

RingCentral[®] DEVELOPERS

Building Smarter Communications Using Artificial Intelligence

October 15, 2019

Table of Contents

Build Your RingCentral Virtual Voicemail Assistant for Business (Part 1)	3
Build Your RingCentral Virtual Voicemail Assistant for Business (Part 2)	15
Building Machine Learning Models with MonkeyLearn	30
Voice Communications Analytics	39



Learn more about RingCentral and earn great prizes and rewards in the process!

<https://gamechanging.dev>

Build Your RingCentral Virtual Voicemail Assistant for Business — Part 1



Paco Vu



Nowadays, consumers have a variety of options for obtaining services and getting the help they need. They can use web chat, email, the Internet and face-to-face contact, yet telephone customer service is still the first choice for most customers when they have questions or a problem that needs to be resolved.

In order to ensure your customers are happy with the customer service they receive, it's even more important for you to provide exceptional customer service, including outstanding telephone service. Consumers expect better service than ever before, and the capabilities of modern telephone communications allow you to offer them the satisfaction and resolution they demand.

IVR (Interactive Voice Response) can be a great tool for your company in decreasing customer wait time and increasing customer satisfaction, but customers are sometimes not overly fond of automatic response systems—especially when they have bad experiences. They don't want to go through a lengthy menu and after listening to a machine's voice and pressing keys, they still end up in a call queue—impatiently waiting for their turn to talk to a person; If they get dropped out of the queue for any reason, they have to repeat the lengthy process again and would end up at the tail of the queue.

Voicemail saves your customers time. If they cannot afford to wait on hold indefinitely or continue to try to call you throughout the day, voicemail gives them the opportunity to leave a message and let you return their call. Voicemail ensures accuracy because it is an exact recording of what the caller actually said. Using voicemail and taking messages properly are crucial to making sure your customers are happy. If they cannot talk to the appropriate person when they call, they are already less than satisfied; if their message is not relayed properly to the right individual, you will disappoint them even more. Listening to a voicemail before calling back your customer would also give you a chance to prepare for the conversation.

This article discusses the voicemail capabilities of the RingCentral cloud communications system, and AI (Artificial Intelligence) solutions that can be employed to build an effective virtual voicemail assistant for your telephone customer services.

