Paper 4940-2020

Real-time call analytics using SAS® Viya® and open source platforms

Ranjit Jangam, Comcast; Anvesh Reddy Minukuri, Comcast;

ABSTRACT

IVR calls in call centers is one of the primary engagement channels for customers to interact with service and product provider companies. Call data is very rich and can play a big role in improving customer experience e.g. provide real-time agent recommendation and customer feedback. This data can also be used for post-call analytics to help drive effective and efficient future handling of calls and improve agent performance thereby meeting company service level agreements (SLA) e.g. first call resolution. Customer call interaction can be captured by converting audio transcripts to text. However, there are very few systems that are capable of manually converting audio transcripts to text on a real-time basis. SAS® Viya® provides a variety of new capabilities, one of these capabilities being voice transcriptions and its execution on cloud on a real-time basis. Viya also supports interactions with other open source technologies which extends machine learning solution development capabilities and supports real-time API integration with ease.

Objective of this paper is to analyze customer audio transcripts between agents and customers to better understand real-time customer/agent responses and measure customer satisfaction using SAS Viya audio translator and Visual Text Analytics/EM Text Mining. Topics are extracted from audio transcripts along with sentiments/emotions (positive or negative), and call intent is identified from complex unstructured audio transcripts to determine appropriate agent recommendation. These extracted features can be used for various analytical purposes to assess company's SLA's on a real-time basis including first call resolution, abandon rate, repeat callers to name a few. This project will establish a foundation to leverage SAS VIYA and REST API- Python SWAT package on a real-time basis for accelerated insightful agent recommendations. See below for few applications of this paper:

- Improved customer experience during a call, driven by sentiment analysis identifying number of positive & negative sentiments
- Extract topics (text categorization) that identifies drivers of poor customer experience. Topic modeling can also help with post-call analytics to mitigate repeat callers, mapping callers to the right agent (agent routing), improved agent training to better handle calls in future
- Augmented predictive power of customer-centric models by utilizing extracted topics used as input features in modeling framework to improve model significance
- Identify agent traits and behavioral elements that improves customer experience through first call resolution and minimizing churn
- Maximize Care KPI's e.g. first call resolution, call quality monitoring to gauge transactional net promoter score (tNPS),

INTRODUCTION

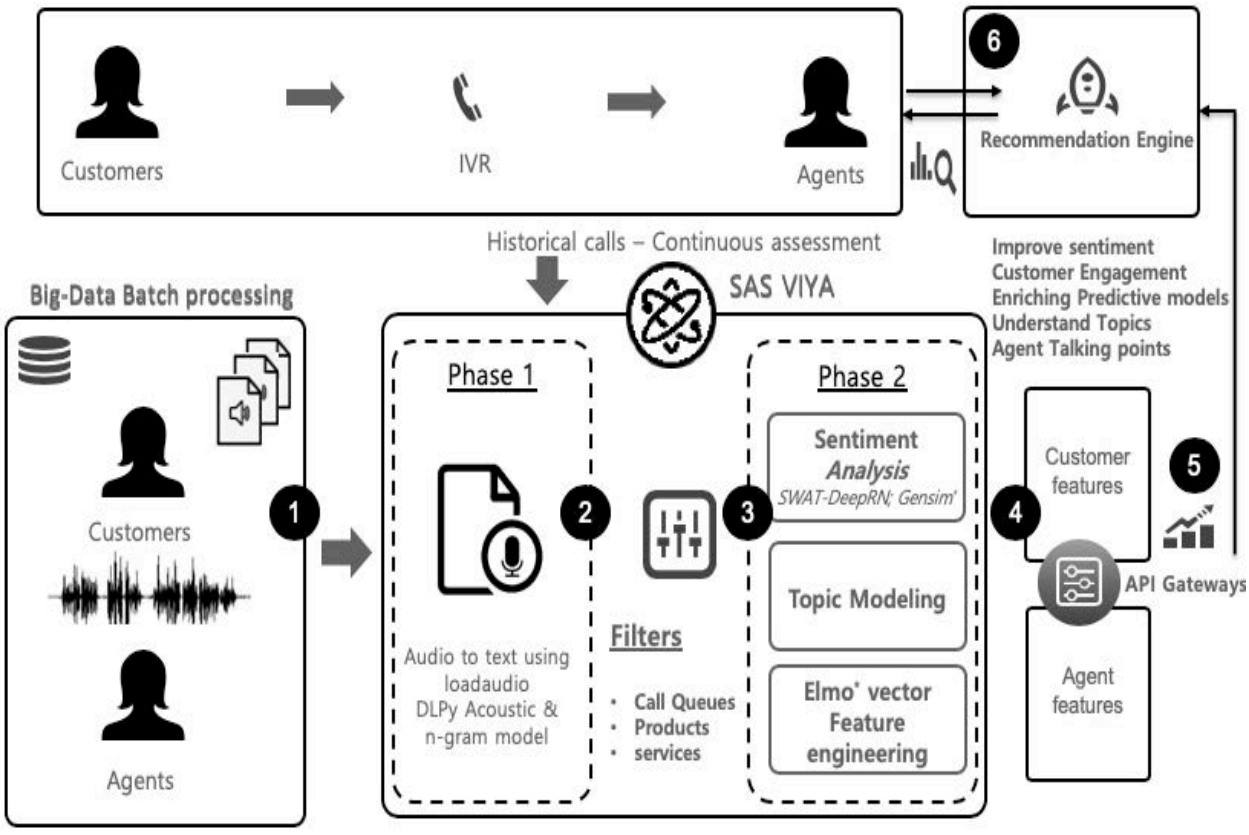
Over the decade, Call centers have been one of the most preferred engagement channels for customers to interact with companies for their products and services. Research has shown,

