1. Part A. Read AWS official documentation on SageMaker and Comprehend. Learn how these services are used in event-driven applications and explore various use cases.
2. Part B. Build an event-driven serverless application using AWS SageMaker.

In this part of the assignment, you need to use S3 bucket, and Lambda Functions.

1. Create your 1st S3 bucket SourceDataB00xxxxxx and upload the files (from 001 to 299) given in the Train folder. You need to write a script or use the SDK to upload the files on the bucket.

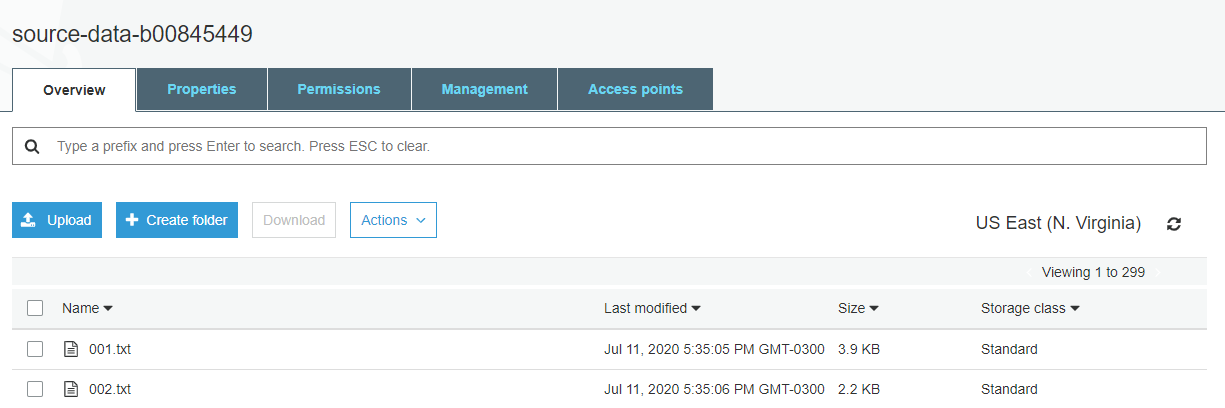


Figure a source-data-b00845449

All training files are uploaded to the bucket named “source-data-b00845449”.

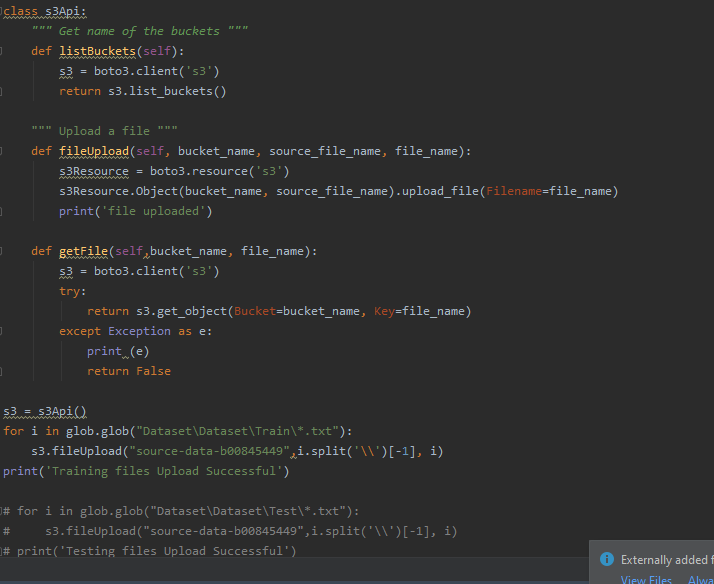


Figure b bucket upload program

Above is the script which uploads all the training files to the bucket.

1. Once a file is uploaded, a Lambda function - “generateVector” should extract words from all the files (remove the stop words). Then compute Levenshtein distance between the Current, and Next word.

