ILLINOIS

# A review of Google's Cloud Natural Language Service

nadiaw2@illinois.edu

## Introduction

Google's Cloud Platform provides various machine learning capabilities to users. The main machine learning capabilities with Google's ecosystem are TensorFlow, ML Engine, Cloud AutoML, Cloud Vision API, Cloud Speech API, Cloud Translational API, Cloud Natural API, Cloud Video Intelligence API. This review paper will focus on Google's Cloud Natural Language Platform, which consists of AutoML Natural Language, Natural Language API and Healthcare Natural Language API.

## Services

Google's Natural Language Platform provides a way for users to generate insightful text analysis by using machine learning. Three distinct services are available to the users. AutoML Natural Language, Natural Language API and Healthcare Natural Language AI.

AutoML allows users to train their own Machine Learning models to perform classification, extraction and sentiment detection using the AutoML technology. It allows users to upload their own dataset train and test custom models.

Natural Language API allows developers to easily apply natural language understanding to their applications with capabilities including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis.

Healthcare Natural Language API allows users to extract insights from unstructured medical text and feature like entity extraction allows to build custom models for embedding in healthcare apps.

This paper will be exploring the Google's Natural Language API in detail.

## Natural Language API

Google's Natural Language API can reference in applications written in these popular languages:

- C#
- Go

- Java
- Node.js
- PHP
- Python
- Ruby

The API has several features which allow a developer to perform analysis and annotation on a given text. These are as follows:

### Sentiment Analysis

This feature of the API inspects given text and identifies the emotional opinion within the text. It identifies if the writer's attitude is positive, negative, or neutral. The method `analyzeSentiment` is used to perform sentiment analysis on a document passed to this method. The API returns two values: the "score" which describes emotional leaning of the text -1 (negative) to +1 (postitive) , and 0 (neutral) and the "magnitude" which measure the strength of the emotion. Google's sentiment analysis model is trained on a very large dataset.

### Entity Analysis

This feature of the API inspects the given text for known entities. Known entities are proper nouns such as public figures, landmarks, etc. and common nouns such as restaurant, stadium etc. and returns information about those entities. If a Wikipedia link is found for such entities, that information is also included in the returned result. Entity analysis can be performed by using `analyzeEntities` method of the API. Dates, people, places, significant days etc. are identified as entities

### Entity sentiment Analysis

This feature of the API combines the functionality of the above two APIs and inspects the given text for known entities and emotion especially to determine a writer's attitude toward the entity as positive, negative, or neutral. Entity analysis is performed with the `analyzeEntitySentiment` method. The API returns entity name, type, entity sentiment score and entity sentiment magnitude.

### Syntactic Analysis

This feature of the API extracts semantic information, breaking up the given text into a series of sentences and tokens (generally, word boundaries), providing further analysis on those tokens. Syntactic Analysis is performed with the `analyzeSyntax` method. For each word, a detailed analysis is returned containing its type (noun, verb, etc.), gender, grammatical case, tense, grammatical mood, grammatical voice etc. The method also returns a dependency tree which describes the syntactic structure of each sentence.

### Content Classification

This feature of the API analyzes text content and returns a content category for the content. This model is a plug-and-play text classification model. Content

classification is performed by using the `classifyText` method. The categories are structured hierarchical. The categories are defined by Google and these are to be used as is and cannot be modified.