more than 70% of customer interactions are handled by call center agents. Customer engagement with call center agents is one of the top measures to gauge customer satisfaction & experience. With customer experience being one of the major SLA's for almost all companies across all industries, it is crucial to analyze various elements of call data continuously and on a real-time basis.

For this analysis, a sample set of audio files were utilized between agent and customers to extract and uncover insights on a real-time basis using Audio translator, NLP machine learning and linguistic rules.

This paper will highlight IVR call analytics using SAS Viya CAS architecture on Big-Data environment. This paper will review real-time SAS translator and NLP that understands both agent and customer interaction signals in larger context of their conversation to make the right recommendation and improve customer experience. This paper won't illustrate transcription costs and duration times associated with call transcription due to limited data that is utilized for this demonstration.

Display 1: Overview of IVR call Analytics Architecture. SAS Viya provides extensive integration to Big-Data platforms, open source algorithms, REST API's



Note: 1. Viya platform provides power of NLP , ML & Linguistic Rules. For max benefit, install on Bigdata platform
2. * represents the open source execution on Viya platform.

VIYA CALL ANALYTICS PROCESS OVERVIEW

Phase 1:

1. Ingest: Big-Data batch processing of IVR Audio files; VIYA CASUTIL for loading audio files
2. Speech to Text: SAS CAS Load Audio to translate IVR Audio files to text files (Acoustic, N-gram)

Phase 2:

1. Text Mining: CAS and Open source Text Analytics (NLP: DeepRN, Gensim for Polarity)
2. Keywords & Topics: Topic Modeling – LDA, Feature Engg. – Elmo Vectors

Phase 3:

1. Predictive Insights
2. API's to gauge the various engagement assessments and flexible integration

OVERVIEW OF BIG-DATA VIYA PLATFORM AND BATCH PROCESSING

SAS Viya has an excellent integration with Hadoop system that can leverage computing power available on the edge nodes. SAS Viya can be installed on these nodes to create an orchestration that can read huge amounts of data and process them in parallel using the existing infrastructure.

SAS can ingest aggregated audio file data in a distributed file system. This data can be processed on a daily basis to create insights and can be customized to organizational needs. Multiple reports can be run on these aggregations at 1, 6, 12, 24-hour windows that may benefit stakeholder requests and drive necessary insights.

Below is sample script that can ingest data into the engine. Below code sets a connection to hive libraries using CASUTIL. This can seamlessly read the data stored on distributed file system and read it memory. SAS Viya provides different levels of data abstraction: data can be used locally for a session or persist for a user or promote it to larger user base depending upon the use case.

```
options msglevel=N nosource nosource2 casdatalimit=ALL;caslib hivelib drop;
caslib hivelib datasource=
(srctype="hadoop"
server="company-cluster-nodes"
hadoopconfigdir="/opt/sas/hadoop/sitexmls/"
hadoopjarpath="/opt/sas/hadoop/jars/"
dtm='Parallel'
dfDebug=epAll
properties='mapreduce.job.queuename=queuename'
    schema="abc");
/* List Hadoop/Hive Tables */
proc casutil;
list files incaslib="hivelib" ;
```

```
quit;proc casutil incaslib="casuser";
   droptable casdata="indata_sas";
run;
/*Load  source table from Hadoop to  target table in CAS */
proc casutil;
load casdata='sastable' incaslib=hivelib outcaslib=casuser
 casout='indata_sas';     *casdata=HDP source table, casout =CAS target
table;
quit;
```

PHASE1: AUDIO TRANSLATOR – DLPY CAS LOAD AUDIO & FILTERS

Speech processing of audio data on Viya is transcribed using audio action set. Viya Hadoop installation provides us with flexibility to directly feed audio files from distributed platform to SAS Cloud Analytical Services (CAS) tables. Acoustic model and language model are then leveraged for sound files to predict text (transcripts).

