whitehorses

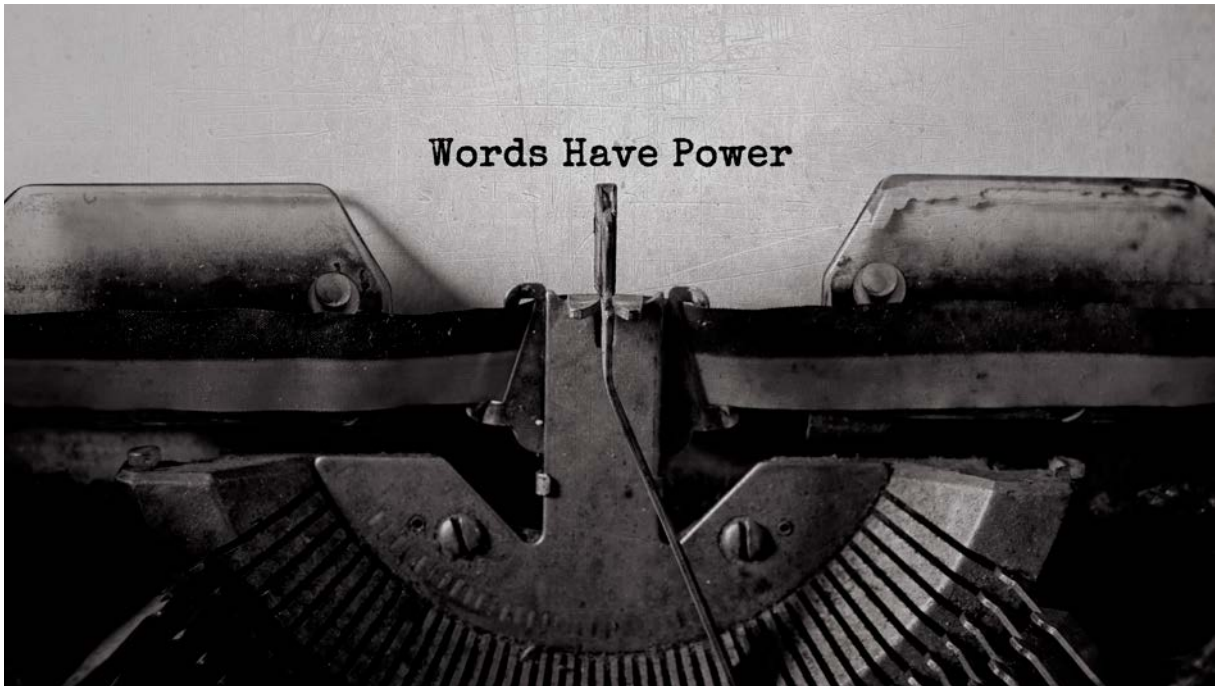
member of the human network

WHITEBOOK

Sentiment analyse en categoriseren met Google NLP

Maart 2021

Auteur:
Laurens van der Starre
INTEGRATIESPECIALIST



Inleiding

Tegenwoordig gaat alles digitaal. Als iets niet via je mobiel kan worden geregeld, is het al snel lastig. Klantcontact gaat veelal via de app, via contactformulieren op de website of gewoon via e-mail. Je kan ondersteunen op diverse manieren. De sky is the limit, chatbots, chatten met medewerkers, wizards etc etc. Soms wil je als gebruiker gewoon simpelweg een standaard webformulier invullen, of gewoon een simpele e-mail sturen. Dit kan veel werk opleveren voor de backoffice. Al de binnenkomende formulieren en mail moeten worden gelezen en verwerkt. Zou het niet handig zijn wanneer er een automatische analyse en categorisering plaatsvindt voordat het bij een medewerker belandt? Met de *Natural Language Processing* (NLP) van Google's Natural Language API kan je zeer eenvoudig binnenkomende tekst categoriseren en het sentiment analyseren. Met deze gegevens zou je al direct het binnenkomend klantcontact kunnen routeren naar de juiste afdeling of persoon, en kunnen verrijken met de juiste tags om de backoffice medewerker te kunnen ondersteunen.