## Exploration of Natural Language API with Node.js

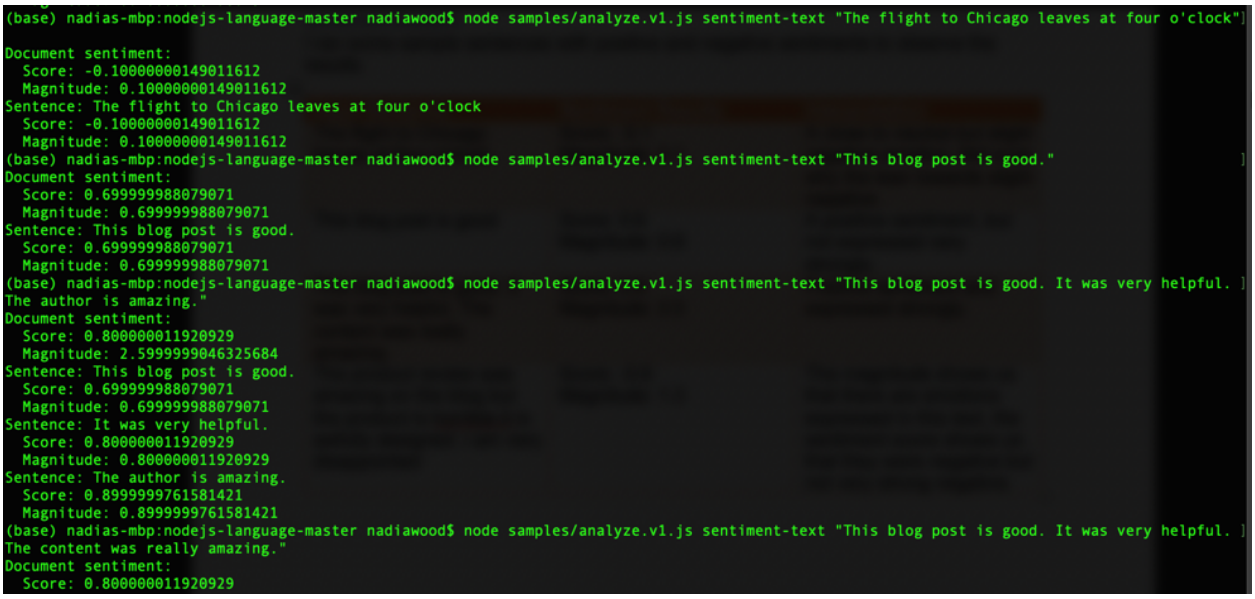
For this exploration exercise, I cloned Google's Getting Started Git Node.js repo. I passed in several sentences to test out the various analysis features. I created a Google project via Google console. The next step was to enable NLP on the project. I also created a Google service account to be able to call the NLP APIs from my local computer.

Here are results of some the analysis of the API with various text sentences as inputs.

### Sentiment Analysis

I ran some sample sentences with positive and negative sentiments to observe the results.

| Input Sentence | Sentiment Results | Interpretation |
|---|---|---|
| The flight to Chicago leaves at four o'clock | Score: -0.1<br>Magnitude: 0.1 | A close to neutral but slight negative emotion. Not sure why the lean towards slight negative. |
| This blog post is good | Score: 0.6<br>Magnitude: 0.6 | A positive sentiment, but not expressed very strongly. |
| This blog post is good. It was very helpful. The content was really amazing. | Score: 0.8<br>Magnitude: 2.5 | Positive sentiment and expressed strongly. |
| The product review was amazing on the blog but the product is horrible.It is awfully designed. I am very disappointed | Score: -0.6<br>Magnitude: 1.3 | The magnitude shows us that there are emotions expressed in this text, the sentiment score shows us that they were negative but not very strong negative. |




## Entity Analysis

For Entity Analysis, I wanted to use a test sentence with significant names, dates and places. The results include the type of entity, a Wikipedia link and a salience score for each of the entity. This score provides information about the importance or centrality of that entity to the entire document text. Scores closer to 0 are less salient, while scores closer to 1.0 are highly salient.
Input sentence “President Obama spoke to Ellen DeGeneres in Hollywood on Christmas Eve in December 2016

| **Detected Entity** |
|---|
| Obama |
| - Type: PERSON, Salience: 0.7878537178039551 |
| - Wikipedia URL: https://en.wikipedia.org/wiki/Barack_Obama |
| Hollywood |
| - Type: LOCATION, Salience: 0.09212922304868698 |
| Ellen DeGeneres |
| - Type: PERSON, Salience: 0.09106013178825378 |
| - Wikipedia URL: https://en.wikipedia.org/wiki/Ellen_DeGeneres |
| Christmas Eve |
| - Type: EVENT, Salience: 0.028956955298781395 |

| |
|---|
| - Wikipedia URL: https://en.wikipedia.org/wiki/Christmas_Eve |
| December 2016 |
| - Type: DATE, Salience: 0 |
| 2016 |
| - Type: NUMBER, Salience: 0 |

