This tutorial is designed to let you quickly start exploring and developing applications with the Google Cloud Natural Language API. It is designed for people familiar with basic programming, though even without much programming knowledge, you should be able to follow along. Having walked through this tutorial, you should be able to use the Reference documentation (/natural-language/docs/reference/rest/) to create your own basic applications.

This tutorial steps through a Natural Language API application using Python code. The purpose here is not to explain the Python client libraries, but to explain how to make calls to the Natural Language API. Applications in Java and Node.js are essentially similar. Consult the Natural Language API Samples (/natural-language/docs/samples) for samples in other languages (including this sample within the tutorial).

This tutorial has several prerequisites:

- You've set up a Cloud Natural Language API project (/natural-language/docs/getting-started#set_up_a_project) in the Google Cloud Console.
- You've set up your environment using Application Default Credentials (/natural-language/docs/common/auth#adc).
- You have basic familiarity with Python (https://www.python.org/) programming.
- You have set up your Python development environment. It is recommended that you have the latest version of Python, `pip`, and `virtualenv` installed on your system. For instructions, see the Python Development Environment Setup Guide (https://cloud.google.com/python/setup) for Google Cloud Platform.
- You've installed the Google Cloud Client Library for Python (/natural-language/docs/reference/libraries)

This tutorial walks you through a basic Natural Language API application, using an **`analyzeSentiment`** request, which performs sentiment analysis on text. Sentiment analysis attempts to determine the overall attitude (positive or negative) and is represented by numerical **`score`** and **`magnitude`** values. (For more information on these concepts, consult Natural Language Basics (/natural-language/docs/basics#sentiment-analysis-values).)

We'll show the entire code first. (Note that we have removed most comments from this code in order to show you how brief it is. We'll provide more comments as we walk through the code.)

To copy the code to your clipboard, click the copy widget that appears in the top-right of the code snippet when you hover over the code snippet. Copy this code into a **`sentiment_analysis.py`** file within your development directory.

For more information on installing and using the Google Cloud Natural Language Client Library for Python, see Natural Language API Client Libraries (https://cloud.google.com/natural-language/docs/reference/libraries).

thub.com/GoogleCloudPlatform/python-docs-samples/blob/master/language/sentiment/sentiment_analysis.py)

This simple application performs the following tasks:

- Imports the libraries necessary to run the application
- Takes a text file and passes it to the `main()` function
- Reads the text file and makes a request to the service
- Parses the response from the service and displays it to the user

We'll go over these steps in more detail below.

For more information on installing and using the Google Cloud Natural Language Client Library for Python, see Natural Language API Client Libraries

(https://cloud.google.com/natural-language/docs/reference/libraries).

thub.com/GoogleCloudPlatform/python-docs-samples/blob/master/language/sentiment/sentiment_analysis.py)

We import `argparse`, a standard library, to allow the application to accept input filenames as arguments.

For using the Cloud Natural Language API, we'll also want to import the `language` module from the `google-cloud-language` library. The `types` module contains classes that are required for creating requests.

thub.com/GoogleCloudPlatform/python-docs-samples/blob/master/language/sentiment/sentiment_analysis.py)

Here, we simply parse the passed argument for the text filename and pass it to the `analyze()` function.

Before communicating with the Natural Language API service, you need to authenticate your service using previously acquired credentials. Within an application, the simplest way to obtain credentials is to use Application Default Credentials (/natural-language/docs/common/auth#adc) (ADC). By default, ADC will attempt to obtain credentials from the `GOOGLE_APPLICATION_CREDENTIALS` environment file, which should be set to point to your service account's JSON key file. (You should have set up your service account and environment to use ADC in the Quickstart (/natural-language/docs/getting-started). See Setting Up a Service Account (/natural-language/docs/common/auth#set_up_a_service_account) for more information.)

The Google Cloud Client Library for Python automatically uses the application default credentials.

Now that our Natural Language API service is ready, we can access the service by calling the `analyze_sentiment` method of the `LanguageServiceClient` instance.

The client library encapsulates the details for requests and responses to the API. See the Natural Language API Reference (/natural-language/docs/reference/rest) for complete information on the specific structure of such a request.

For more information on installing and using the Google Cloud Natural Language Client Library for Python, see Natural Language API Client Libraries (https://cloud.google.com/natural-language/docs/reference/libraries).

thub.com/GoogleCloudPlatform/python-docs-samples/blob/master/language/sentiment/sentiment_analysis.py)

This code snippet performs the following tasks:

1. Instantiates a `LanguageServiceClient` instance as the client.
2. Reads the filename containing the text data into a variable.
3. Instantiates a `Document` object with the contents of the file.
4. Calls the client's `analyze_sentiment` method.

thub.com/GoogleCloudPlatform/python-docs-samples/blob/master/language/sentiment/sentiment_analysis.py)

We walk through the response to extract the sentiment `score` values for each sentence, and the overall `score` and `magnitude` values for the entire review, and display those to the user.

To run our sample, we'll test it on a set of (fake) movie reviews for the movie "Bladerunner."

1. Download the samples from Google Cloud Storage:

   `gsutil` is usually installed as a part of Cloud SDK. To install the latest version of Cloud SDK, please refer to Cloud SDK documentation (https://cloud.google.com/sdk/docs/).

2. Unzip those samples, which will create a "**reviews**" folder:

3. Run our sentiment analysis on one of the specified files:

The above example would indicate a review that was relatively positive (score of `0.5`), and relatively emotional (magnitude of `5.5`).

Running analysis on the other examples should produce values similar to those shown below:

Note that the magnitudes are all similar (indicating a relative equal amount of emotionally significant sentiment) except for the "neutral" case, which indicates a review with not very much emotional sentiment, either positive or negative. (For more information on sentiment scores and magnitude, and how to interpret these values, see Interpreting Sentiment Analysis Values (/natural-language/docs/basics#interpreting_sentiment_analysis_values).)

If you wish to explore sentiment analysis with more data, Stanford provides a dataset of IMDB (http://www.imdb.com) movie reviews. To retrieve these movie reviews:

1. Download the Large Movie Review dataset (http://ai.stanford.edu/%7Eamaas/data/sentiment/aclImdb_v1.tar.gz).
2. Unzip the file into your working directory. The movie reviews are divided into `pos` and `neg` directories within `train` and `test` data directories, with each text file containing one movie review.
3. Run the `sentiment_analysis.py` tool on any of the movie review text files.

Congratulations! You've performed your first inference tasks using the Google Cloud Natural Language API!