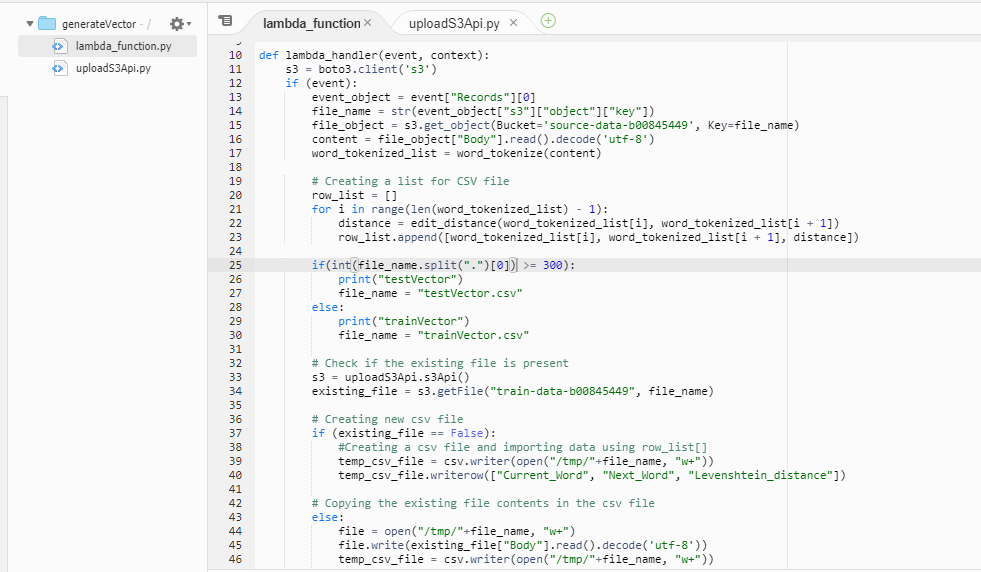


Figure c generateVector program 1

After the files are uploaded, the lambda function fetches the files, and find lavenshtein distance between the adjacent words of the file. The csv file is generated and is uploaded to another bucket “train-data-b00845449”. Each time the lambda function checks the filename because the files 000-299 are training files and hence will be appended to trainVector.csv. If the filename is between 300-401, then the lavenshtein distance of words will be appended to testVector.csv

The program checks if the file “trainVector.csv” or “testVector.csv” is present in the bucket or not; if present, then it will append the data otherwise it will create a new file with the headings and send it.

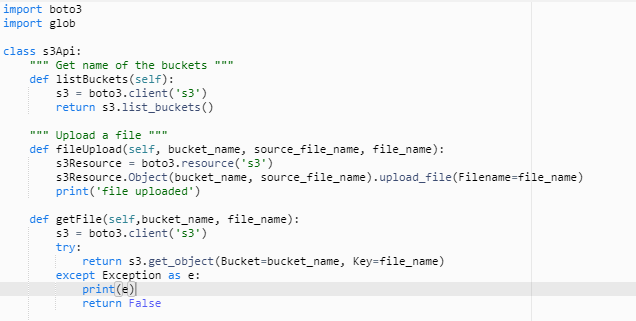


Figure d generateVector program 2

Above is the script used for the uploading and fetching the files from the AWS S3. This file is being called in the main lambda function.

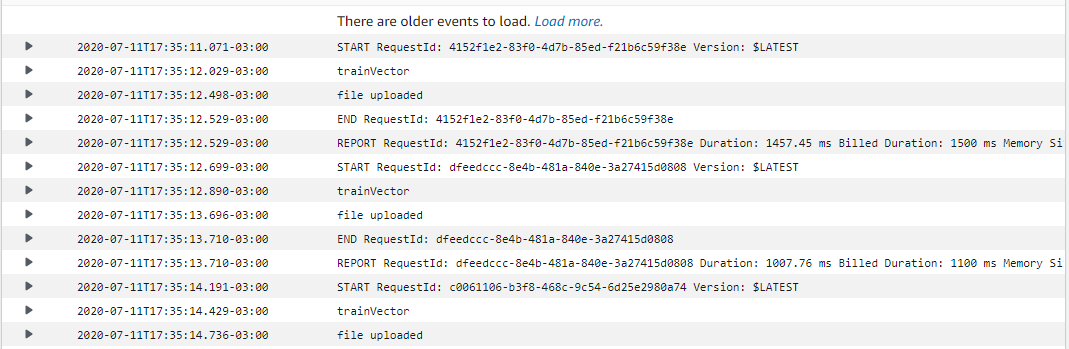


Figure e cloudwatch logs for generateVector

Above is the snapshot of cloudwatch logs which shows the log of files uploaded and also the file (“trainVector or testVector”) where it gets appended.

