# Topic Classification Using AWS Comprehend

***This project report is submitted to***

***Aptus Data Labs, Bangalore***

***in partial fulfillment of the requirements for the award of the certification of***

#### Post Graduate Diploma in

**Data Science (2019-2020)**

##### *Submitted by*

##### Mr. Uday Chandu Bachina (19FTDDS020)

##### Mr. Mohd. Afzal Ramiz (19FTDDS018)

##### Mr. Sourav Sambit Panda (19FTDDS003) &

##### Mr. Karan Mitra (19FTDDS009)

***Under the Esteemed Supervision of***

##### Mr. Ravindra Swamy

##### Mr. Pramit Kumar Patra

##### Mr. Akshay Barik &

##### Mr. Siddharth

***Duration: From 20th April 2020 To 27th May 2020***

****

**DECLARATION**

I hereby certify that: -

1. The work contained in the project is original and has been done by myself under the supervision of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have conformed to the norms and guidelines given to us by the Project Review Committee of our department.
4. Whenever I have used materials (data, theoretical analysis and text) from other sources, I have given due credit to them by citing them in the text of the project and giving their details in the references.

Date: 03rd June 2020

Place: Bhubaneswar

Mr. UdayChandu Bachina (19FTDDS020)

Mr. Mohd. Afzal Ramiz (19FTDDS018)

Mr. Sourav Sambit Panda (19FTDDS003) &

Mr. Karan Mitra (19FTDDS009)

## ACKNOWLEDGEMENT

I would take this opportunity to express my gratitude to all who have made this project possible. I would like to express my deepest appreciation to our team of Aptus Data Labs who provided me the possibility to complete this project. I would like to give a special gratitude to my project supervisor, **Mr. Akshay Barik & Mr. Siddharth,** whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report. I take this opportunity to express my sincere thanks to our CTO of Aptus Data Labs Mr. Ravindra Swamy Sir for providing the necessary facilities. Furthermore, I would also like to acknowledge with much appreciation the crucial role of our friends and parents for encouraging us and helping us complete this project.

Mr. UdayChandu Bachina (19FTDDS020)

Mr. Mohd. Afzal Ramiz (19FTDDS018)

Mr. Sourav Sambit Panda (19FTDDS003) &

Mr. Karan Mitra (19FTDDS009)

## ABSTRACT

Topic classification is the process of assigning tags or categories to documents of text according to its content. It’s one of the fundamental tasks in [Natural Language Processing](https://monkeylearn.com/blog/definitive-guide-natural-language-processing/) (NLP) with broad applications such as topic labeling, spam detection, and intent detection.

Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, support tickets, survey responses, and so on. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature.

Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes.

**TABLE OF CONTENTS**

[TITLE i](#_Toc4769034)

[DECLARATION ii](#_Toc4769036)

[ACKNOWLEDGEMENT iii](#_Toc4769035)

[ABSTRACT](#_Toc4769034) iv

[TABLE OF CONTENTS](#_Toc4769036) v-vi

[LIST OF FIGURES](#_Toc4769037) vi

[CHAPTER 1 1](#_Toc4769038)

[INTRODUCTION 1](#_Toc4769039)

[1.1 Problem Statement 1](#_Toc4769040)

[1.2 Solution 1](#_Toc4769041)

[CHAPTER 2 2](#_Toc4769042)

[ARCHITECTURE 2](#_Toc4769043)

[2.1 Software Requirements 2](#_Toc4769044)

[2.2 Package Requirements 3](#_Toc4769045)

[2.3 Architecture 4](#_Toc4769046)

[CHAPTER 3 5](#_Toc4769047)

[WORKING FLOW(Step By Step) 5-6](#_Toc4769048)

[CHAPTER 4 7](#_Toc4769047)

[KNOWN ISSUES/METHODOLOGY 7](#_Toc4769048)-9

[CHAPTER 5 10](#_Toc4769060)

[MODEL EVALUATION 10-11](#_Toc4769061)

[CHAPTER 6 12](#_Toc4769047)

[SCOPE OF ENHANCEMENT 12](#_Toc4769048)

[CHAPTER 7 13](#_Toc4769047)

[RESOURCES LINKS 1](#_Toc4769048)3

[7.1 Github Links 13](#_Toc4769044)

[CHAPTER 8 14](#_Toc4769083)

[CONCLUSION 14](#_Toc4769084)

[REFERENCES 15](#_Toc4769085)

**LIST OF FIGURES**

**Figure No. Description**

Fig 1 The flow of data for retrieving the classified topics for a given set of documents.

