# AWS Comprehend is able to detect the following:

## Dominant Language

Amazon Comprehend defines language as, the primary language used in the text.

Given a text Amazon Comprehend uses the DetectDominantLanguage operation will respond with the language code and the associated score. For example, the text “It’s a sunny day in Spokane” will return

{

"Languages": [

{

"LanguageCode": "en",

"Score": 0.9793661236763

}

]

}

To use this operation in Java the code is:

DetectDominantLanguageRequest detectDominantLanguageRequest = new DetectDominantLanguageRequest().withText(text);

DetectDominantLanguageResult detectDominantLanguageResult = comprehendClient.detectDominantLanguage(detectDominantLanguageRequest);

## Named Entities

Amazon Comprehend defines entities as the persons, places, dates, ect.

Given a text “It's a Monday and the sky is clear in Spokane. I think I will take my bike for a ride around the park and hope that I do not fall and scrape my knee. I know when Tyree rode his bike around the same path he fell and broke his pinkie toe.” Amazon Comprehend uses the DetectEntities operation to extract the entities from the text. To use this operation the language code must be specified, for our project we will only use the language code of “en”. DetectEntities operation will return

{

"Entities": [

{

"Score": 0.9798023104667664,

"Type": "DATE",

"Text": "Monday",

"BeginOffset": 7,

"EndOffset": 13

},

{

"Score": 0.9862631559371948,

"Type": "LOCATION",

"Text": "Spokane",

"BeginOffset": 38,

"EndOffset": 45

},

{

"Score": 0.9818150401115417,

"Type": "PERSON",

"Text": "Tyree",

"BeginOffset": 162,

"EndOffset": 167

}

]

}

In the text three entities are found, a person, place and a date. Each entity has an associated score as well as the beginning and ending index of the entity in the text.

To use in Java, the code is:

DetectEntitiesRequest detectEntitiesRequest = new DetectEntitiesRequest().withText(text).withLanguageCode("en");

DetectEntitiesResult detectEntitiesResult = comprehendClient.detectEntities(detectEntitiesRequest);

## Key Phrases

Amazon Comprehend defines the key phrases as the nouns in the text.

Given a text “Rose is an indoor cat, she likes to look out the window and watch the birds every day.” Amazon Comprehend uses the operation DetectKeyPhrases to extract the key phrases from the text and the language code must be specified. In the given text there are five key phrases,

{

"KeyPhrases": [

{

"Score": 0.9941727519035339,

"Text": "Rose",

"BeginOffset": 0,

"EndOffset": 4

},

{

"Score": 0.998713493347168,

"Text": "an indoor cat",

"BeginOffset": 8,

"EndOffset": 21

},

{

"Score": 0.9989926815032959,

"Text": "the window",

"BeginOffset": 45,

"EndOffset": 55

},

{

"Score": 0.9982677698135376,

"Text": "the birds",

"BeginOffset": 66,

"EndOffset": 75

},

{

"Score": 0.9991843700408936,

"Text": "every day",

"BeginOffset": 76,

"EndOffset": 85

}

]

}

Similarly to the entities, each key phrase has an associated score as well as the beginning and ending index of the key phrase in the text.

To use in Java, the code is:

DetectKeyPhrasesRequest detectKeyPhrasesRequest = new DetectKeyPhrasesRequest().withText(text).withLanguageCode("en");

DetectKeyPhrasesResult detectKeyPhrasesResult = comprehendClient.detectKeyPhrases(detectKeyPhrasesRequest);

## Sentiment

Amazon Comprehend defines sentiment as the emotional sentiment, whether positive, negative, neutral, or mixed, conveyed by the text. The text will only have one sentiment but the sentiment score will break down each positive, negative, neutral, or mixed with an associated confidence interval on a 1.0 scale.

Using the same text as above and using the DetectSentiment operation will determine the overall emotion of the text

{

"Sentiment": {

"Sentiment": "NEUTRAL",

"SentimentScore": {

"Positive": 0.36940476298332214,

"Negative": 0.008159413933753967,

"Neutral": 0.6204316020011902,

"Mixed": 0.002004295587539673

}

}

}

To use in Java, the code is:

DetectSentimentRequest detectSentimentRequest = new DetectSentimentRequest().withText(text).withLanguageCode("en");

DetectSentimentResult detectSentimentResult = comprehendClient.detectSentiment(detectSentimentRequest);

## Syntax

Amazon Comprehend will parse the given text and determine each individual words parts of speech. We must specify the language when determining the parts of speech, and since we are using the English language there are 8 parts of speech, noun, pronoun, adjective, verb, adverb, preposition, conjunction and interjection.