1. This file (“trainVector.csv”) is saved in a new bucket TrainDataB00xxxxxx.

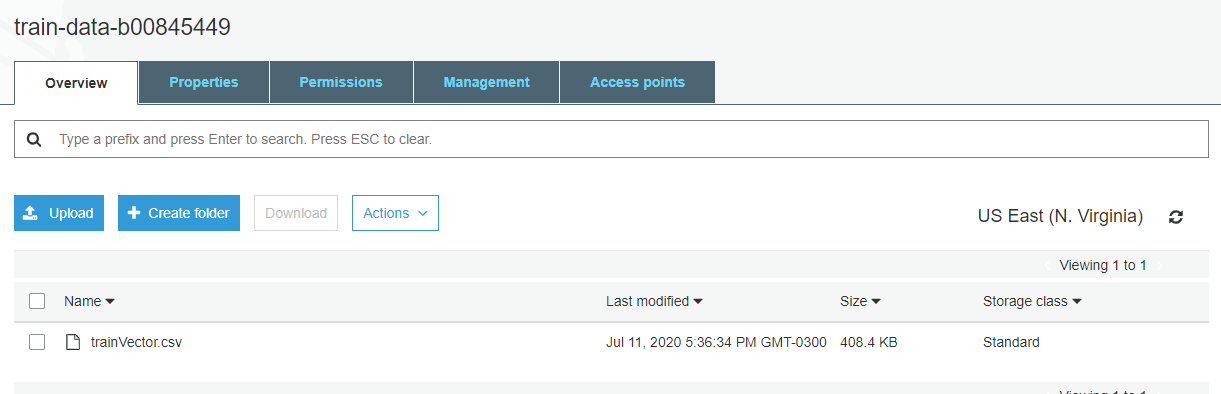


Figure f train-data-b00845449

As the above screenshot shows that the trainVector.csv file is getting generated from the lambda function “generateVector” and the lavenshtein distance of all the words of the training data is then appended to the same file.