Setup a voicemail inbox

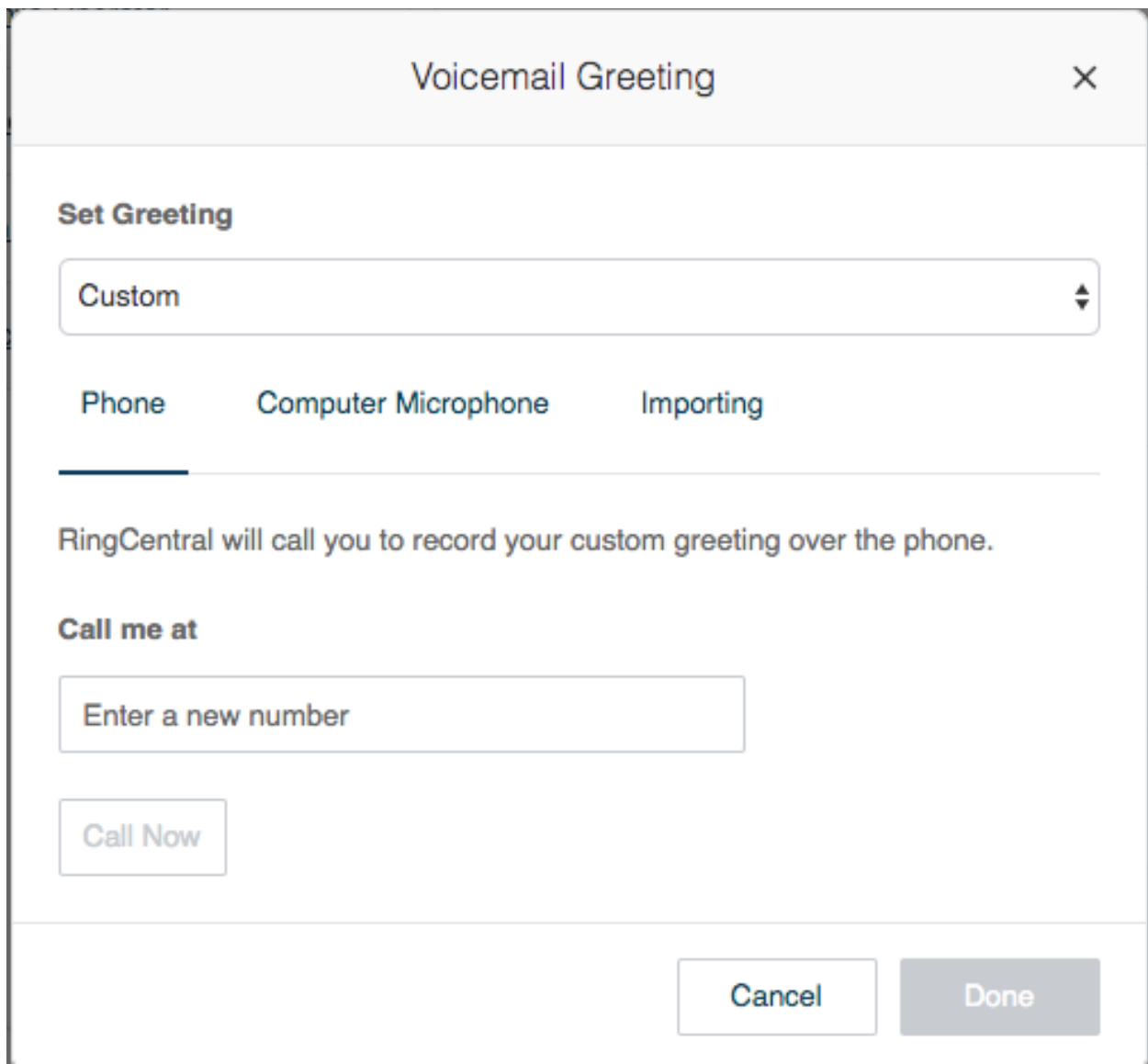
Under a RingCentral account, each extension (user) and department has its own free voicemail inbox with multiple options for managing voice messages. For the purpose of building a virtual voicemail assistant, you might want to create a new user named Voicemail Assistant, assign a direct phone number and set it up for a dedicated telephone customer service.

The screenshot displays the RingCentral Admin Portal. At the top, the RingCentral logo is on the left, and user information 'Paco' with a phone number '(312) 982-8160 Ext. 101' and an 'Admin Portal' dropdown are on the right. Below the header is a navigation bar with 'Phone System' (highlighted with a blue box), 'Users', 'Call Log', 'Billing', and 'Tools'. The main content area is divided into two sections. The left section is a sidebar with icons and labels for 'Company Info', 'Phone Numbers', 'Auto-Receptionist', '1 Group(s) 0 Other(s)', and 'Phones & Devices'. The right section contains three large circular icons with labels: 'Edit Business Hours', 'Edit Company Call Handling & Greetings', and 'Set Caller ID'. Below these is a 'Tutorials' section with a list of links: 'Change what callers hear', 'Set up call forwarding', 'Change company voicemail', 'Set up notification for calls, voicemails, and faxes', 'Use call queues', and 'Get more help'. Each link has a right-pointing arrow icon.

The voicemail greeting message is an important element of your business's voicemail system because it is often the first impression of your service that customers will have.

The default voicemail greeting message is adequate. But you should create a really good greeting message to impress customers by putting your best face on, while increasing the chances that you'll retain their business. With RingCentral, you have several options to customize your voicemail greeting message:

- Let the system call your phone number and you will have a chance to record a greeting message.
- Use the computer's microphone to record a greeting message.
- Upload a prerecorded audio file.