Wat is Natural Language Processing?

Natural Language Processing (NLP) is een combinatie van taal-, computer- en kunstmatige intelligentie wetenschap waarmee menselijke taal wordt geanalyseerd. Het resultaat is dat de computer de taal, context en nuances begrijpt. Met deze informatie kan menselijke taal worden gecategoriseerd, en kunnen allerlei inzichten uit de taal worden geëxtraheerd. Dit heeft vele toepassingen. Hoe beter het systeem je begrijpt, hoe beter je er gebruik van kan maken. Je wilt als gebruiker dat het system jou begrijpt, in plaats van dat jij moet uitvinden hoe je het systeem zo ver gaat krijgen zodat het uiteindelijk doet wat het moet doen. Denk hier aan digitale assistenten, chat-robots voor klantenservices of zoekmachines.

Google Natural Language API

Google is natuurlijk heer en meester op het gebied van NLP. Het heeft diverse producten in haar portfolio die onder water veel gebruik maken van NLP. Denk alleen al aan de Assistant, Translate, maar ook de zoekmachine, en wat dacht je van de automatische transcriptie van bijvoorbeeld YouTube filmpjes.

In de Google Cloud is de Natural Language API beschikbaar voor gebruikers. Deze is onderdeel van Googless AI portfolio. Deze API stelt je in staat tekst te analyseren, en de metadata te retourneren.

De API kan verschillende analyses uitvoeren op tekst:

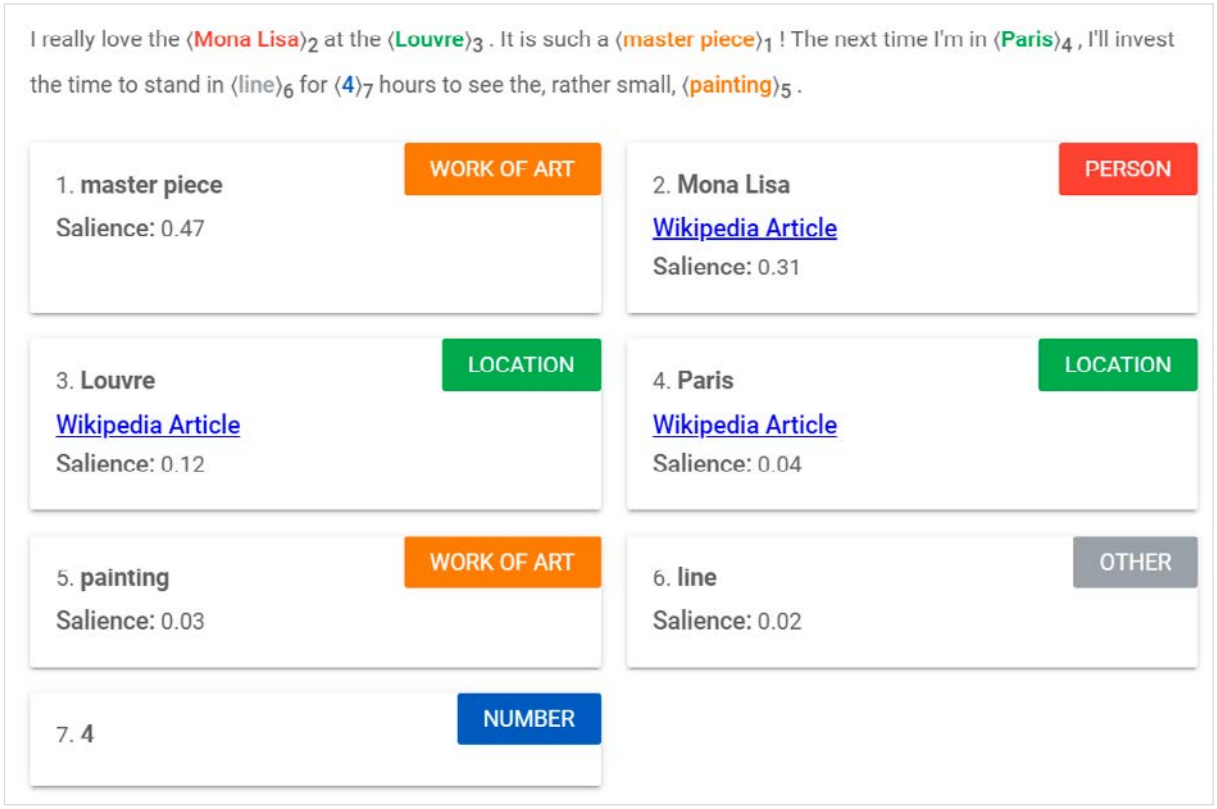
- *Entity* analyse
  - Sentiment analyse
- Van de complete tekst
- Per *Entity*
  - Categoriseren van de tekst
  - Syntax analyse (grammaticaal).