Fig 2 Applying ‘create\_document\_classifier’ method to train our dataset.

Fig 3 Applying ‘create\_endpoint’ method to evaluate our training.

Fig 4 Applying ‘start\_document\_classification\_job’ method to **f**ind out the

predicted classes.

**Chapter 1**

**Introduction**

**1.1 Problem Statement:**

Here we have a dataset that resembles a real-world news content. The news comprises various fields from business to technology and our dataset contains 2225 documents from the [BBC](http://news.bbc.co.uk/) news website corresponding to stories in five topical areas from 2004-2005. It consists of five different Class Labels (business, entertainment, politics, sport, tech).

The objective of this project is to classify and assign tags or categories to the text documents of each topic namely business, entertainment, politics, sport, tech, according to its content, which is also known as topic labeling.

**1.2 Solution:**

Our target is to classify the documents into those different predefined topics. The output of each document should be a map of a topic name. For example, {“Doc1”: Business, “Doc2”: Entertainment} and so on by using the **Amazon Comprehend** as it uses natural language processing (NLP) to extract insights about the content of documents without the need of any special preprocessing.

The overall approach was to prepare the training dataset from the given dataset and here we have taken 300 text files from the given dataset as train dataset and have trained the model using AWS custom comprehend and then have used the trained model to predict the topic of the new document using AWS comprehend APIs and here 25 randomly text files have been taken as test dataset for that task.

**Chapter 2**

**Architecture**

**2.1 Software Requirements:**

The softwares required for building the AWS comprehend:

1. **Python Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

1. **AWS CLI**

The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

We have configured the AWS CLI with Access\_Key\_Id ,Secret\_Key,Region and the format of output by the help of a command

> AWS configure

**>aws configure**

**AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XJUE]:**

**AWS Secret Access Key [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FtKs]:**

**Default region name [us-west-2]:**

**Default output format [json]:**

Above details are passed correctly to use the services of **AWS (**Amazon Web Services**)**

**2.2 Package Requirements:**

Packages used:

1. **Pandas**

We have used the pandas package for the purpose of structuring the documents as a DataFrame. So that the input for AWS comprehend is made easy.

1. **boto3**

In order to use of boto3 the developer needs a account in AWS. So, by using the credentials of AWS account the services are easily accessed.

The methods we used in boto3 are:

1. boto3.client('comprehend')

to access the services from AWS, we used the client () method.

1. client.create\_document\_classifier()

for training the model, document classifier is helpful

1. client.create\_endpoint()

in order to show the customer about the output for a understanding purpose we have endpoints () method. This is a synchronous purpose.

1. client.describe\_endpoint()

to know the details of endpoint we have describe\_endpoint() method.

1. client.list\_endpoints()

in case there are multiple endpoints for a single trained model,list\_endpoints() is used to know.

1. client.classify\_document()

this uses the endpoint arn to predict the output for a given set of documents.

1. client.delete\_endpoint()

for deleting purpose of an endpoint this can be used.

1. client.start\_document\_classification\_job()

this is for creating a job (set of documents are passed as a batch) for predictions.

Asynchronous kind of classification.

1. client.describe\_document\_classification\_job()

for describing the details of a created job describe\_document\_classification\_job() is used.