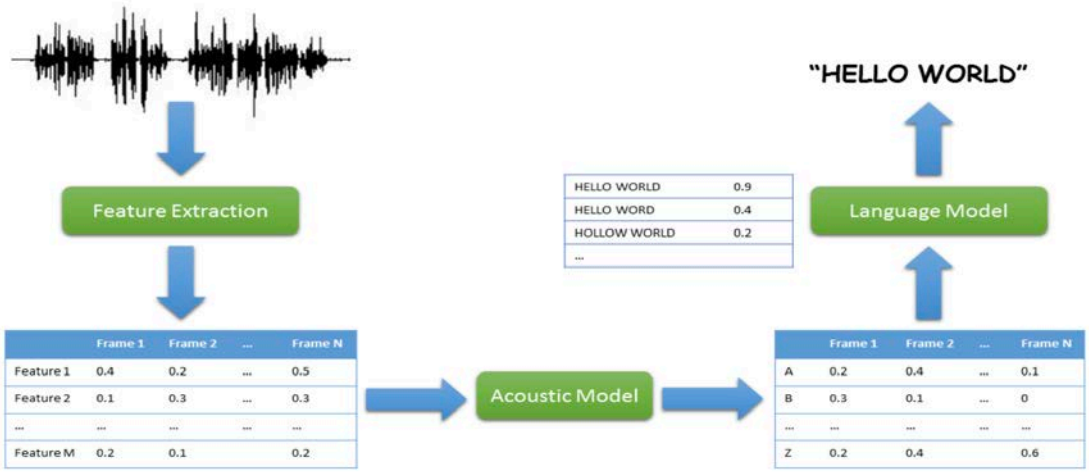
1. Audio files are loaded using load audio function, features are extracted using MFCC - Mel Frequency Cepstral Coefficients (speech processing). Feature extraction is the main part as it helps extracting the features that helps in translating using Acoustic model.
2. Acoustic Speech recognition model is created using audio tables that has both the MFCC and metadata features. Similarly, language model is imported, and n-gram model is used. In this particular case, pre-trained acoustic model is available and hence will not be covered in this topic.
3. Create a speech object by utilizing the audio files, Acoustic and n-gram model
4. Make use of transcribe () method to convert speech to text. The transcribe () or language model decode will yield the transcripts of IVR audio files. Transcript accuracy was ensured using the Character/word/sentence error rate and acoustic model was accessed using fit error metric.

```
#Configure the Environment
from swat import *
#Next, import dlpy.speech.
import dlpy.speech
s = CAS(cashost, casport)
#create a speech object
spch2txt = dlpy.speech.Speech(s,
                              data_path= "/user/hive/warehouse/path",
                              local_path= "\\localpath",
                              acoustic_model_path=
"/user/hive/warehouse/acoustic_x.sashdat",
                              language_model_path=
"/user/hive/warehouse/language_model.sashdat")
```

```
#convert speech to Text
import IPython.display as ipd
ipd.Audio("ivr_35969.wav
#speech. Transcribe () method to convert speech to text.
spch2txt.transcribe("ivr_35969.wav")
```

```
In [ ]: # step 1: import the SAS Scripting Wrapper for Analytics Transfer library
        from swat import *
        #import the dlpy speech package
        import dlpy.speech

        # step 2: start a CAS session
        s = CAS(host, port)

        # step 3: load CAS action sets
        s.loadactionset("audio")
        s.loadactionset("deepLearn")
        s.loadactionset("langModel")

        #Add model and data caslib
        s.addCaslib(name = "model", path = "/bigdisk/lax/xiaoli/",
                    dataSource = dict(srcType = "path"), subdirs = "TRUE", activeOnAdd = "FALSE")
        s.addCaslib(name = "data", path = "/bigdisk/lax/xixche/",
                    dataSource = dict(srcType = "path"), subdirs = "TRUE", activeOnAdd = "FALSE")

        # step 4: load audio data for test set
        s.audio.loadaudio(path    = "test.listing",
                          casOut = {"name": "test_audio", "replace": True},
                          )

        # step 5: extract MFCC features from audio files for test set

        s.audio.computefeatures(
            table                  = "test_audio",
            casOut                 = {"name": "test_dlscore_input", "replace": True},
            audioColumn            = "_audio_",
            copyvars               = ["_path_"],
            frameExtractionOptions = {"frameShift": 10, "frameLength": 25, "dither": 0.0},
            melBanksOptions        = {"nBins": 40},
            mfccOptions            = {"nCeps": 40},
            featureScalingMethod   = "STANDARDIZATION",
            nOutputFrames          = 1500,
        )

        # step 6: import the pretrained RNN acoustic model if it's not in-memory
        s.table.loadtable(path = "acoustic_model.sashdat",
                          casOut = {"name": "asr", "replace": True},
                         )
        s.table.loadtable(path = "acoustic_model_weights.sashdat",
                          casOut = {"name": "pretrained_weights", "replace": True},
                         )
        s.table.loadtable(path = "acoustic_model_weights_attr.sashdat",
                          casOut = {"name": "pretrained_weights_attr", "replace": True},
                         )
        s.table.attribute(task = "ADD",
                          name = "pretrained_weights",
                          attrtable = "pretrained_weights_attr"
                          )

        # step 7: use pretrained RNN model to score acoustic features of test set
        s.dlScore(table          = "test_dlscore_input",
                  model          = "asr",
                  initweights    = "pretrained_weights",
                  layerImageType = 'WIDE',
                  copyvars       = ['_path_'],
                  casOut         = {"name": "test_rnn_scores", "replace": True},
                  )

        # step 8: import an n-gram language model
        s.langModel.lmImport(table  = "language_model.sashdat",
                             casOut = "language_model"
                            )

        # step 9: decode RNN scores
        s.langModel.lmDecode(table          = "test_rnn_scores",
                             langModelTable = "language_model",
                             blankLabel     = " ",
                             spaceLabel     = "&",
                             copyvars       = ['_path_'],
                             casOut         = "test_results"
                            )
        # step 10: save the results
        s.save(table = "test_results",
               name  = "test_results.csv",
               replace = True
```