Tevens kan Googles AutoML machine learning worden ingezet om een custom classificerings AI in te zetten voor de NLP analyse.

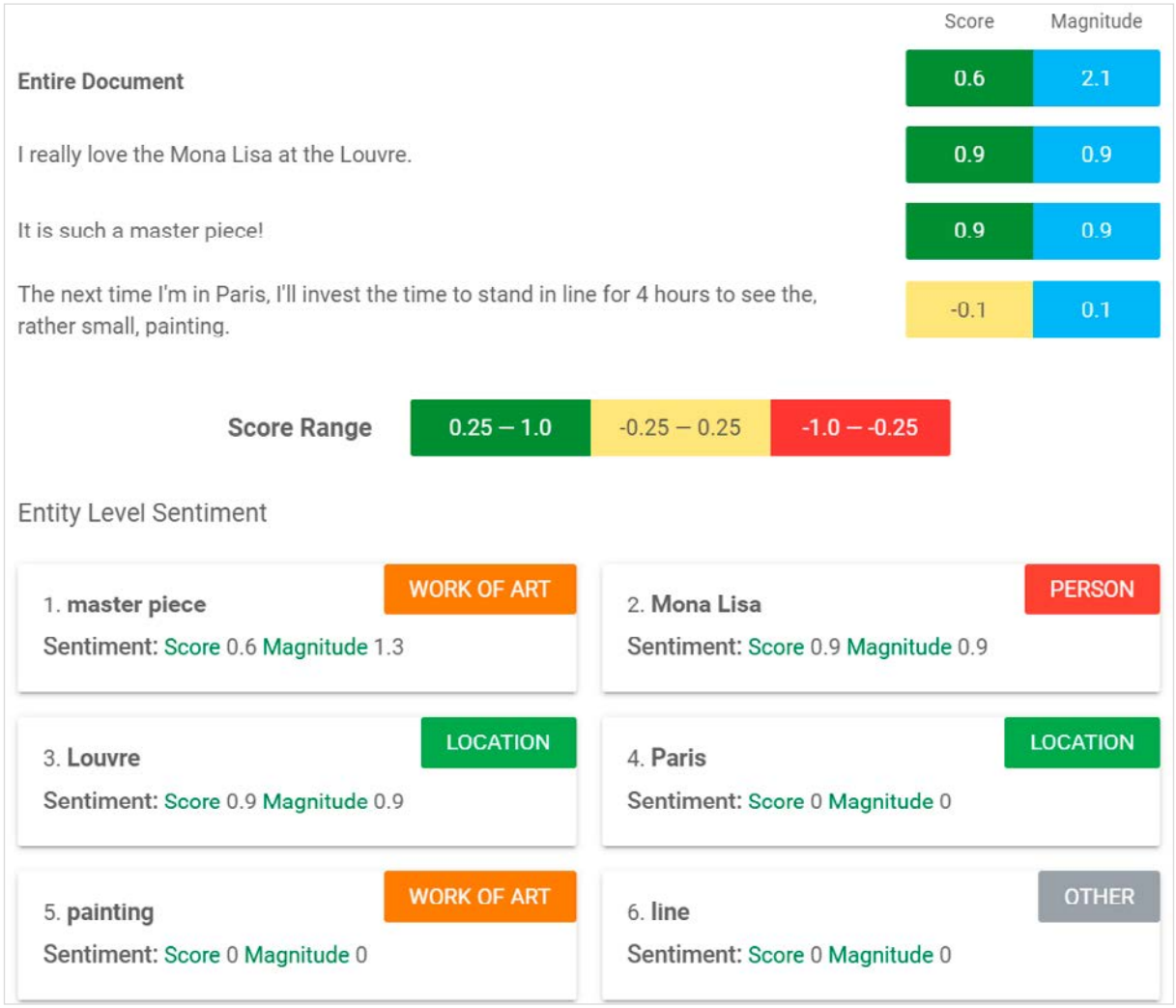
Entity analyse

Een Entity is een onderdeel van de tekst. Het is bijvoorbeeld een zelfstandig naamwoord, een locatie, een beroemd persoon, een kunstwerk e.d. De analyse van de API kan de Entities uit de tekst destilleren, en deze onderverdelen in een van de Entity Types: UNKNOWN, PERSON, LOCATION, ORGANIZATION, EVENT, WORK_OF_ART, CONSUMER_GOOD, OTHER, PHONE_NUMBER, ADDRESS, DATE, en NUMBER. Elke Entity krijgt dus automatisch een categorisering. Daarnaast wordt de Entity verder verrijkt met meta-data, zo zal bijvoorbeeld een WORK_OF_ART veelal een link naar de desbetreffende Wikipedia pagina bevatten, en is een LOCATION soms verrijkt met de link naar Google Maps.

De demo op de site van Google[1] geeft een duidelijke representatie van wat de API kan.



Sentiment analyse:



Sentiment analyse

De API kan het sentiment van de tekst bepalen. Zowel het sentiment in de gehele tekst, als ook *per Entity*.

Het sentiment wordt weergeven in twee factoren. Het is opgebouwd uit de score en de *magnitude* van die score. De score is een getal dat ligt tussen -1 en +1. Een negatieve score is een weergave van een negatief sentiment en een positieve score van een positief sentiment. De “heftigheid” van de score is uitgedrukt in *magnitude*, die van 0 tot oneindig loopt. Hoe hoger, hoe heftiger.

