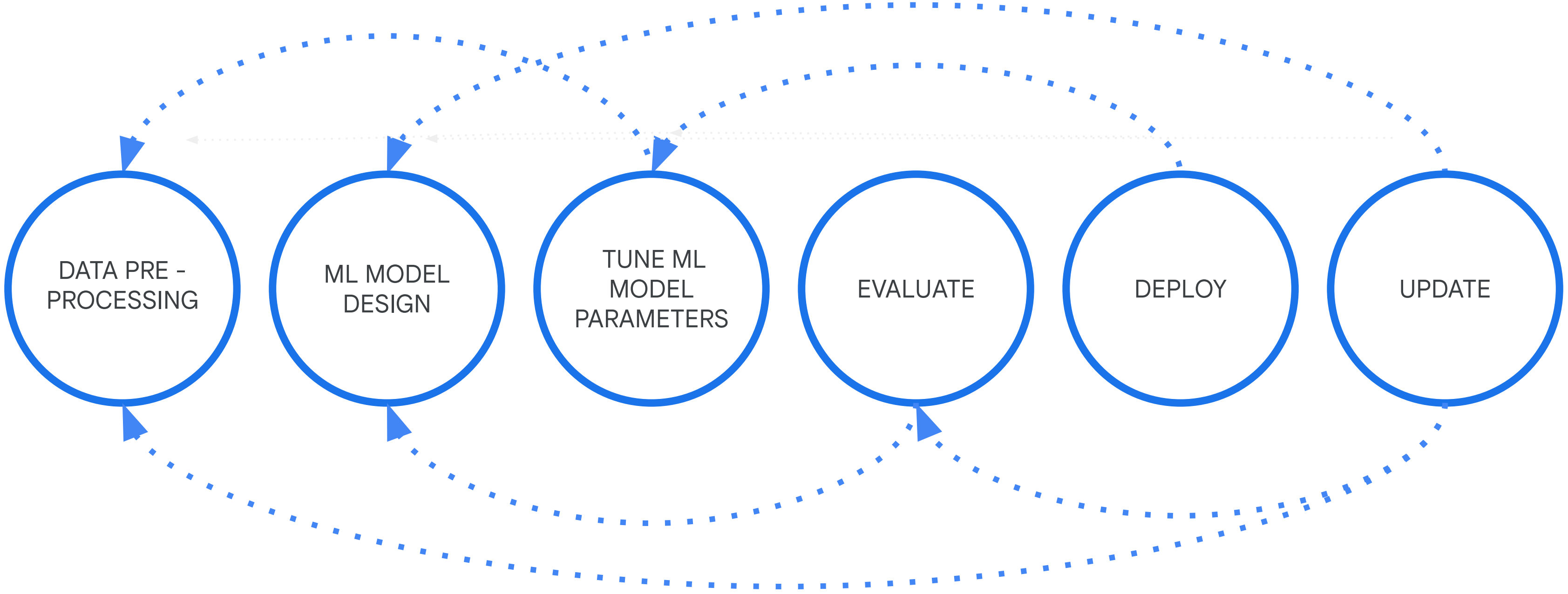
AutoML

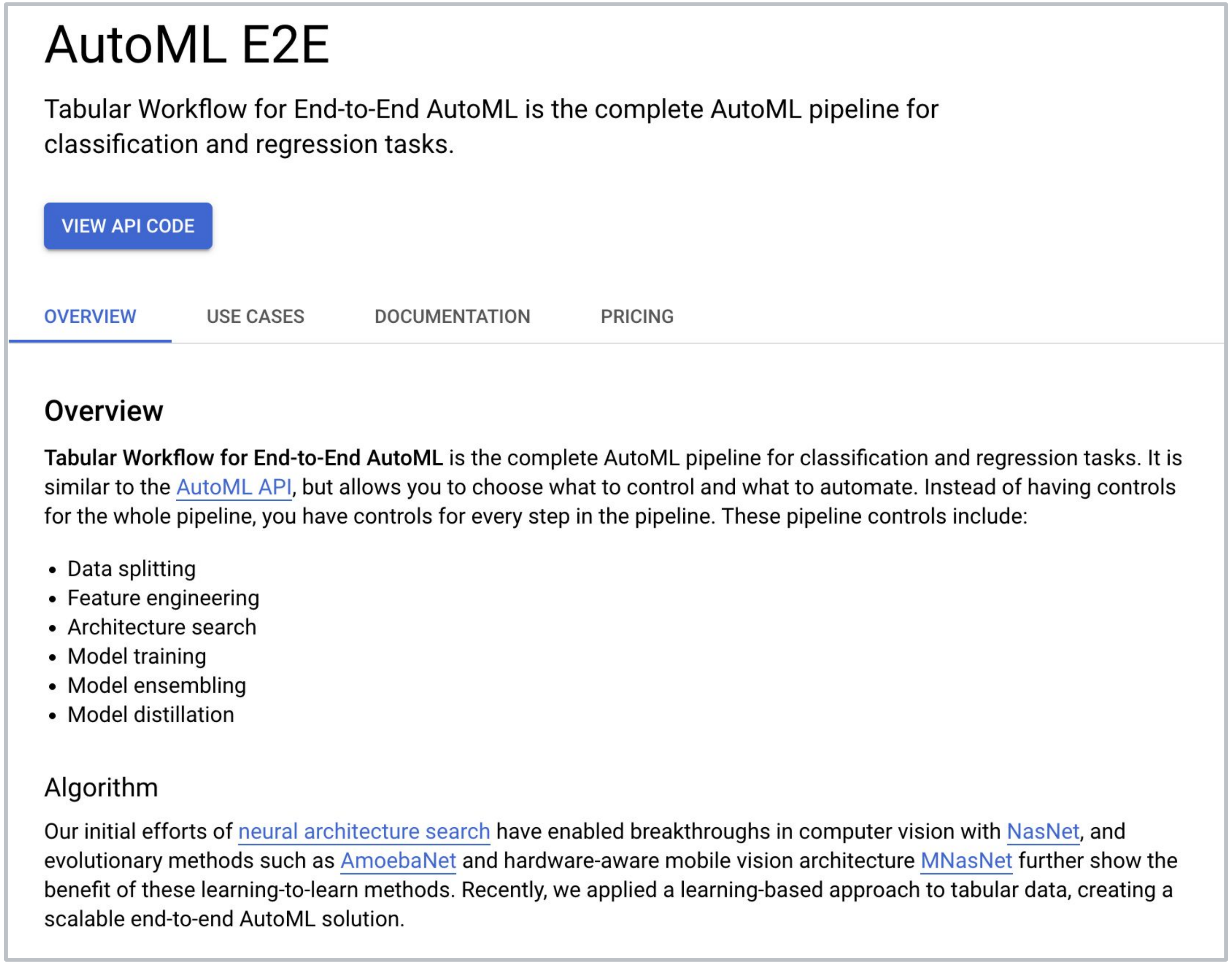

Text dataset
AutoML Text
Generate predictions with a REST API
Train
Deploy
Serve

AutoML Prepared Dataset Format



Codeless model building with AutoML





AutoML E2E

Tabular Workflow for End-to-End AutoML is the complete AutoML pipeline for classification and regression tasks.