Figure 1: Audio translation workflow



IVR DATA: SAMPLE VIEW OF TRANSCRIPTS FROM AUDIO ACTION SET

See below sample view of customer and agent interaction that are translated using SAS Viya. Transcripts can also be transcribed with a timestamp. The higher accuracy translation and logical timestamp points will allow us to assess text mining elements in a chronological order that leads to path analysis.

Display 2: Sample view of call transcript

##########

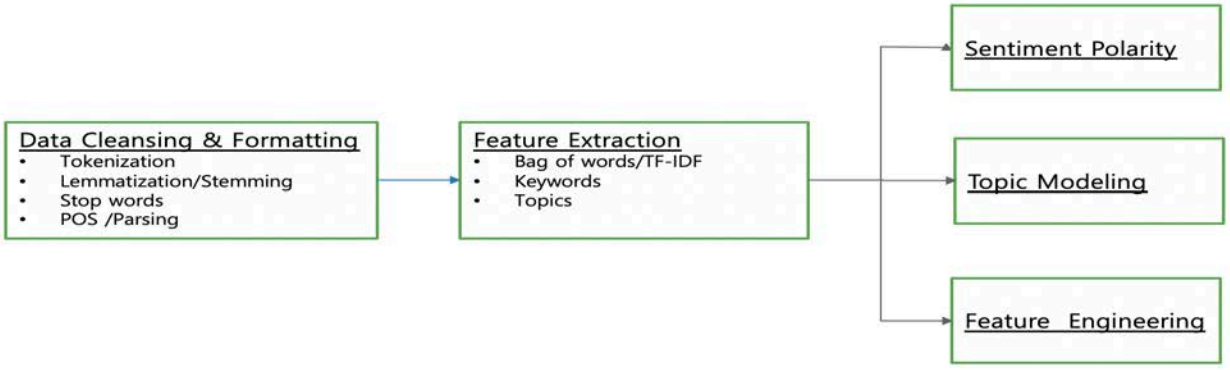
Agent: Thank you for choosing Company this is agent1 and how may i help you
customer1: hi um i was just calling to see how much it would cost um for your internet
Agent: I'll be more than happy to review our internet packages with you and who am I speaking with
customer1: abc thank you turn it yeah
Agent: Is it okay to okay sure ask you a few questions to better assist you
customer1: sure alright
Agent: may i have the address where you'd like to set up services?
customer1: Okay.
Agent: And is there an apartment or unit number?
Customer: Apartment
Agent: The price of package is sixty nine ninety nine
##########

Agent1: Thank you for choosing Company. Home of Xfinity. This is xxx. How may I help you?
Customer2: my name is abc and I'm calling to uh set up some service here today.
Agent1: Okay Miss I can definitely assist you here. Do you mind if I ask you a couple of questions to get you started?
Sure.
Customer2: Alright.