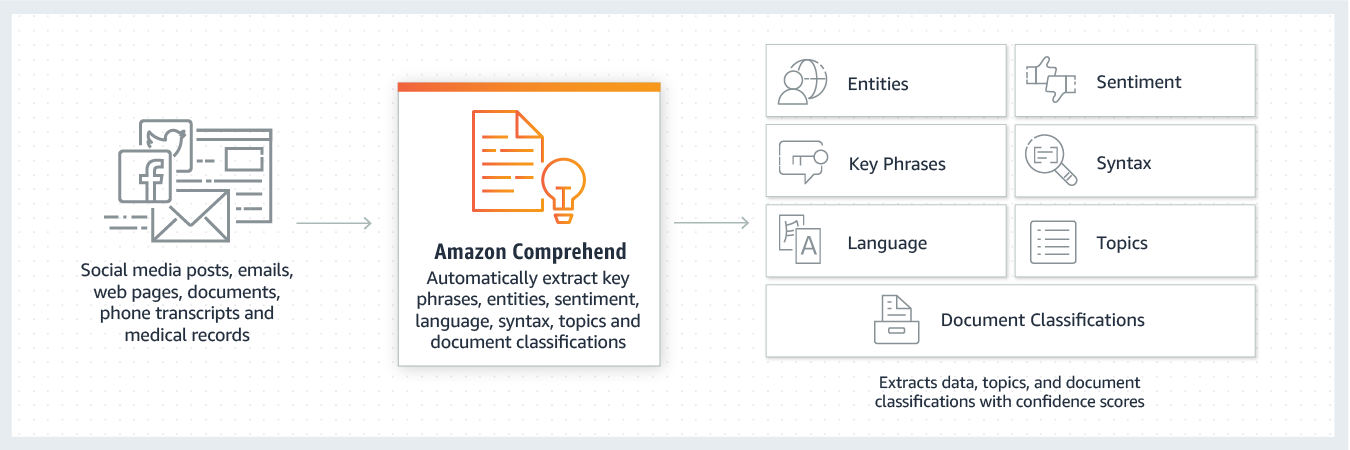
1. client.list\_document\_classification\_jobs()

If there are multiple jobs are present in the list, list\_document\_classification\_jobs() delivers the total list.

**2.3 Architecture:**

The following figure shows the flow of data for retrieving the classified topics for a given set of documents.

The data from different platforms can be collected and are placed in the S3 storage in AWS and are passed to the AWS Comprehend for different kind of outputs as shown.



*Fig-1*

## Custom Document Classification:

The Custom Classification API enables you to easily build custom text classification models using your business-specific labels without learning ML. For example, your customer support organization can use Custom Classification to automatically categorize inbound requests by problem type based on how the customer has described the issue. Creating a custom model is simple. You provide examples of text for each of the labels you want to use, and Comprehend trains on those to create your custom model. No machine learning experience required, you can build your custom model without using a single line of code. An SDK is available for you to integrate your customer classifier into your current applications. With your custom model, it is easy to moderate website comments, triage customer feedback, and organize workgroup documents. Refer to [this documentation page](https://docs.aws.amazon.com/comprehend/latest/dg/how-document-classification.html) for more details.

**Chapter 3**

In our project the solution needs to be built using AWS Comprehend which is a machine learning service that makes the task of Text analysis.

With AWS Comprehend, a user can provide documents in an S3 bucket and use the comprehend APIs to detect entities, sentiment, language, etc. in text documents. Amazon comprehend also supports adding custom topics or entities to classify documents for specific business needs.

AWS Comprehend can be used in multiple ways.

* We can directly use the AWS Comprehend console to upload documents and perform tasks like entity extraction, key phrase, language etc.
* We can use AWS Comprehend APIs with the SDK provided by AWS for different programming languages.

**Working Flow (Step by Step)**

**Step1: Data Collection**

Here, the data collected is in a raw text format.

In the given use case, there are 2225 documents from different areas (sports, business, tech, politics and entertainment.) collectively.

**Step2: Structuring the data content and getting the train and test dataset**

We made some transformation (labelling) on data in order to get some structure to it. After the data is in structured format, we have done some preprocessing on the text to divide it to training dataset and testing dataset.

As the raw data cannot be given as input to comprehend, as a input provider, we have converted the text documents into a DataFrame and removed the suspicious content like ‘\n’ in every document.

From the whole 2225 documents, the trained data and test data are separated by the ratio of 300 files from each area. Therefore, the next procedure can be processed.