Bij de bepaling van het sentiment van de gehele tekst kan de score i.c.m. de magnitude dus het algemene sentiment bepalen. De API geeft zelf geen waardeoordeel anders dan deze twee getallen. Dit is aan de gebruiker om te tweaken. Bij het sentiment over de gehele tekst, is de score min of meer het gemiddelde. Dit betekent dat een score van rond de nul, met een lage magnitude een neutraal sentiment weergeeft, maar dezelfde score bij een hoge magnitude geeft een wisselend sentiment weer. Hoe meer positief (of negatief) de score is, i.c.m. een hoge magnitude geeft een goede indicatie voor een positief (of negatief) sentiment.

Naast de sentiment analyse over de gehele tekst, kan de API ook per *Entity* een sentiment bepalen. Dit bepaalt dus (wederom met een *score* en *magnitude*) het sentiment van de desbetreffende Entity in de gebruikte context. Zo kan een tekst bijvoorbeeld gaan over de prachtige *graffiti* kunst (waar de Entity graffiti hoogstwaarschijnlijk een positief sentiment zal krijgen), of kan een tekst gaan over graffiti vandalisme (waar de Entity *graffiti* hoogstwaarschijnlijk een negatief sentiment zal krijgen).

Categoriseren van de tekst

Een tekst is vaak wel globaal te categoriseren. De API kan categorieën bepalen voor een tekst. Er kunnen er meerdere worden bepaald per tekst, en elke categorie heeft ook een zekerheidsscore. Zo is de hierboven genoemde tekst over de Mona Lisa gecategoriseerd als “/Arts & Entertainment/Visual Art & Design”.