PHASE2: TEXT ANALYTICS USING SAS VIYA SUITE AND OPEN SOURCE INTEGRATION

Text mining enables us to transform unstructured corpus documents into meaningful and actionable information. It determines keywords, patterns, and topics associated with corpus text quickly and summarizes the data. Raw data is processed by applying data formatting and treatment techniques to the corpus text before applying Topic modeling/Sentiment Analysis/Features extraction.

Figure 2: Text analytics workflow overview



Data cleansing and formatting are done by utilizing SAS Viya NLP and Linguistic rules. The SWAT package allows us to utilize NLTK to pre-process text corpus and performs bag of words, Bigrams, lemmatization and stemming, stop words, TF-IDF. Dictionary is then created to perform topic modeling, sentiment analysis, feature engineering. SAS Viya integrates into both open source and Viya DLPy python support. SAS also has additional tools e.g. Visual Analytics specifically designed to analyze entire life cycle of unstructured corpus text. Visual analytics software is very intuitive and flexible to use. It supports automatic topic discovery – SVD & LDA, Sentiment analysis, Rule Based flexibility.

SENTIMENT ANALYSIS

Sentiment Analysis allows us to determine the emotions expressed by both customer and agent from the call data. Unsupervised and supervised algorithms are used to perform emotion recognition which aims to evaluate the model. SAS Viya Deep RNN action set and open source gensim are used to evaluate the positive and negative emotions expressed in the transcripts with the help of polarity detection and emotional labels.

Error! Reference source not found. Sample output of sentiment analysis illustrating positive or negative sentiments and corresponding probability

```
Selected Rows from Table OUTSENT
_document_    _sentiment_
_probability_
          1      Positive
0.7890
          2      Positive
0.8842
          3      Positive
0.5670
          4      Positive
0.7891
```

```
          5      Positive
0.6912
```

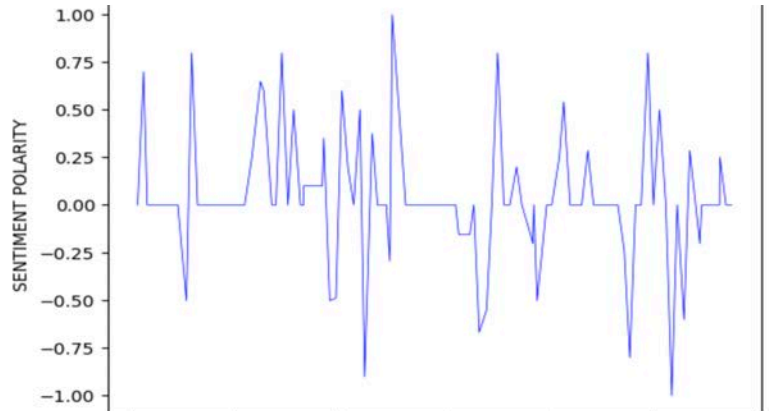
Error! Reference source not found.2 Sample output of sentiment analysis illustrating words that correspond to positive sentiments

```
# Units
model.wv.most_similar(positive=['terabyte', 'channels', 'channel', 'charge', 'credit'])
```

```
[('gives', 0.9903610944747925),
 ('card', 0.987489640712738),
 ('cost', 0.9804607629776001),
 ('promotions', 0.9798007011413574),
 ('packages', 0.9786516427993774),
 ('level', 0.9785544276237488),
 ('secondary', 0.9771579504013062),
 ('save', 0.976134181022644),
 ('without', 0.9754191637039185),
 ('prorated', 0.9751393795013428)]
```

Sentiment path analysis can also be performed from transcripts in chronological order. This tracks the sentiment polarity during the call that can consolidate extreme sentiments to better handle calls in future. SAS Visual studio can be used to perform sentiment path analysis.

Error! Reference source not found.3 Sample output of sentiment polarity illustrating magnitude of positive and negative sentiments



TOPIC MODELING

It is vital to extract abstracts from each customer conversation and consolidate it for Agent via API to serve faster and efficiently. Mallet's Implementation LDA from gensim is used to extract the topics as it runs faster and identifies relevant topics with high discrimination. The gensim allows evaluation of topic importance using a threshold which in turn allows filtering desired topics. SAS Visual studio has the capability to perform automatic topic discovery using unsupervised methods – SVD & LVD.