**Step3: Training the data**

We have uploaded our training dataset to S3 bucket. Then we have imported boto3 client.

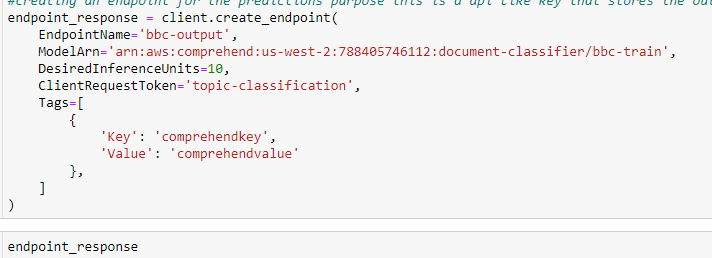
We have used ‘create\_document\_classifier’ method to train our dataset. To create a classifier, we provide a set of training documents that labeled with the categories that we want to use. After the classifier is trained we can use it to categorize a set of labeled documents into the categories.



*Fig-2*

**Step4: Evaluating trained model**

We have used ‘create\_endpoint’ method to evaluate our training. After training the model, we provide the output of ‘create\_document\_classifier’ arn as an input to ‘create\_endpoint’ method to evaluate it. In the production phase this method is used only for predicting the api that has generated by the developer to predict for a single text document. For the overall prediction of test documents we go for another method called start\_document\_classifier\_job().



*Fig-3*

**Step5: Finding out the predicted classes**

We have used ‘start\_document\_classification\_job’ method for this. After evaluating the model, we provide the output of ‘create\_document\_classifier()’s’ which is DocumentClassifierArn as an input to ‘start\_document\_classification\_job’ method and specifies the location to store the output to S3 bucket.

Keeping the AWS comprehend charges in consideration, we have taken the first 5 documents of each area apart from the training set. Thus, the 25 documents are validated with the help of start\_document\_classifier\_job() method.

Those 25 documents are passed as a batch inorder o predict the documents.



*Fig-4*

**Chapter 4**

**What Is Amazon Comprehend?**

Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents. Amazon Comprehend processes any text file in UTF-8 format. It develops insights by recognizing the entities, key phrases, language, sentiments, and other common elements in a document. Use Amazon Comprehend to create new products based on understanding the structure of documents. For example, using Amazon Comprehend you can search social networking feeds for mentions of products or scan an entire document repository for key phrases.

You work with one or more documents at a time to evaluate their content and gain insights about them. Some of the insights that Amazon Comprehend develops about a document include:

* **Entities** – Amazon Comprehend returns a list of entities, such as people, places, and locations, identified in a document.
* **Key phrases** – Amazon Comprehend extracts key phrases that appear in a document. For example, a document about a basketball game might return the names of the teams, the name of the venue, and the final score.
* **Language** – Amazon Comprehend identifies the dominant language in a document. Amazon Comprehend can identify 100 languages.
* **Sentiment** – Amazon Comprehend determines the emotional sentiment of a document. Sentiment can be positive, neutral, negative, or mixed.
* **Syntax** – Amazon Comprehend parses each word in your document and determines the part of speech for the word. For example, in the sentence "It is raining today in Seattle," "it" is identified as a pronoun, "raining" is identified as a verb, and "Seattle" is identified as a proper noun.

**Comprehend Custom**

Customize Comprehend for your specific requirements without the skillset required to build machine learning-based NLP solutions. Using automatic machine learning, or Auto ML, Comprehend Custom builds customized NLP models on your behalf, using data you already have. Training and calling custom comprehend models are both asynchronous (batch) operations.

* **Custom Classification:**

Create custom document classifiers to organize your documents into your own categories. For each classification label, provide a set of documents that best represent that label and train your classifier on it. Once trained, a classifier can be used on any number of unlabelled document sets. Customers can use the console for a code-free experience or install the latest AWS SDK.

* **Custom Entities:**

Create custom entity types that analyse text for your specific terms and noun-based phrases. Customers can train custom entities to extract terms like policy numbers, or phrases that imply a customer escalation. Customers provide a list of the entities and a set of documents that contain them, to train the model. Once the model is trained, customers can submit analysis jobs against it to extract their custom entities.