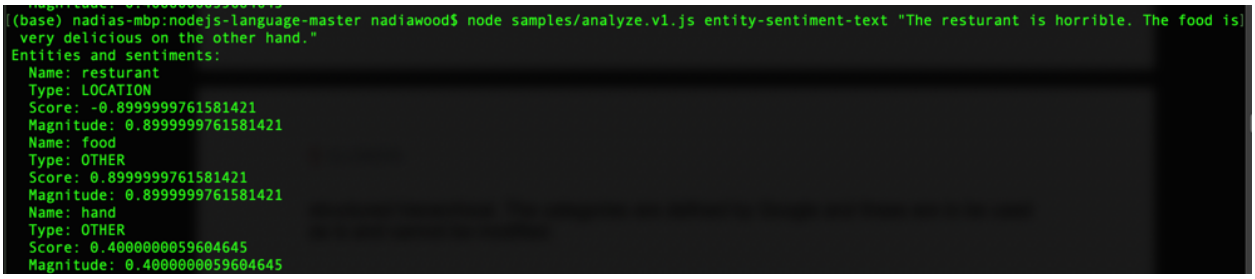
Interestingly, the year appears twice in the analysis results for some odd reason.

```
(base) nadias-mbp:nodejs-language-master nadiawood$ node samples/analyze.v1.js entities-text "President Obama spoke to Ellen DeGeneres in Ho
llywood on Christmas Eve in December 2016."
Entities:
Obama
 - Type: PERSON, Salience: 0.7878537178039551
 - Wikipedia URL: https://en.wikipedia.org/wiki/Barack_Obama
Hollywood
 - Type: LOCATION, Salience: 0.09212922304868698
Ellen DeGeneres
 - Type: PERSON, Salience: 0.09106013178825378
 - Wikipedia URL: https://en.wikipedia.org/wiki/Ellen_DeGeneres
Christmas Eve
 - Type: EVENT, Salience: 0.028956955298781395
 - Wikipedia URL: https://en.wikipedia.org/wiki/Christmas_Eve
December 2016
 - Type: DATE, Salience: 0
2016
 - Type: NUMBER, Salience: 0
(base) nadias-mbp:nodejs-language-master nadiawood$ node samples/analyze.v1.js
analyze.v1.js <command>
```

## Entity sentiment Analysis

Entity and sentiment analysis API detects entities and the emotions associated with them. In addition to recognizes entities, it also detects any dependencies within entities in the document and the associated sentiments.
An example sentence, where the first entity holds a negative sentiment but the second entity dependent on the first entity holds a positive sentiment.
“The restaurant is horrible. The food is very delicious on the other hand.”

| Entities and sentiments: |
| --- |
| Name: restaurant |
| Type: LOCATION |
| Score: -0.8999999761581421 |
| Magnitude: 0.8999999761581421 |
| Name: food |
| Type: OTHER |
| Score: 0.8999999761581421 |
| Magnitude: 0.8999999761581421 |
| Name: hand |
| Type: OTHER |
| Score: 0.4000000059604645 |
| Magnitude: 0.4000000059604645 |




## Syntactic Analysis

The syntactic API breaks each word down and gives us information about the word. Below are the results of the API, when used an input sentence “She sells seashells on the seashore”.
Notice in the results that each word has a morphology returned as JSON. The contains the type of word it is, pronoun, verb, noun, punctuation etc.
Notice the it didn’t consider “seashells” as plural.

| Tokens: |
| --- |
| PRON: She |

Morphology: { tag: 'PRON',
  aspect: 'ASPECT_UNKNOWN',
  case: 'NOMINATIVE',
  form: 'FORM_UNKNOWN',
  gender: 'FEMININE',
  mood: 'MOOD_UNKNOWN',
  number: 'SINGULAR',
  person: 'THIRD',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'TENSE_UNKNOWN',
  voice: 'VOICE_UNKNOWN' }

VERB: sells

Morphology: { tag: 'VERB',
  aspect: 'ASPECT_UNKNOWN',
  case: 'CASE_UNKNOWN',
  form: 'FORM_UNKNOWN',
  gender: 'GENDER_UNKNOWN',
  mood: 'INDICATIVE',
  number: 'SINGULAR',
  person: 'THIRD',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'PRESENT',
  voice: 'VOICE_UNKNOWN' }

NOUN: seashells

Morphology: { tag: 'NOUN',
  aspect: 'ASPECT_UNKNOWN',
  case: 'CASE_UNKNOWN',
  form: 'FORM_UNKNOWN',
  gender: 'GENDER_UNKNOWN',
  mood: 'MOOD_UNKNOWN',
  number: 'SINGULAR',
  person: 'PERSON_UNKNOWN',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'TENSE_UNKNOWN',
  voice: 'VOICE_UNKNOWN' }