Topic modeling helps understand most common reasons why customers are calling the IVR. It helps identify repeat callers, most common issues encountered by customers, top performing agent traits, customer pain points, agent recommendations for future interaction etc. Filters have to be used prior to running LDA on pre-processed text.

Provided below is the code for both the LDA versions to improve and find the appropriate topics

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

```
# Download File: http://mallet.cs.umass.edu/dist/mallet-2.0.8.zip
mallet_path = str(Path(data_dir).parents[0] / 'mallet-2.0.8/bin/mallet') # update this path
ldamallet = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, num_topics=11, id2word=id
2word)
```

```
# Show Topics
pprint(ldamallet.show_topics(formatted=False))

# Compute Coherence Score
coherence_model_ldamallet = CoherenceModel(model=ldamallet, texts=data_lemmatized, dictionary=id2w
ord, coherence='c_v')
coherence_ldamallet = coherence_model_ldamallet.get_coherence()
print('\nCoherence Score: ', coherence_ldamallet)
```

Topic selection is usually very subjective and difficult. However, LDA provides coherence value metric to choose number of relevant topics. Topics with highest coherence value can be identified.

Error! Reference source not found.4 Coherence value by topics

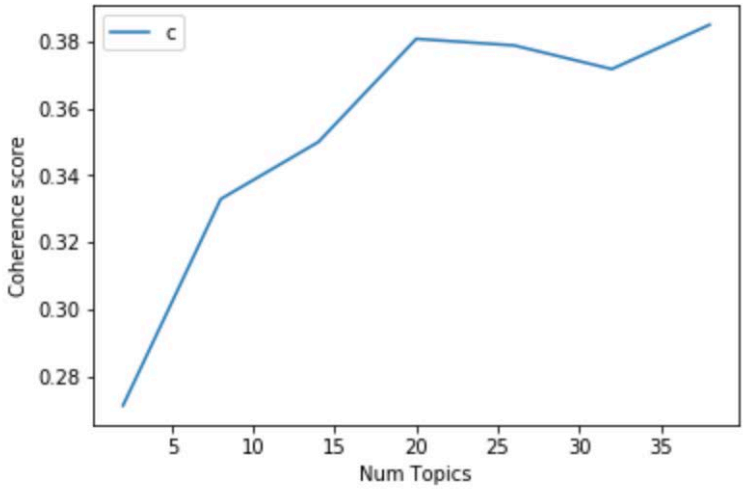


Table 1 Illustrates sample topics identified

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Documents
0	6.0	sir, account, refund, give, hold, dollar, pull, ill, email_address, email	27.0	0.1120
1	1.0	go_ahead, check, end, correct, ill, issue, one_moment, good, moment, alright	15.0	0.0622
2	7.0	card, account, service, number, phone_number, online, change, give, call, pin	7.0	0.0290
3	18.0	modem, work, connect, back, technician, check, internet, light, mister, service	8.0	0.0332
4	1.0	go_ahead, check, end, correct, ill, issue, one_moment, good, moment, alright	6.0	0.0249
5	17.0	dollar, bill, charge, month, line, buy, switch, mobile, free, forty_five	7.0	0.0290
6	4.0	connect, device, box, purchase, work, internet, signal, tv, hour, sign	13.0	0.0539

ELMO VECTOR FEATURE ENGINEERING

80% of customer data is unstructured. Businesses want to understand every aspect of customer relationship and it can be fulfilled only when we are able to identify the patterns and extract the qualitative data. Transforming the non-searchable content like video, audio, corpus text into meaningful structured patterns needs to be performed that can be consumed by Analytical DataMart's to better asses and predict customer needs. IVR qualitative features actually improve the predictive power of customer centric models by 5-10 basis points. Elmo extraction has been utilized on the corpus text to extract features and these features are fed to Predictive operations.

```
elmo = hub.Module("https://tfhub.dev/google/elmo/2", trainable=True)
```

```
# Extracting ELMo features prototype
x = ["Its nice weather in Philadelphia"]
embeddings = elmo(x, signature="default", as_dict=True)["elmo"]
embeddings.shape
print(embeddings)
```

```
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
Tensor("module_apply_default_6/aggregation/mul_3:0", shape=(1, 5, 1024), dtype=float32)
```

```
def elmo_vectors(x):
      embeddings = elmo(x.tolist(), signature="default", as_dict=True)["elmo"]

      with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        sess.run(tf.tables_initializer())
        # return average of ELMo features
        return sess.run(tf.reduce_mean(embeddings,1))
```

```
list_df = [df[i:i+100] for i in range(0,df.shape[0],100)]
```