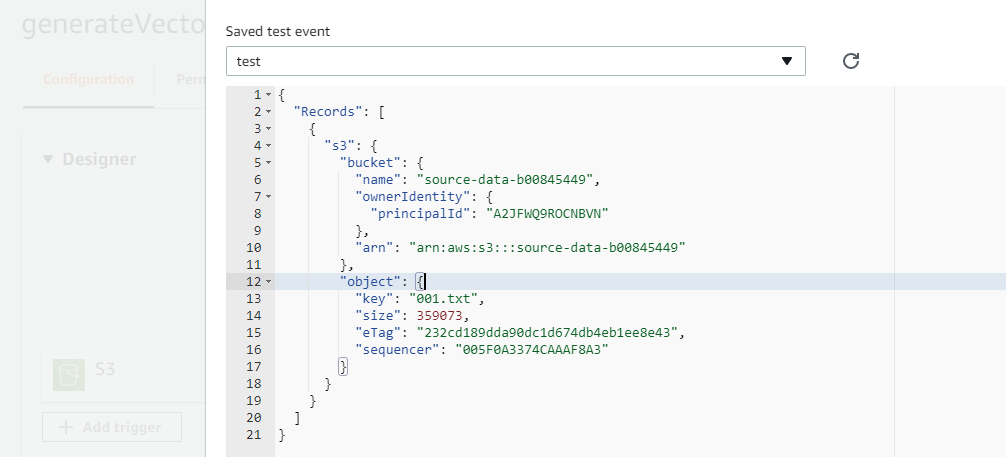


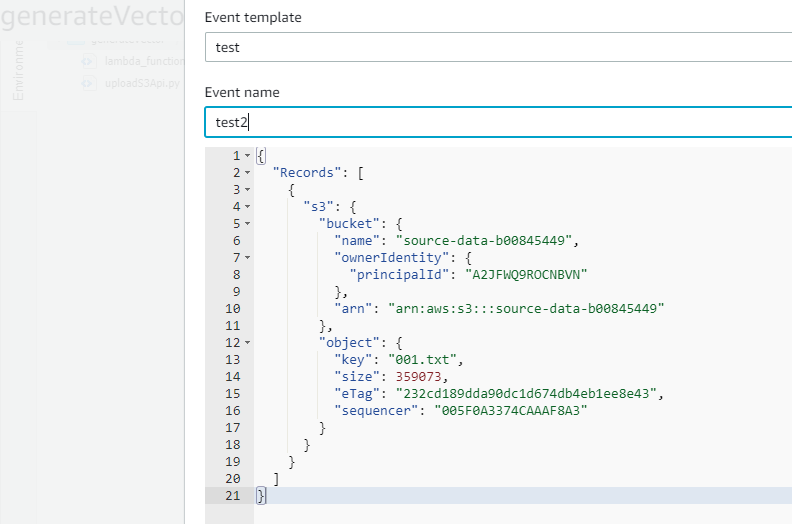
Figure g sample of csv file

Above figure shows the csv file sample. The columns are “FirstWord”, “NextWord”, “Lavenshtein distance” respectively.

Note: As the AWS educate account does not provide permission to complete further task which requires AWS Comprehend, hence no further work could be done.

Testing for generateVector lambda function





1. Part C. Build an event-driven serverless application using AWS Comprehend.
2. Create your 1st S3 bucket TwitterDataB00xxxxxx and upload the files given in the tweets folder. You need to write a script or use the SDK to upload the files on the bucket.

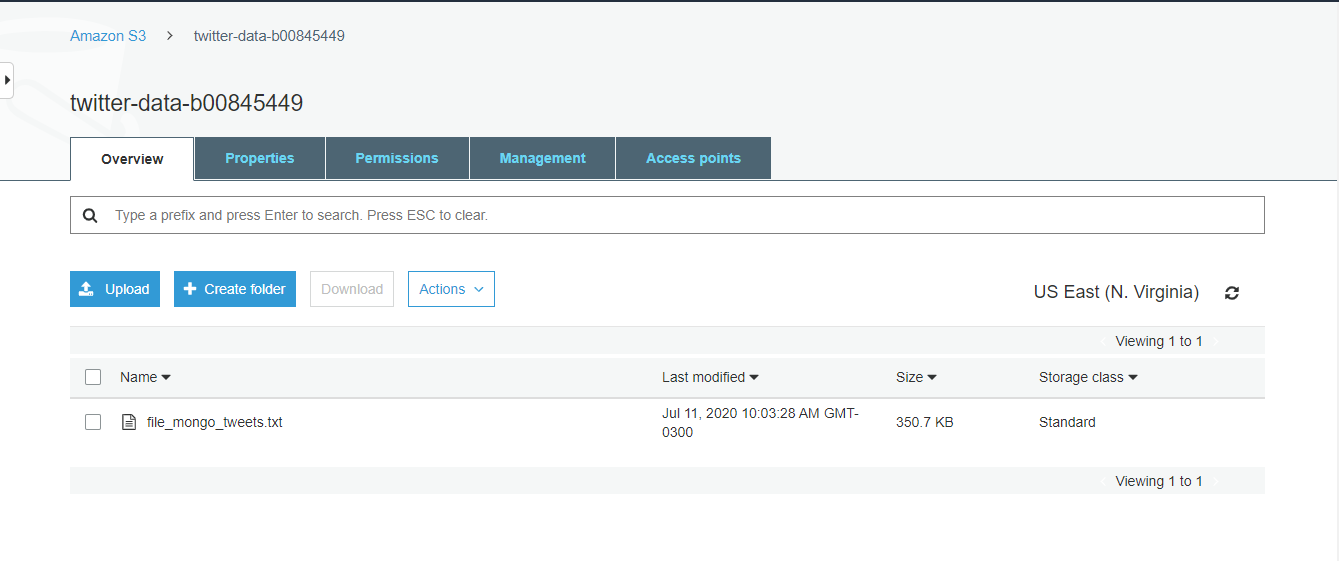


Figure h twitter-data-b00845449

Above figure shows the tweets file uploaded to the bucket named “twitter-data-b00845449”.

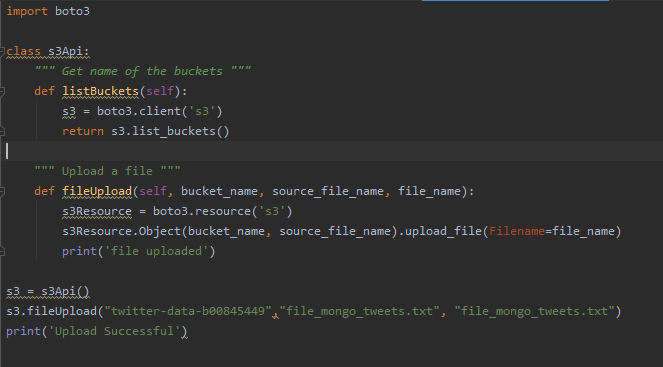


Figure i uploading tweets to bucket

Above is the snapshot of the script which is used to upload the file to the bucket.