* **Document Clustering (Topic Modelling):**

You can also use Amazon Comprehend to examine a corpus of documents to organize them based on similar keywords within them. Document clustering (topic modelling) is useful to organize a large corpus of documents into topics or clusters that are similar based on the frequency of words within them.

Topic modelling is an asynchronous process, you submit a set of documents for processing and then later get the results when processing is complete. Amazon Comprehend does topic modelling on large document sets, for best results you should include at least 1,000 documents when you submit a topic modelling job.

Examples

The following examples show how you might use the Amazon Comprehend operations in your applications.

* **Example 1: Find documents about a subject**

Find the documents about a particular subject using Amazon Comprehend topic modelling. Scan a set of documents to determine the topics discussed, and to find the documents associated with each topic. You can specify the number of topics that Amazon Comprehend should return from the document set.

* **Example 2: Find out how customers feel about your products**

If your company publishes a catalogue, let Amazon Comprehend tell you what customers think of your products. Send each customer comment to the Detect Sentiment operation and it will tell you whether customers feel positive, negative, neutral, or mixed about a product.

* **Example 3: Discover what matters to your customers**

Use Amazon Comprehend topic modelling to discover the topics that your customers are talking about on your forums and message boards, then use entity detection to determine the people, places, and things that they associate with the topic. Finally, use sentiment analysis to determine how your customers feel about a topic.

Some of the benefits of using Amazon Comprehend include:

* **Integrate powerful natural language processing into your apps**

Amazon Comprehend removes the complexity of building text analysis capabilities into your applications by making powerful and accurate natural language processing available with a simple API. You don't need textual analysis expertise to take advantage of the insights that Amazon Comprehend produces.

* **Deep learning based natural language processing**

Amazon Comprehend uses deep learning technology to accurately analyse text. Our models are constantly trained with new data across multiple domains to improve accuracy.

* **Scalable natural language processing**

Amazon Comprehend enables you to analyse millions of documents so that you can discover the insights that they contain.

* **Integrate with other AWS services**

Amazon Comprehend is designed to work seamlessly with other AWS services like Amazon S3, AWS KMS, and AWS Lambda. Store your documents in Amazon S3 or analyse real-time data with Kinesis Data Firehose. Support for AWS Identity and Access Management (IAM) makes it easy to securely control access to Amazon Comprehend operations. Using IAM, you can create and manage AWS users and groups to grant the appropriate access to your developers and end users.

* **Encryption of output results and volume data**

Amazon S3 already enables you to encrypt your input documents, and Amazon Comprehend extends this even farther. By using your own KMS key, you can not only encrypt the output results of your job, but also the data on the storage volume attached to the compute instance that processes the analysis job. The result is significantly enhanced security.

* **Low cost**

With Amazon Comprehend, you only pay for the documents that you analyse. There are no minimum fees or upfront commitments.

**Chapter 5**

**Model Evaluation**

**Custom Classifier Metrics:**

Amazon Comprehend provides you with metrics to help you estimate how well a custom classifier should work for your job. They are based on training the classifier model, and so while they accurately represent the performance of the model during training, they are only an approximation of the API performance during classification.

Amazon Comprehend also support the following metrics:

* **Accuracy**
* **Precision (Macro Precision)**
* **Recall (Macro Recall)**
* **F1 Score (Macro F1 Score)**
* **Hamming Loss**
* **Micro Precision**
* **Micro Recall**
* **Micro F1 Score**

**Accuracy**

Accuracy indicates the percentage of labels from the test data that are predicted exactly right by the model. In other words, this is the fraction of the labels that were correct recognized. It is computed by dividing the number of labels in the test documents that were correctly recognized by the total number of labels in the test documents.

**Precision (Macro Precision)**

Precision is a measure of the usefulness of the classifier results in the test data. It's defined as the number of documents correctly classified, divided by the total number of classifications for the class. High precision means that the classifier returned substantially more relevant results than irrelevant ones.