The operation used is, DetectSyntax and given the same text as above will return

{

"TokenId": 7,

"Text": "she",

"BeginOffset": 23,

"EndOffset": 26,

"PartOfSpeech": {

"Tag": "PRON",

"Score": 0.9999942779541016

}

}

Where the word “she” is tagged as a pronoun.

The Java code,

DetectSyntaxRequest detectSyntaxRequest = new DetectSyntaxRequest()

.withText(text)

.withLanguageCode("en");

DetectSyntaxResult detectSyntaxResult = comprehendClient.detectSyntax(detectSyntaxRequest);

Will return, the token ID, the beginning and ending index of the word in the text, the part of speech and associated score.

# Product Reviews

For our project we will be using products reviews for Amazon AAA batteries. The following program will extract all Insights from a single review and display them to the standard output.

package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentialsProvider;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;

import com.amazonaws.regions.Regions;

import com.amazonaws.services.comprehend.AmazonComprehend;

import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;

import com.amazonaws.services.comprehend.model.DetectDominantLanguageRequest;

import com.amazonaws.services.comprehend.model.DetectDominantLanguageResult;

import com.amazonaws.services.comprehend.model.DetectEntitiesRequest;

import com.amazonaws.services.comprehend.model.DetectEntitiesResult;

import com.amazonaws.services.comprehend.model.DetectKeyPhrasesRequest;

import com.amazonaws.services.comprehend.model.DetectKeyPhrasesResult;

import com.amazonaws.services.comprehend.model.DetectSentimentRequest;

import com.amazonaws.services.comprehend.model.DetectSentimentResult;

import com.amazonaws.services.comprehend.model.DetectSyntaxRequest;

import com.amazonaws.services.comprehend.model.DetectSyntaxResult;

import com.amazonaws.services.comprehend.model.StartKeyPhrasesDetectionJobRequest;

import com.amazonaws.services.s3.AmazonS3;

import com.amazonaws.services.s3.AmazonS3Client;

public class App

{

public static void main(String... args )

{

String text = "A battery is a battery these work just as good if not better than the name brands like Duracell or Enginzer";

AmazonComprehend comprehendClient =

AmazonComprehendClientBuilder.standard()

.withRegion(Regions.US\_EAST\_2)

.build();

//detect language

System.out.println("Calling DetectDominantLanguage:");

DetectDominantLanguageRequest detectDominantLanguageRequest = new DetectDominantLanguageRequest().withText(text);

DetectDominantLanguageResult detectDominantLanguageResult = comprehendClient.detectDominantLanguage(detectDominantLanguageRequest);

detectDominantLanguageResult.getLanguages().forEach(System.out::println);

//detect entities

System.out.println("\nCalling DetectEntities:");

DetectEntitiesRequest detectEntitiesRequest = new DetectEntitiesRequest().withText(text)

.withLanguageCode("en");

DetectEntitiesResult detectEntitiesResult = comprehendClient.detectEntities(detectEntitiesRequest);

detectEntitiesResult.getEntities().forEach(System.out::println);

System.out.println("\nCalling DetectKeyPhrases:");

DetectKeyPhrasesRequest detectKeyPhrasesRequest = new DetectKeyPhrasesRequest().withText(text)

.withLanguageCode("en");

DetectKeyPhrasesResult detectKeyPhrasesResult = comprehendClient.detectKeyPhrases(detectKeyPhrasesRequest);

detectKeyPhrasesResult.getKeyPhrases().forEach(System.out::println);

System.out.println("\nCalling DetectSentiment:");

DetectSentimentRequest detectSentimentRequest = new DetectSentimentRequest().withText(text)

.withLanguageCode("en");

DetectSentimentResult detectSentimentResult = comprehendClient.detectSentiment(detectSentimentRequest);

System.out.println(detectSentimentResult);

System.out.println("\nCalling DetectSyntax:");

DetectSyntaxRequest detectSyntaxRequest = new DetectSyntaxRequest()

.withText(text)

.withLanguageCode("en");

DetectSyntaxResult detectSyntaxResult = comprehendClient.detectSyntax(detectSyntaxRequest);

detectSyntaxResult.getSyntaxTokens().forEach(System.out::println);

}//end main

}//end class

/\* OUTPUT--------------------------------------------------------------------------------------------------------------------