ADP: on

```
Morphology: { tag: 'ADP',
  aspect: 'ASPECT_UNKNOWN',
  case: 'CASE_UNKNOWN',
  form: 'FORM_UNKNOWN',
  gender: 'GENDER_UNKNOWN',
  mood: 'MOOD_UNKNOWN',
  number: 'NUMBER_UNKNOWN',
  person: 'PERSON_UNKNOWN',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'TENSE_UNKNOWN',
  voice: 'VOICE_UNKNOWN' }
```

DET: the

```
Morphology: { tag: 'DET',
  aspect: 'ASPECT_UNKNOWN',
  case: 'CASE_UNKNOWN',
  form: 'FORM_UNKNOWN',
  gender: 'GENDER_UNKNOWN',
  mood: 'MOOD_UNKNOWN',
  number: 'NUMBER_UNKNOWN',
  person: 'PERSON_UNKNOWN',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'TENSE_UNKNOWN',
  voice: 'VOICE_UNKNOWN' }
```

NOUN: seashore

```
Morphology: { tag: 'NOUN',
  aspect: 'ASPECT_UNKNOWN',
  case: 'CASE_UNKNOWN',
  form: 'FORM_UNKNOWN',
  gender: 'GENDER_UNKNOWN',
  mood: 'MOOD_UNKNOWN',
  number: 'SINGULAR',
  person: 'PERSON_UNKNOWN',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'TENSE_UNKNOWN',
  voice: 'VOICE_UNKNOWN' }
```

PUNCT: .

```
Morphology: { tag: 'PUNCT',
  aspect: 'ASPECT_UNKNOWN',
  case: 'CASE_UNKNOWN',
  form: 'FORM_UNKNOWN',
  gender: 'GENDER_UNKNOWN',
  mood: 'MOOD_UNKNOWN',
  number: 'NUMBER_UNKNOWN',
  person: 'PERSON_UNKNOWN',
  proper: 'PROPER_UNKNOWN',
  reciprocity: 'RECIPROCITY_UNKNOWN',
  tense: 'TENSE_UNKNOWN',
  voice: 'VOICE_UNKNOWN' }
```

### Content Classification

Google's content classification API is designed to take text as an input and categorize the content in categories and sub-categories. I ran this sample text from Apple's website about the upcoming iPhone 12.

"iPhone 12, 5G transforms iPhone with accelerated wireless speeds and better performance on congested networks.5 Now you can download huge files on the go or stream high-quality HDR movies. Without. All. The. Lag. iPhone also has the most 5G bands of any smartphone so you get 5G in more places. And all that speed opens up amazing possibilities for the future of apps."

The API was 89% confident that the text belonged to Mobile Phones category and 87% confidence that it also belonged to a category of Consumer Electronics.

| **Categories:** |
| --- |
| Name: /Internet & Telecom/Mobile & Wireless/Mobile Phones, Confidence: 0.8999999761581421 |
| Name: /Computers & Electronics/Consumer Electronics, Confidence: 0.8799999952316284 |

## Cost

Google charges its users on a per-request basis for all services of the Natural Language API. This has the advantage that there are no fixed costs for any deployment servers. The disadvantage is that it can become pricey for very large datasets.

This table shows the prices (per 1,000 requests) depending on the number of monthly requests:

Monthly prices

| Feature | 0 - 5K | 5K+ - 1M | 1M+ - 5M | 5M+ - 20M |
|---|---|---|---|---|
| Entity Analysis | Free | $1.00 | $0.50 | $0.25 |
| Sentiment Analysis | Free | $1.00 | $0.50 | $0.25 |
| Syntax Analysis | Free | $0.50 | $0.25 | $0.125 |
| Entity Sentiment Analysis | Free | $2.00 | $1.00 | $0.50 |

| Feature | 0 - 30K | 30K+ - 250K | 250K+ - 5M | 5M+ |
|---|---|---|---|---|
| Content Classification | Free | $2.00 | $0.50 | $0.10 |

## Conclusion

Google's out of the box Natural Language APIs provide a very strong suite of capabilities for text analysis. The options are very quick and convenient. The models are trained on very large generic datasets. In a real-world environment, one may want to run more specific and tailored solutions that the standardized Natural Language API functions can provide. In cases like these AutoML Natural Language would be more suitable.

## References

http://cloud.google.com
https://cloud.google.com/natural-language/docs/morphology