Syntax analyse

De analyse van de syntax is waar een taaldocent trots op kan zijn. Woorden kunnen bijvoorbeeld als zelfstandig naamwoord of werkwoord voorkomen, een persoon kan over zichzelf praten, over een ander, of reflecteren op zichzelf, en andere linguïstiek. Deze analyse valt buiten de scope van dit Whitebook.

Toepassing

Deze NLP is toepasbaar op diverse gebieden. In dit Whitebook wil ik het dicht bij huis houden. Neem een gemeente, deze kan uiteraard investeren in een app, of een dure backoffice applicatie hebben om een loket te bieden aan de inwoners voor vragen over allerlei gemeentezaken of bijvoorbeeld meldingen over openbare ruimten. Veel inwoners gebruiken zo'n loket waarschijnlijk maar zelden, of heel sporadisch. Veel mensen zullen niet een specifieke app hiervoor hebben geïnstalleerd (of überhaupt niet weten dat er een app is, of welke app het zou moeten zijn), of de website niet kunnen vinden of navigeren. Een e-mail sturen naar een info@ - adres weet iedereen te vinden.

De Natural Language API kan helpen binnenkomende e-mails te categoriseren om zo te routeren naar de juiste afdeling of persoon, en tevens het sentiment bepalen waardoor het wellicht eerder als klacht of aanvraag kan worden behandeld. Tevens kunnen de bepaalde Entities worden gebruikt als tags om zo gelijksoortige communicatie bij elkaar te zoeken (en te routeren). Je zou aan de hand van de tags labels aan de e-mails kunnen hangen in je mail zodat je ze bij elkaar brengt, of als gemeente toch maar eens gaan kijken op een bepaalde locatie die telkens weer naar voren komt.

Helaas is de API nog niet volledig te gebruiken in de Nederlandse taal. Alleen de sentiment analyse werkt met de Nederlandse taal. Het is een kwestie van tijd totdat die wel beschikbaar komt. Daarom is de case in dit Whitebook Engelstalig.

Allereerst dien je in het Google Cloud console een project aan te maken en de API te activeren voor het account en project. Door het aanmaken van een service account is de API snel te gebruiken in een test-omgeving, al is het aan te raden voor een productieomgeving de Google Cloud Secret Manager toe te passen.

Mijn voorbeeld is geschreven in Java Springboot. Door de pom.xml uit te breiden met:

```
<dependency>
  <groupId>com.google.cloud</groupId>
  <artifactId>google-cloud-language</artifactId>
</dependency>
```

krijg je de beschikking tot de API vanuit Java.

Ik heb vervolgens een eenvoudige RestController gemaakt die een platte tekst kan ontvangen, en deze analyseert. De uitvoer is een JSON object die de volgende informatie bevat:

- Categorie (0 .. n)
- Sentiment van de complete tekst
- Entity sentiment, onderverdeeld in:
  - Positieve
  - Negatieve
  - Neutraal of mixed.

De code drukt zelf een stempel op het sentiment. Een human readable sentiment leest namelijk eenvoudiger dan twee getallen (score en magnitude). Zeker als er een backoffice medewerker mee aan de gang moet. Het sentiment wordt als volgt bepaald:

Score	Magnitude	Omschrijving
> 0.25	>= 1.0	Very Positive
>0.25	< 1.0	Positive
0.1 ..0.25	>= 1.0	Rather Positive
0.1 ..0.25	< 1.0	A bit Positive
0.0..0.1	>= 1.0	Mixed
0.0..0.1	< 1.0	Neutral

Voor het negatieve sentiment gelden dezelfde regels, alleen is de score dan negatief.

