Start Lab 01:07:30

Entity and Sentiment Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Previous Next

Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Analysis with the Natural

Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

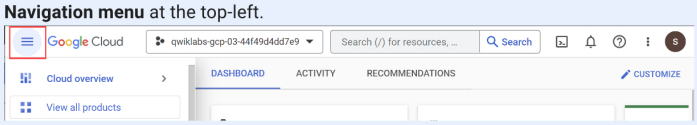
GSP038

Analysis with the Natural Language API

Lab 45 minutes 5 Credits Intermediate

GSP038

Note: To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left.



Task 1. Create an API key

Since you use `curl` to send a request to the Natural Language API, you must generate an API key to pass in your request URL.

Services > **Credentials**.

2. Click **Create credentials** and select **API key**.
3. Copy the generated API key and click **Close**.

Click **Check my progress** to verify the objective.

Create an API Key

Check my progress

In order to perform next steps please connect to the instance provisioned for you via **SSH**.

1. Click **Navigation menu** > **Compute Engine**. You should see the provisioned linux

2. Click on the **SSH** button. You will be brought to an interactive shell.
3. In the command line, enter in the following, replacing `<YOUR_API_KEY>` with the key you just copied:

```
export API_KEY=<YOUR_API_KEY>
```

Task 2. Make an entity analysis request

the API can extract entities (like people, places, and events) from text. To try it out the API's entity analysis, use the following sentence:

Joanne Rowling, who writes under the pen names J. K. Rowling and Robert Galbraith, is a British novelist and screenwriter who wrote the Harry Potter fantasy series.

You build your request to the Natural Language API in the file, `request.json`.

1. Use nano (a code editor) to create the file `request.json`:

```
nano request.json
```

2. Type or paste the following code into `request.json`:

```
{
  "document":{
    "content":"Joanne Rowling, who writes under the pen names J.
K. Rowling and Robert Galbraith, is a British novelist and
screenwriter who wrote the Harry Potter fantasy series."
  },
  "encodingType":"UTF8"
}
```

3. Press **CTRL+X** to exit nano, then **Y** to save the file, then **ENTER** to confirm.

In the request, you're telling the Natural Language API about the text being sent. Supported type values are `PLAIN_TEXT` or `HTML`. In content, you pass the text to send to the Natural Language API for analysis.

The Natural Language API also supports sending files stored in Cloud Storage for text processing. If you wanted to send a file from Cloud Storage, you would replace `content` with `gcsContentUri` and give it a value of the text file's uri in Cloud Storage.

Click **Check my progress** to verify the objective.

Make an Entity Analysis Request

Check my progress

Task 3. Call the Natural Language API

1. You can now pass your request body, along with the API key environment variable (all in one single command line):

```
curl
"https://language.googleapis.com/v1/documents:analyzeEntities?
key=${API_KEY}" \
```

```
  -s -X POST -H "Content-Type: application/json" --data-binary
@request.json > result.json
```

2. In order to check the response run:

```
cat result.json
```

The beginning of your response should look like this:

```
          },
          "type": "PROPER"
        },
        {
          "text": {
            "content": "Rowling",
            "beginOffset": 53
          },
          "type": "PROPER"
        },
        {
          "text": {
            "content": "novelist",
            "beginOffset": 96
          },
          "type": "COMMON"
        },
        {
```

```
            "beginOffset": 65
          },
          "type": "PROPER"
        }
      ]
    },

    ...
  ]
}
```

] }

For each entity in the response, you get the entity `type`, the associated Wikipedia URL if there is one, the `salience`, and the indices of where this entity appeared in the text. Salience is a number in the [0,1] range that refers to the centrality of the entity to the text as a whole.

ways. Take a look at the `mentions` list in the response: the API is able to tell that "Joanne Rowling", "Rowling", "novelist" and "Robert Galbriath" all point to the same thing.

Click **Check my progress** to verify the objective.

Check the Entity Analysis response

Check my progress

Language API

In addition to extracting entities, the Natural Language API also lets you perform sentiment analysis on a block of text. This JSON request will include the same parameters as the request above, but this time change the text to include something with a stronger sentiment.