1. To perform any pre or post processing of the files, you can write Lambda functions.

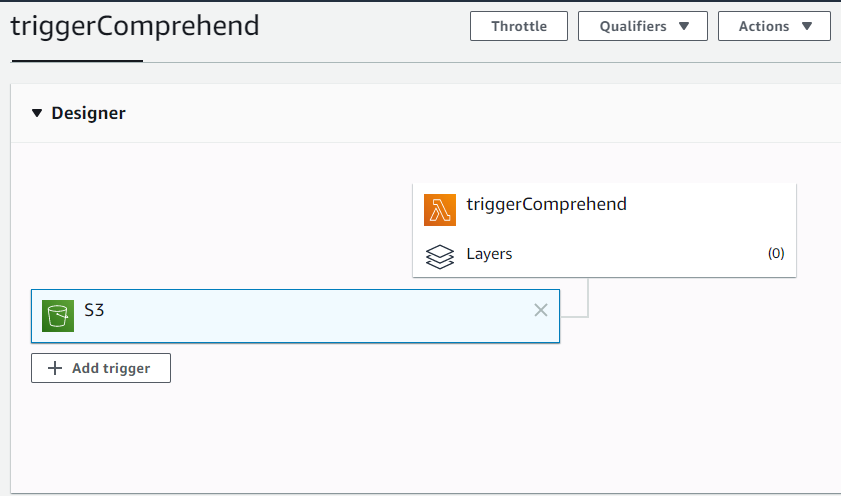


Figure j triggerComprehend lambda trigger

The lambda function named “triggerComprehend” is created to do preprocessing task. The lambda function is triggered on PUT operation of S3 bucket named “twitter-data-b00845449”.

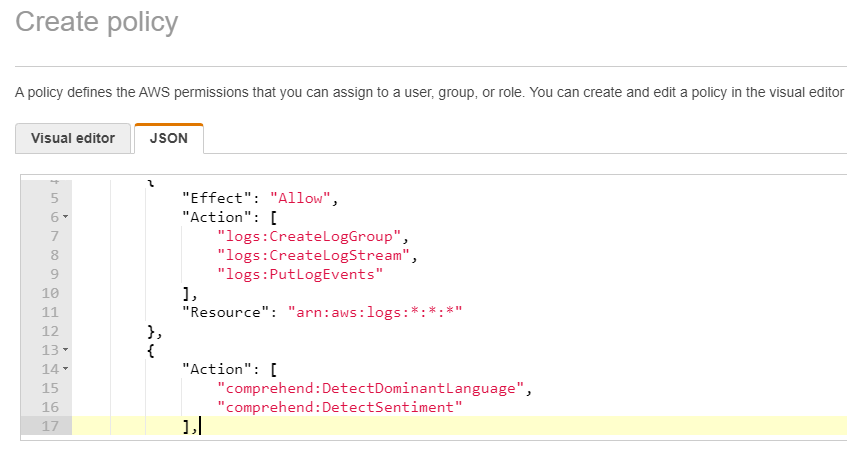


Figure k create policy

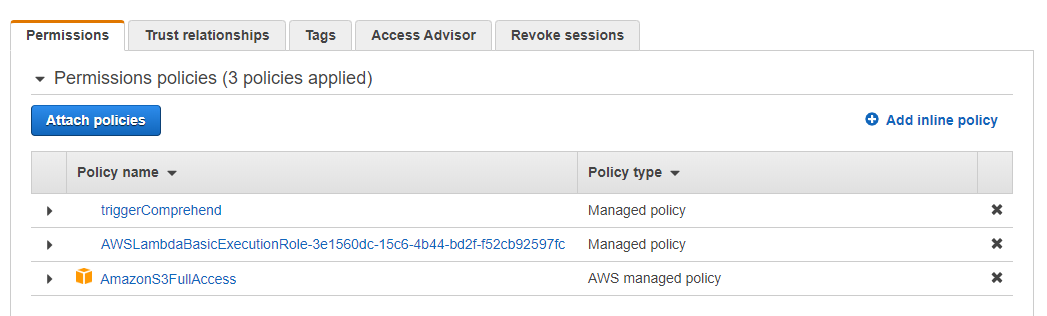


Figure l permissions and role

The policy for Lambda function is created and attached to the role. A policy to access Comprehend is also attached.