The Precision metric is also known as *Macro Precision*

**Recall (Macro Recall)**

This indicates the percentage of correct categories in your text that the model can predict. This metric comes from averaging the recall scores of all available labels. Recall is a measure of how complete the classifier results are for the test data.

High recall means that the classifier returned most of the relevant results.

The Recall metric is also known as *Macro Recall*.

**F1 Score (Macro F1 Score)**

A combination of the Precision and Recall metrics. The F1 score is the harmonic mean of the Precision and Recall metrics. This score is based on the Precision and Recall created by the averaging method and is also known as the Macro F1 score. A measure of how accurate the classifier results are for the test data. It is derived from the Precision and Recall values. The F1Score is the harmonic average of the two scores. The highest score is 1, and the worst score is 0.

The F1 Score metric is also known as the *Macro F1 Score*.

**Hamming Loss**

The fraction of labels that are incorrectly predicted. Also seen as the fraction of wrong labels compared to the total number of labels. Scores closer to zero are better.

**Micro Precision**

As Precision above, except that instead of averaging the precision scores of all available labels, this is based on the overall score of all precision scores added together.

**Micro Recall**

As Recall above, except that instead of averaging the recall scores of all labels, this is based on the overall score of all recall scores added together.

**Micro F1 Score**

As F1 Score above, but instead a combination of the Micro Precision and Micro Recall metrics.

**Chapter 6**

**Scope of Enhancement**

A lot of businesses have data that is highly unstructured. This makes it extremely difficult for them to analyze, understand, and sort data on a huge scale. To solve this problem, businesses leverage text classification with machine learning because of its scalability and real-time analysis of unstructured data. This, in turn, saves time, automates business processes, and helps make informed business decisions.

Below are some of the scope of topic classification:

* **Speed:**

These days everybody wants and expects immediate turnaround of any process. Delaying a business process in insurance, credit, retail or any other business because someone needs to go through a huge stack of papers is unthinkable.

* **Efficiency**:

While technology is not always cheap and not every manual action is a candidate for automation, most of the time automation brings about process reliability and economic sense to the business.

* **Economies of scale:**

And once automation is introduced the ability to grow the business in a fast and agile manner is available for the management who can tackle growth peaks by just adding some - or none at all! - technical resources.

**Chapter 7**

**Resources Link**

* **Data available at**<http://mlg.ucd.ie/files/datasets/bbc-fulltext.zip>
* [https://docs.google.com/document/d/15kFOSLPjY2j-xVp\_rPFt82Sfq0PUT\_CIA8bC1nLDKng/edit?ts=5ea28c7a#](https://docs.google.com/document/d/15kFOSLPjY2j-xVp_rPFt82Sfq0PUT_CIA8bC1nLDKng/edit?ts=5ea28c7a)
* <https://docs.aws.amazon.com/comprehend/latest/dg/comprehend-general.html>
* <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/comprehend.html>
* <https://docs.aws.amazon.com/comprehend/latest/dg/getting-started-console-classifier.html>

**Chapter 8**

**Conclusion**

With the help of AWS Comprehend on the given data set, key matters were drawn in classifying 5 different class label.

For this internship, we have used the SDK to interact with the AWS Comprehend APIs to perform the required tasks. An SDK is available to integrate your customer classifier into your current applications. The Custom Classification API enables you to easily build custom text classification models using your business-specific labels without learning ML. For example, your customer support organization can use Custom Classification to automatically categorize inbound requests by problem type based on how the customer has described the issue. Creating a custom model is simple. You provide examples of text for each of the labels you want to use, and Comprehend trains on those to create your custom model.

In our project, the overall approach was to prepare the training dataset from the given data set and train the model using AWS custom comprehend. And use the trained model to predict the topic of the new document using AWS comprehend APIs.

**References:**

* <https://monkeylearn.com/blog/introduction-to-topic-modeling/>
* <https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html>
* <https://docs.aws.amazon.com/comprehend/latest/dg/how-document-classification.html>
* <https://docs.aws.amazon.com/comprehend/latest/dg/API_ClassifyDocument.html>
* <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/comprehend.html>
* <https://aws.amazon.com/comprehend/pricing/>