1. Use nano to replace the code in `request.json` with the following, and feel free to replace the `content` below with your own text:

```
{
 "document":{
   "type":"PLAIN_TEXT",
   "content":"Harry Potter is the best book. I think everyone
should read it."
 },
```

2. Press **CTRL+X** to exit nano, then **Y** to save the file, then **ENTER** to confirm.

3. Next you send the request to the API's `analyzeSentiment` endpoint:

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

Your response should look like this:

```
{
  "documentSentiment": {
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "Harry Potter is the best book.",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0.9,
        "score": 0.9
      }
    },
    {
      "text": {
        "content": "I think everyone should read it.",
        "beginOffset": 31
      },
        "score": 0.9
      }
    }
  ]
}
```

Note: Don't be alarmed if your scores differ slightly than the example output.

Notice that you get two types of sentiment values: sentiment for the document as a whole, and sentiment broken down by sentence. The sentiment method returns two

values:

- `score` - is a number from -1.0 to 1.0 indicating how positive or negative the statement is.
- `magnitude` - is a number ranging from 0 to infinity that represents the weight of

Longer blocks of text with heavily weighted statements have higher magnitude values. The score for the first sentence is positive (0.7), whereas the score for the second sentence is neutral (0.1).

Task 5. Analyzing entity sentiment

In addition to providing sentiment details on the entire text document, the Natural Language API can also break down sentiment by the entities in the text. Use this sentence as an example:

I liked the sushi but the service was terrible

In this case, getting a sentiment score for the entire sentence as you did above might not be so useful. If this was a restaurant review and there were hundreds of reviews for the same restaurant, you'd want to know exactly which things people liked and didn't like in their reviews. Fortunately, the Natural Language API has a method that lets you get the sentiment for each entity in the text, called `analyzeEntitySentiment`. Let's see how it works!

1. Use nano to update `request.json` with the sentence below:

```
{
 "document":{
   "type":"PLAIN_TEXT",
   "content":"I liked the sushi but the service was terrible."
 },
 "encodingType": "UTF8"
}
```

3. Then call the `analyzeEntitySentiment` endpoint with the following curl command:

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

In the response you get back two entity objects: one for "sushi" and one for "service". Here's the full JSON response:

```
{
  "entities": [
    {
      "type": "CONSUMER_GOOD",
      "metadata": {},
      "salience": 0.51064336,
      "mentions": [
        {
          "text": {
            "content": "sushi",
            "beginOffset": 12
```

You can see that the score returned for "sushi" was a neutral score of 0, whereas "service" got a score of -0.7. Cool! You also may notice that there are two sentiment objects returned for each entity. If either of these terms were mentioned more than once, the API would return a different sentiment score and magnitude for each mention, along with an aggregate sentiment for the entity.

Note: Don't be alarmed if your scores differ slightly than the example output.

Task 6. Analyzing syntax and parts of speech

Use **syntactic analysis**, another of the Natural Language API's methods, to dive deeper into the linguistic details of the text. `analyzeSyntax` extracts linguistic information, breaking up the given text into a series of sentences and tokens (generally, word boundaries), to provide further analysis on those tokens. For each word in the text, the API tells you the word's part of speech (noun, verb, adjective, etc.) and how it relates to other words in the sentence (Is it the root verb? A modifier?).

Try it out with a simple sentence. This JSON request will be similar to the ones above, with the addition of a features key. This tells the API to perform syntax annotation.

```
{
  "document":{
    "type":"PLAIN_TEXT",
    "content": "Joanne Rowling is a British novelist,
screenwriter and film producer."
  },
  "encodingType": "UTF8"
}
```

2. Press **CTRL+X** to exit nano, then **Y** to save the file, then **ENTER** to confirm.

3. Then call the API's `analyzeSyntax` method:

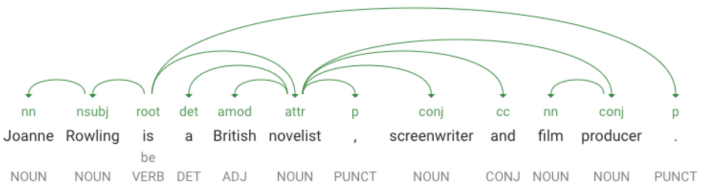
```
curl
  -s -X POST -H "Content-Type: application/json" --data-binary
@request.json
```