Calling DetectDominantLanguage:

{LanguageCode: en,Score: 0.9832828}

Calling DetectEntities:

{Score: 0.97644955,Type: ORGANIZATION,Text: Duracell,BeginOffset: 87,EndOffset: 95}

{Score: 0.9727436,Type: ORGANIZATION,Text: Enginzer,BeginOffset: 99,EndOffset: 107}

Calling DetectKeyPhrases:

{Score: 0.9994282,Text: A battery,BeginOffset: 0,EndOffset: 9}

{Score: 0.99089104,Text: a battery,BeginOffset: 13,EndOffset: 22}

{Score: 0.7892732,Text: these work,BeginOffset: 23,EndOffset: 33}

{Score: 0.9894793,Text: the name brands,BeginOffset: 66,EndOffset: 81}

Calling DetectSentiment:

{Sentiment: POSITIVE,SentimentScore: {Positive: 0.5611031,Negative: 0.050375346,Neutral: 0.31922027,Mixed: 0.06930124}}

Calling DetectSyntax:

{TokenId: 1,Text: A,BeginOffset: 0,EndOffset: 1,PartOfSpeech: {Tag: DET,Score: 0.9995931}}

{TokenId: 2,Text: battery,BeginOffset: 2,EndOffset: 9,PartOfSpeech: {Tag: NOUN,Score: 0.99987245}}

{TokenId: 3,Text: is,BeginOffset: 10,EndOffset: 12,PartOfSpeech: {Tag: VERB,Score: 0.996846}}

{TokenId: 4,Text: a,BeginOffset: 13,EndOffset: 14,PartOfSpeech: {Tag: DET,Score: 0.99998724}}

{TokenId: 5,Text: battery,BeginOffset: 15,EndOffset: 22,PartOfSpeech: {Tag: NOUN,Score: 0.9965569}}

{TokenId: 6,Text: these,BeginOffset: 23,EndOffset: 28,PartOfSpeech: {Tag: DET,Score: 0.9681557}}

{TokenId: 7,Text: work,BeginOffset: 29,EndOffset: 33,PartOfSpeech: {Tag: NOUN,Score: 0.97155625}}

{TokenId: 8,Text: just,BeginOffset: 34,EndOffset: 38,PartOfSpeech: {Tag: ADV,Score: 0.9990339}}

{TokenId: 9,Text: as,BeginOffset: 39,EndOffset: 41,PartOfSpeech: {Tag: ADV,Score: 0.7961517}}

{TokenId: 10,Text: good,BeginOffset: 42,EndOffset: 46,PartOfSpeech: {Tag: ADJ,Score: 0.94888467}}

{TokenId: 11,Text: if,BeginOffset: 47,EndOffset: 49,PartOfSpeech: {Tag: SCONJ,Score: 0.80172473}}

{TokenId: 12,Text: not,BeginOffset: 50,EndOffset: 53,PartOfSpeech: {Tag: ADV,Score: 0.6733074}}

{TokenId: 13,Text: better,BeginOffset: 54,EndOffset: 60,PartOfSpeech: {Tag: ADJ,Score: 0.6780679}}

{TokenId: 14,Text: than,BeginOffset: 61,EndOffset: 65,PartOfSpeech: {Tag: ADP,Score: 0.95338273}}

{TokenId: 15,Text: the,BeginOffset: 66,EndOffset: 69,PartOfSpeech: {Tag: DET,Score: 0.99999106}}

{TokenId: 16,Text: name,BeginOffset: 70,EndOffset: 74,PartOfSpeech: {Tag: NOUN,Score: 0.99773836}}

{TokenId: 17,Text: brands,BeginOffset: 75,EndOffset: 81,PartOfSpeech: {Tag: NOUN,Score: 0.9976961}}

{TokenId: 18,Text: like,BeginOffset: 82,EndOffset: 86,PartOfSpeech: {Tag: ADP,Score: 0.9871956}}

{TokenId: 19,Text: Duracell,BeginOffset: 87,EndOffset: 95,PartOfSpeech: {Tag: PROPN,Score: 0.98575807}}

{TokenId: 20,Text: or,BeginOffset: 96,EndOffset: 98,PartOfSpeech: {Tag: CONJ,Score: 0.99999785}}

{TokenId: 21,Text: Enginzer,BeginOffset: 99,EndOffset: 107,PartOfSpeech: {Tag: PROPN,Score: 0.9848514}}

\*/