De verdere werking van de API is verrassend eenvoudig. Door de eerder genoemde dependency in de pom.xml te laden heb je de beschikking over de `com.google.cloud.language.v1 packages`. Allereerst moet de language client worden gemaakt.

```
LanguageServiceClient language = LanguageServiceClient.
create();
```

Ik maak gebruik van platte tekst als invoer, dus ik maak een PLAIN TEKST input document:

```
Document doc = Document.newBuilder().setContent(text).
setType(Type.PLAIN_TEXT).build();
```

Neemt niet weg dat hier ook bestanden uit de Google cloud kunnen worden ingeladen, of files van disk e.d. Nu we de service client en de invoer hebben, kunnen we de verschillende analyses uitvoeren. De API bevat duidelijke classes voor de juiste analyses, zoals `ClassifyTextRequest` en `ClassifyTextResponse`, `AnalyzeSentimentResponse en AnalyzeEntitySentimentRequest en AnalyzeEntitySentimentResponse`.

Als we dit allemaal samenvoegen, en de uitvoer in een JSON-object onderbrengen ziet dat er als volgt uit:

```
private static JSONObject analyzeTheInput(String text)
throws IOException {

        JSONObject responseDoc = new JSONObject();


        try (LanguageServiceClient language =
LanguageServiceClient.create()) {
            // set content to the text string
            Document doc = Document.newBuilder().
setContent(text).setType(Type.PLAIN_TEXT).build();
```

```
            ClassifyTextRequest request =
ClassifyTextRequest.newBuilder().setDocument(doc).build();
            // detect categories in the given text
            ClassifyTextResponse classifyResponse =
language.classifyText(request);

            JSONArray catArray = new JSONArray();

            for (ClassificationCategory category :
classifyResponse.getCategoriesList()) {

                catArray.put(category.getName());

            }

            responseDoc.put("Categories", catArray);

            AnalyzeSentimentResponse
analyzeSentimentResponse = language.analyzeSentiment(doc);
            Sentiment sentiment = analyzeSentimentResponse.
getDocumentSentiment();
            if (sentiment == null) {
                responseDoc.put("Overall sentiment", "No
sentiment found");
            } else {

                responseDoc.put("Overall sentiment",
getSentimentAsString(sentiment.getScore(), sentiment.
getMagnitude()));
            }

            AnalyzeEntitySentimentRequest entityRequest =
AnalyzeEntitySentimentRequest.newBuilder().setDocument(doc)
                    .setEncodingType(EncodingType.UTF16).
build();
            // detect entity sentiments in the given string
            AnalyzeEntitySentimentResponse entityResponse =
language.analyzeEntitySentiment(entityRequest);
```

```
            JSONArray positiveEntityArray = new JSONArray();
            JSONArray negativeEntityArray = new JSONArray();
            JSONArray otherEntityArray = new JSONArray();

            for (Entity entity : entityResponse.
getEntitiesList()) {

                JSONObject entityObj = new JSONObject();
                String sentimentString =
getSentimentAsString(entity.getSentiment().getScore(),
                    entity.getSentiment().getMagnitude());

                    entityObj.put("Type", entity.getType().
toString());
                    entityObj.put("Entity", entity.
getName());
                    entityObj.put("Sentiment",
sentimentString);

                    if (sentimentString.
contains("Positive")) {
                        positiveEntityArray.put(entityObj);
                    } else if (sentimentString.
contains("Negative")) {
                        negativeEntityArray.put(entityObj);
                    } else {
                        otherEntityArray.put(entityObj);
                    }
            }
            responseDoc.put("Other Entity Sentiment",
otherEntityArray);
            responseDoc.put("Negative Entity Sentiment",
negativeEntityArray);
            responseDoc.put("Positive Entity Sentiment",
positiveEntityArray);
        }
        return responseDoc;
    }
```

Als we dit in de context van de casus testen met de volgende ingezonden tekst:

Hi,

I'm wondering something about the upcoming food festival in the main park this July. Will there be shuttle busses going from the central parking to the park? What will the route be? Not through my street right? 'd rather not have those terrible busses through my quiet street! I hate those stinking old diesel busses.