Overview

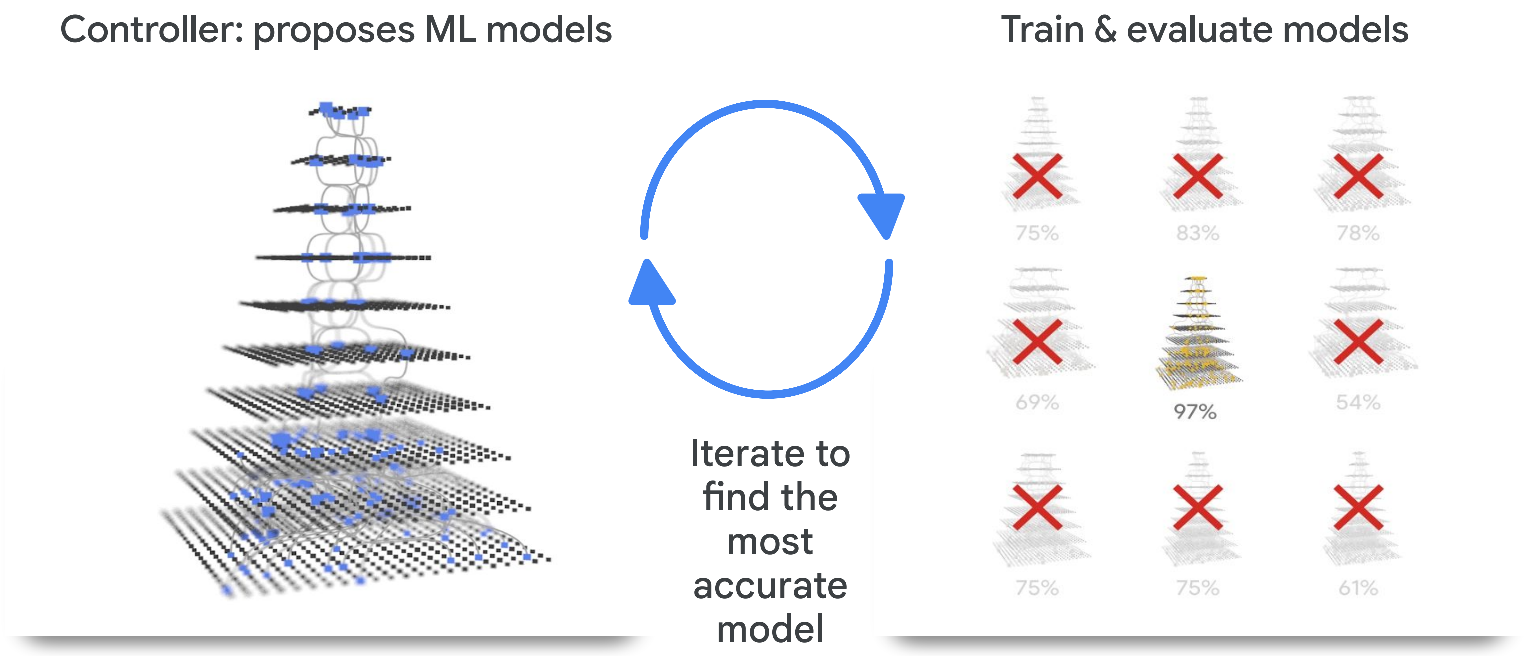
Tabular Workflow for End-to-End AutoML is the complete AutoML pipeline for classification and regression tasks. It is similar to the AutoML API, but allows you to choose what to control and what to automate. Instead of having controls for the whole pipeline, you have controls for every step in the pipeline. These pipeline controls include:

- Data splitting
- Feature engineering
- Architecture search
- Model training
- Model ensembling
- Model distillation

Algorithm

Our initial efforts of neural architecture search have enabled breakthroughs in computer vision with NasNet, and evolutionary methods such as AmoebaNet and hardware-aware mobile vision architecture MNasNet further show the benefit of these learning-to-learn methods. Recently, we applied a learning-based approach to tabular data, creating a scalable end-to-end AutoML solution.

AutoML is built with Neural Architecture Search


Controller: proposes ML models
Train & evaluate models
75%
83%
78%
69%
97%
54%
75%
75%
61%
Iterate to find the most accurate model

Task-specific Solutions

01	Ways to use task-specific models
02	Using the Python SDK
03	AutoML
04	**Lab: Text Classification with Model Garden**

Lab:
Text Classification with Model Garden