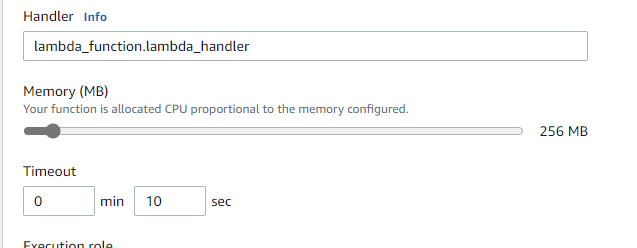


Figure m triggerComprehend settings

As the processing will require more time and memory, the memory is increased to 256 MB and time is increased to 10 seconds.

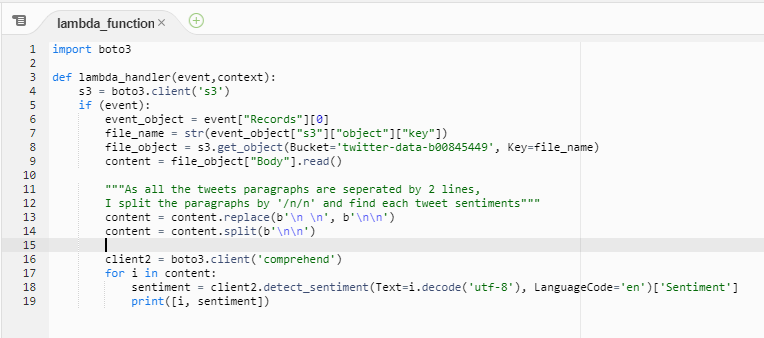


Figure n triggerComprehend function program

Above is the script of the lambda function which fetches the content from each bucket object and detects sentiment. I analysed that each tweet is separated by a single line, hence I created list of tweets by splitting using ‘\n\n’. AWS Comprehend is used to find sentiment of each tweet.

1. Once all the files are uploaded on the bucket, AWS Comprehend is used to perform sentiment analysis of tweets.

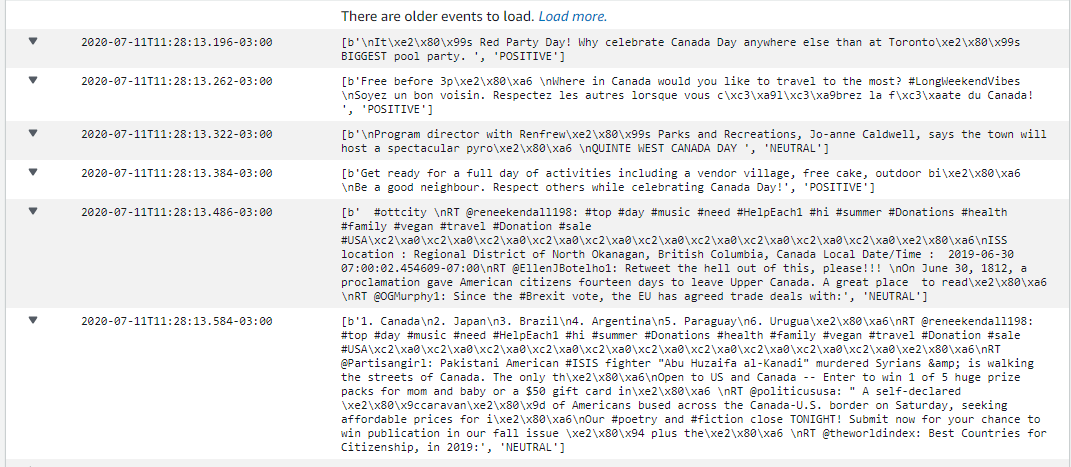


Figure o triggerComprehend cloudwatch logs

After the lambda function detects sentiments of each tweet, we can see the result on cloudwatch logs. Each log shows the tweet and the sentiment (Positive, Negative, Neutral).

Testing for triggerComprehend lambda function