The response should return an object like the one below for each token in the sentence:

```
{
      "text": {
        "content": "is",
        "beginOffset": 15
      },
      "partOfSpeech": {
        "tag": "VERB",
        "aspect": "ASPECT_UNKNOWN",
        "case": "CASE_UNKNOWN",
        "form": "FORM_UNKNOWN",
        "gender": "GENDER_UNKNOWN",
        "mood": "INDICATIVE",
        "person": "THIRD",
        "proper": "PROPER_UNKNOWN",
        "reciprocity": "RECIPROCITY_UNKNOWN",
        "tense": "PRESENT",
        "voice": "VOICE_UNKNOWN"
      },
      "dependencyEdge": {
        "headTokenIndex": 2,
        "label": "ROOT"
      },
      "lemma": "be"
    },
```

Let's break down the response:

- `partOfSpeech` tells you that "Joanne" is a noun.
- `dependencyEdge` includes data that you can use to create a dependency parse tree of the text. Essentially, this is a diagram showing how words in a sentence relate to



Note: You can create your own dependency parse trees in the browser with the Natural Language demo available in the Natural Language AI Guide

- `headTokenIndex` is the index of the token that has an arc pointing at "Joanne". Think of each token in the sentence as a word in an array.

- `headTokenIndex` of 1 for "Joanne" refers to the word "Rowling", which it is connected to in the tree. The label `NN` (short for noun compound modifier)

- `lemma` is the canonical form of the word. For example, the words *run*, *runs*, *ran*, and *running* all have a lemma of *run*. The lemma value is useful for tracking occurrences of a word in a large piece of text over time.

Task 7. Multilingual natural language processing

The Natural Language API also supports languages other than English (full list can be found in the Language Support Guide).

1. Modify the code in `request.json` with a sentence in Japanese:

```
{
  "document":{
    "type":"PLAIN_TEXT",
    "content":"日本のグーグルのオフィスは、東京の六本木ヒルズにあります"
  }
}
```

2. Press **CTRL+X** to exit nano, then **Y** to save the file, then **ENTER** to confirm.

Notice that you didn't tell the API which language the text is, it can automatically detect it!

3. Next, you send it to the `analyzeEntities` endpoint:

```
curl
"https://language.googleapis.com/v1/documents:analyzeEntities?
@request.json
```

And you get the following response:

```
{
  "entities": [
    {
      "name": "日本",
      "type": "LOCATION",
      "metadata": {
        "mid": "/m/03_3d",
        "wikipedia_url": "https://en.wikipedia.org/wiki/Japan"
      },
      "salience": 0.23854347,
      "mentions": [
        {
          "beginOffset": 0
        },
        "type": "PROPER"
      }
    ]
  },
  {
    "name": "グーグル",
    "type": "ORGANIZATION",
```

```
      "metadata": {
        "mid": "/m/045c7b",
        "wikipedia_url": "https://en.wikipedia.org/wiki/Google"
      },
      "salience": 0.21155767,
      "mentions": [
        {
          "text": {
            "content": "グーグル",
```

```
          "type": "PROPER"
        }
      ]
    },
    ...
  ]
  "language": "ja"
}
```


The wikipedia URLs even point to the Japanese Wikipedia pages - so cool!

Congratulations!

You've learned how to perform text analysis with the Cloud Natural Language API by extracting entities, analyzing sentiment, and doing syntax annotation. In this lab, you created a Natural Language API request and called the API with `curl`, extracted entities and ran sentiment analysis on text with the Natural Language API, performed linguistic analysis on text, and created a Natural Language API request in a different language.

Next steps

- Check out the Natural Language API tutorials in the documentation.

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated February 14, 2024

Lab Last Tested October 13, 2023