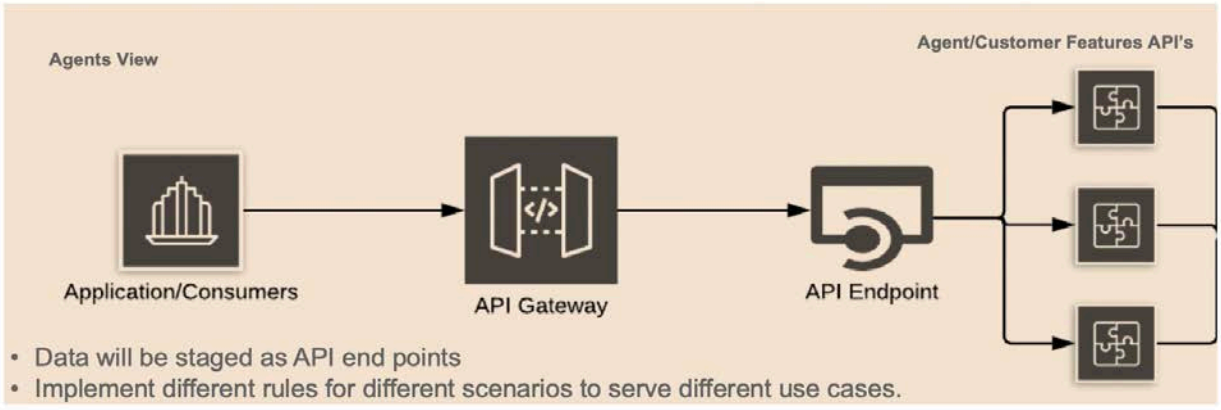
CUSTOMER AND AGENT FEATURES

The above text mining methods have allowed us to create the customer and agent features that provides the detailed view of call duration, interests, calling reasons, emotions, recommendations, opinions, agent recommendations, customer issues and pain points. These features from the above three methods are provided as microservice API as it allows us to bypass the native GUI and feed on the end Agent platforms with ease.

PHASE 3: API GATEWAY'S INTEGRATION FOR BOTH SAS REST API & OTHER API'S

SAS CAS integration with REST API's facilitates integration into many programming languages and external GUI's. An API wrapper was created for text mining outputs to balance the load and is feed into the Agent GUI through recommendation engine dashboard. Through this functionality, the agent has access to real-time overview of historical IVR performance and data metrics that helps them in serving the customer efficiently.

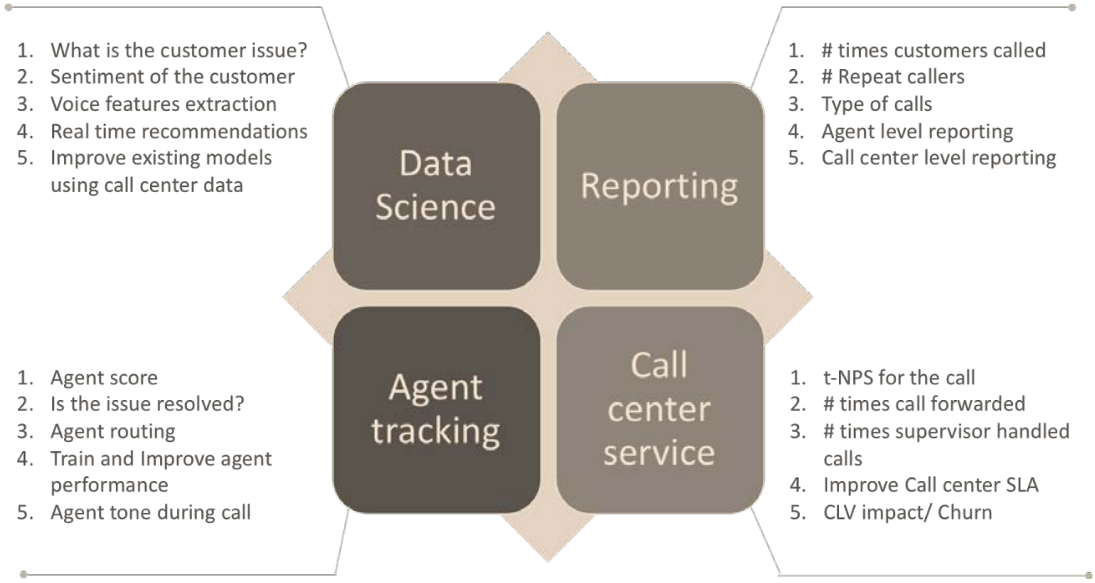
Display 3: API Gateway integration workflow



RECOMMENDATION ENGINE AND ITS USAGE

Real-time recommendation engine is structured to fit many scenarios that gauge and improves customer service. The architecture is so robust and distributed as it has extensive Multiprocessing parallel capabilities and adaptable for future changes. Below are the services that can be fulfilled with the above IVR Call Analytics assessment.