Analyse:


```
{
    "Overall sentiment": "Rather Negative",
    "Categories": [
        "/Arts & Entertainment"
    ],
    "Other Entity Sentiment": [
        {
            "Entity": "park",
            "Type": "LOCATION",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "something",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "food festival",
            "Type": "EVENT",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "shuttle busses",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        },
```

```
        {
            "Entity": "parking",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "route",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "busses",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "street",
            "Type": "LOCATION",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "d",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        }
    ],
    "Negative Entity Sentiment": [
        {
            "Entity": "street",
            "Type": "LOCATION",
            "Sentiment": "A bit Negative"
        },
        {
            "Entity": "diesel busses",
            "Type": "OTHER",
            "Sentiment": "Negative"
        }
    ],
```

```
    "Positive Entity Sentiment": []
}
```

Het is duidelijk te zien dat er weinig echt positief is aan deze tekst, met een redelijk neutraal begin, en een negatief einde. De automatische categorisering zou deze tekst routeren naar bijvoorbeeld de event planner, want zowel de categorie als ook de locatie en event zelf zijn uit de tekst gehaald.

Ander voorbeeld:

I would like to have a building permit for an extra large shed in my garden. Building this, a tree has to be removed. But when it is finished, I'll have the garden of my dreams!

Dit geeft:

```
{
    "Overall sentiment": "Rather Positive",
    "Categories": [
        "/Business & Industrial",
        "/Home & Garden"
    ],
    "Other Entity Sentiment": [
        {
            "Entity": "building permit",
            "Type": "OTHER",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "garden",
            "Type": "LOCATION",
            "Sentiment": "Neutral"
        },
        {
            "Entity": "shed",
            "Type": "PERSON",
            "Sentiment": "Neutral"
```

```
        }
    ],
    "Negative Entity Sentiment": [
        {
            "Entity": "tree",
            "Type": "OTHER",
            "Sentiment": "A bit Negative"
        }
    ],
    "Positive Entity Sentiment": [
        {
            "Entity": "garden",
            "Type": "LOCATION",
            "Sentiment": "Positive"
        },
        {
            "Entity": "dreams",
            "Type": "OTHER",
            "Sentiment": "Positive"
        }
    ]
}
```


Het is duidelijk te zien dat de schrijver het niet zo op heeft met de boom (want die staat in de weg). De afdeling voor de ruimtelijke ordening zou deze kunnen behandelen, en a.d.v de categorie en de Enities is het direct duidelijk dat het hier om een (kap)vergunning gaat zonder dat iemand dit eerst handmatig hoeft te bepalen. Deze kan op de stapel bij de persoon die hier over gaat.

Conclusie

Met een paar regels code kan tekst worden geanalyseerd door de Google Natural Language API. Deze analyse kan van enorme toegevoegde waarde zijn in de verwerking van de binnengekomen tekst in bijvoorbeeld een backoffice casus. Sentiment analyse, de categorisering en de Entity analyse kan bijvoorbeeld worden gebruikt in routering naar de juiste afdeling, of taggen van meta-data aan de tekst. De meta-data is krachtig, je kan informatie groeperen, zoekbaar maken of trendanalyses toepassen t.b.v marketing of het identificeren van een (onderliggend) probleem. De Natural Languae API komt nog beter tot zijn recht als Googles AutoML wordt toegepast om je met je eigen getrainde naural network je eigen categorisering te verkrijgen. Daarnaast kan samen met Googles Document AI een volledig AI gestuurde document en invoice verwerking plaatsvinden.
Daarnaast is sentiment analyse op de tekst en Entities een zeer krachtige tool in social media. We hebben net de verkiezingen achter de rug, en in zo'n verkiezingstijd is het uitermate krachtig om een Twitter-feed over een bepaald onderwerp of lijsttrekker door de Natural Language API te halen. Door het sentiment te weten, en het sentiment over bepaalde onderwerpen (de Entities) kan je een bepaalde strategie aannemen in de campagne. Dit is niet alleen erg krachtig, maar hier liggen ook de gevaren (kijken naar de Amerikaanse verkiezingen …).

Referenties

1. Try the API!
https://cloud.google.com/natural-language