Comparison of sentiment analysis using NLP libraries vs Amazon Comprehend

Sentiment Analysis refers to the process where we analyze and find the emotion or intent behind a piece of text or speech or any mode of communication. We, humans, communicate with each other in a variety of languages, and any language is just a mediator or a way in which we try to express ourselves. Therefore, whatever we speak or write, it has a sentiment associated with it. The sentiment might be positive or negative or neutral as well. There are various ways in which sentiment analysis can be performed and we plan to compare the process followed for sentiment analysis using standard nlp libraries with the process followed for doing the same through Amazon Comprehend. Amazon Comprehend is one of web services offered by Amazon (AWS) which is based on natural-language processing and uses machine learning to uncover valuable insights and connections in text.

The first step towards getting started with sentiment analysis would be to get hold of a relevant dataset. Some examples could be to use the movie review dataset published by Cornell or a product review dataset from Amazon. Once we know the dataset that we are going to use, for the case of standard workflow, we would need to import the relevant libraries like numpy, nltk, sklearn etc and import the dataset using libraries like pandas dataframe. In the case of Amazon Comprehend, we would need to upload the dataset to an object storage service called amazon S3. Real life text data is messy and would have a lot of unnecessary aspects in it which we need to get rid of before we start our analysis. The reason is because the insights gained from messy data would not be very useful and they would be hard to interpret as well since non-crucial tokens like punctuations, spaces, links etc will ruin the analysis that we will perform moving ahead. Substituting the unnecessary tokens using regex is the most popular way used to clean text. For amazon comprehend, the preprocessing is taken care of under the hood, irrespective of whether we use amazon's api for sentiment analysis or write a custom classification model where we tweak some of the parameters. Once we have a clean dataset, the next step would be to transform the text into a meaningful representation of numbers which is used to fit machine algorithms for prediction. TF-IDF vectorizer from sklearn is popular to do the same since it gives importance to count of occurrences of words (TF) but at the same time rewards the occurrence of rare words (IDF) to represent the text in the best form possible. For amazon comprehend, the vectorization technique used also happens under the hood and we don't know which technique is used or how to change it. Now that we have the text in an appropriate representation, it's time to split our dataset into a training part and the testing part. This is necessary because we don't want our model to learn using all the data in our dataset, if we do that, then we won't have any relevant data to evaluate the performance of the model. Also, splitting the data in such a way ensures that we have some unbiased samples which the model has never seen and therefore lets us check how the model is doing on new data. Sklearn provides a function called "train_test_split" which can be used where we can define the percentage of data that we want to keep for the test data and it will randomly split and keep that percentage of data aside for testing the model. In amazon comprehend, we have two options, either we can go with autosplit which selects 10% of our provided training data for testing or we can exactly specify the test set we want to use which would be the location of the test data uploaded as an object in the S3 service.The next step ahead would be to start fitting a machine learning model to the training data we have got in the last step. Since the task here is sentiment analysis, the simplest case would be to have 2 classes as negative and positive based on the sentiment of the text. Logistic Regression is one of the many machine learning models which can be applied in order or perform this two class classification. Sklearn provides functions which can be used to instantiate the Logistic Regression model and run the model by passing the training documents along with their labels. The Amazon comprehend counterpart for this would be to run a job called "`start-sentiment-detection-job`" which would take the relevant input-output s3 bucket locations,resource access permissions and language to use for parameters. We would get a relevant job-id which can be used to track the status of the job running on amazon comprehend when called using the AWS command line interface. Since the "`start-sentiment-detection-job`" is a black box for us, we don't know whether amazon uses logistic regression or some other machine learning model in order to do the classification. Now that we have trained our model to learn based on the data we have provided, the next step would be to pass our "unknown data" which would be the test documents and ask the model to predict the class values for them. We can do this by using the instance of Logistic Regression instantiated before and by calling the "predict" function on it. For Amazon comprehend, based

on the way we have given the split for test data (autosplit) vs custom data, the job would have predicted the values for test data also during that job run itself. Now that we have got the predicted values based on the machine learning model, it is important to compare those values with the actual classes that were present in the test dataset so that we can know how well our model has performed. One of the popular ways to look at this is through the confusion matrix which is a table with 4 different combinations of predicted and actual values and it helps in calculating different performance measurement measures such as accuracy of the machine learning model. Sklearn provides a function called the confusion_matrix which takes the predicted values for the test data and the actual class values for the test data to create the matrix. While working on amazon comprehend, the confusion matrix is available when running the "CreateDocumentClassifier" API operation. When the operation is run, the confusion matrix is shown in the confusion_matrix.json file, located at s3://*user-defined-path*/*unique-value*/output/output.tar.gz where the user-defined-path is the S3Uri value of the OutputDataConfig parameter in the "CreateDocumentClassifier" operation. Now that we have run the model as well as got the accuracy for it, it would be a good idea to visualize our predictions. One way to do so could be to check the total number of documents that were predicted as positive and total number of documents that were predicted as negative. Matplot has a pyplot functionality which is used popularly in order to make such visualizations.An example could be to create a bar graph having the count of positive documents and negative documents. The Amazon comprehend counterpart for this is a bit more complicated. We cannot directly visualize the results that we have got after running the "start-sentiment-detection-job", instead, to prepare the results of the sentiment job for creating data visualizations, we need to use AWS Glue and Amazon Athena. AWS Glue provides a *crawler* that explores our data and automatically catalogs it in tables in the AWS Glue Data Catalog (a serverless database). The issue however is that the database that gets populated in Glue Data Catalog has nested results. Therefore, we need to run a few SQL statements to unnest them using Amazon Athena. Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. After running the SQL commands on Athena, our sentiment results table would have columns such as filename, line number, sentiment, mixed, negative, neutral, and positive. The table would have one value per cell. The sentiment column would describe the most likely overall sentiment of a particular review. The mixed, negative, neutral, and positive columns would give scores for each type of sentiment. After storing the Amazon Comprehend results in tables, we can connect to and visualize the data with Amazon QuickSight. Amazon QuickSight is an AWS managed business intelligence (BI) tool for visualizing data.To start the visualization process, we will import our unnested sentiment table from Amazon Athena. Now that we can access our data in Amazon QuickSight, we can begin creating visualizations. We can create a pie chart that can show the count of rows in our test data which are positive, neutral, mixed, or negative sentiment associated with them.

Therefore, to conclude, we saw and compared the two different ways to perform sentiment analysis. The first way was using the standard nlp process where we observed that we got more flexibility in terms of choosing the vectorization techniques to use or the models to use when performing sentiment analysis and the visualization process was simple.The second way was through amazon comprehend which is a much faster way to perform sentiment analysis when we want to get quick results because comprehend takes care of data pre-preprocessing, vectorization and training under the hood, however, the visualization aspect is a bit more involved when it comes to using amazon comprehend since it requires us to understand and interact with many more services like AWS Glue, Amazon Athena and Amazon QuickSight even to perform simple visualizations.

References:

1. https://medium.com/analytics-vidhya/nlp-getting-started-with-sentiment-analysis-126fcd61cc4a
2. https://aws.amazon.com/comprehend/
3. https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
4. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
5. https://github.com/bijoyandas/Hands-On-Natural-Language-Processing-with-Python
6. https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html
7. https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-amazon-comprehend/