Display 4: IVR recommendation engine aids agents to make faster customer centric recommendations leveraging available insights that improves customer experience



CONCLUSION

We conclude that a composite real-time call analytics is necessary for businesses as research has shown, more than 70% of customer interactions are handled by call center agents. Real-time call analytics facilitates faster and efficient customer service, improved customer retention and enhanced agent engagement thereby enhancing customer experience. Companies aim to support from all perspectives, however the data at their disposal is 80% unstructured. Above demonstrations including numerous text mining components can be applied to unstructured data on a real-time basis allowing us to get insights into the data to make informed decisions for future and helps in building intelligent routing and mapping of agents to customers. The recommendation engine API's can help monetize agents efficiently and drive revenue and value to the company. Viya provides flexibility to integrate with external micro-service's API's which reduces cost and saves time in efficiently sharing analytical insights computed using SAS & open-source Analytical power. SAS Viya DLPy, Hadoop & open source integration, microservices API's, Visual text analytics applications are flexible enough to develop real-time text analytics engines. Our recommendation engine has the flexibility to constantly train and implement more use-cases with ease. Most call centers utilize through API gateways to establish a connection to call analytics micro-service's (Sentiments, Topics, Features, calling metrics, Agent metrics) through API end points to serve customer needs efficiently. Execution times can also be enhanced by making use of distributed CAS server (MPP) that is co-created with Hadoop integration.

Text Analytics for IVR calls is essential as it is one of the main elements to understand customer satisfaction and their perception about the services and experiences. Customer sentiments, topics and agent recommendations, customer issues & pain points can be supported to make efficient customer-centric strategies that shapes business. Personalization is the ultimate goal of any company to improve customer experience efficiently and effectively allowing competitiveness. Real-time recommendations through SAS Viya and open source call analytics will allow agents/recommendation engines to act faster and easier while tailoring a service to customer.

ACKNOWLEDGMENTS

We would like to thank SAS® Global Forum 2020 conference committee for their encouragement and opportunity to present our extensive work on Unified Call Analytics. We also thank Comcast corporation for supporting us.

RECOMMENDED READING

- *SAS VIYA CAS for BIG-Data Platforms*
- *SAS SWAT package – Load Audio, NLP – DeepRN, DLPy, Acoustic Neural network, n-gram language model*
- *SAS Visual Text Analytics– Comprehensive suite for Text Analytics besides to CAS third party Text programming*
- *CASL & Python executions on CAS*
- *Hadoop environment*
- *Open source – Elmo vector, Gensim, LDA & SVD, NLTK*
- *REST API & SAS Viya*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ranjit Jangam
Comcast Corporation
Ranjit_jangam@comcast.com

Head of Data Science supporting various business verticals at Comcast including Sales, Marketing and Risk driving profitable growth, risk mitigation leveraging Machine Learning/AI tools and framework. 13+ years of experience as a data driven innovator developing ML solutions that span end-to-end life cycle of a consumer that include driving profitable revenue, next best recommendations, retention strategies, improving customer experience & engagement, consumer segmentation for strategic decisioning. Prior experience includes leading data science capabilities with Fortune 500 companies including Farmers Insurance, Progressive Insurance, Plymouth Rock Assurance developing scalable machine learning models, products and solutions.

Anvesh Reddy Minukuri
Comcast Corporation
AnveshReddy_Minukuri@cable.comcast.com

Anvesh Reddy Minukuri is an AI/ML expert working for Comcast as a Lead Data Scientist. Prior to joining Comcast, he worked for an Indian Multinational Company as a Software Engineer. He pursued his Masters in OSU Data Mining Master's program from Oklahoma State University and Computer engineering from JNTU University, India. He joined Comcast in 2015 and is supporting EBI in diverse analytical projects. His areas include marketing models, recommendation engines to drive growth and product adoption, CLV, consumer and messaging segmentation, 360 customer analytics, customer persona and retention strategies. He has published papers at prior SAS global forums and various other analytical conferences.