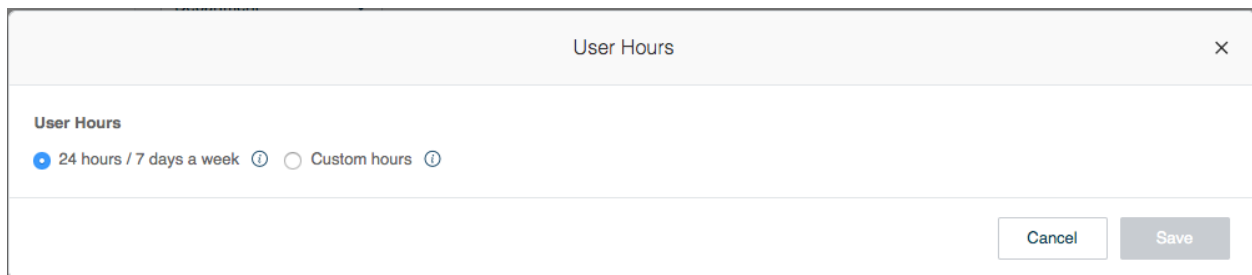
The screenshot shows a 'Voicemail Greeting' dialog box with a close button (X) in the top right corner. The main section is titled 'Set Greeting' and features a dropdown menu currently set to 'Custom'. Below the dropdown are three tabs: 'Phone', 'Computer Microphone', and 'Importing'. The 'Phone' tab is selected, indicated by a blue underline. Below the tabs, a message states: 'RingCentral will call you to record your custom greeting over the phone.' Underneath this message is a section titled 'Call me at' with a text input field containing the placeholder 'Enter a new number'. Below the input field is a 'Call Now' button. At the bottom right of the dialog are two buttons: 'Cancel' and 'Done'.

When your customers dial a number, they normally will hear several messages before reaching the voicemail inbox. This is because of the default system settings, which

enables the user greeting message, plays the ringtone while the phone is ringing, then plays the voicemail greeting message when the call is redirected to a voice mailbox.

Your virtual voicemail assistant is dedicated to a voicemail customer service, meaning that any incoming call to the service phone number should be redirected immediately to the voicemail inbox. So, you may want to cut down all the unnecessary steps including the ringing stage. To do so, you can open the “Screening, Greeting & Hold Music” section and disable the “User Greeting” and the “Connecting Message”. Then browse to the “Call Handling & Forwarding” section and turn off all the forwarding sequences so a call will be redirected immediately to the voice mailbox.

Last but not least, you can set the “User Hours” option to 24 hours and 7 days a week to make your virtual voicemail assistant work around the clock.



That’s pretty much all you need to setup for the voicemail inbox using the [RingCentral service dashboard](#). The next question is how do you access the voicemail programmatically to build your virtual voicemail assistant? The [RingCentral Developer platform](#) allows you to access to a growing number of APIs and integrate rich communication features into your application.

Get new voicemail notification

RingCentral platform supports real-time push notifications. It is an event triggered mechanism to get notified when something happens to the service you are observing. You can choose between [PubNub](#) or [Webhooks](#) delivery mechanism to get notifications. To get notified when there is a new voicemail arriving in a voicemail inbox, you can subscribe for the voicemail event for that particular user (inbox). What data comes together in a voicemail notification message? Below is a typical payload of a voicemail event:

```
{
  "uuid":"473993724243953516",
  "event":"/restapi/v1.0/account/~ /extension/~ /voicemail",
  "timestamp":"2019-06-16T21:27:06.525Z",
```

```

"subscriptionId":"c204cdb4-d8b0-4b3e-808a-xxxx",
"ownerId":"17800xxxx",
"body":{
  uri:
'https://api.ringcentral.com/restapi/v1.0/account/~extension/~
message-store/1054xxxx',
  id: 1054xxxx,
  to: [ {
    phoneNumber: '+1650513xxxx',
    name: '(650) 513-xxxx',
    location: 'San Mateo, CA'
  } ],
  from: {
    phoneNumber: '+1650224xxxx',
    name: 'Henry Taylor',
    location: 'Mountain View, CA'
  },
  type: 'VoiceMail',
  creationTime: '2019-06-16T22:17:27.000Z',
  readStatus: 'Unread',
  priority: 'Normal',
  attachments: [ {
    id: 1054xxxx,
    uri:
'https://media.ringcentral.com/restapi/v1.0/account/~extension/
~/message-store/1054xxxx/content/1054xxxx',
    type: 'AudioRecording',
    contentType: 'audio/mpeg',
    vmDuration: 11
  },{
    id: 2406yyyy,
    uri: 'https://media.ringcentral.com/restapi/v1.0/account/
~/extension/~message-store/1054xxxx/content/2406yyyy',
    type: 'AudioTranscription',
    contentType: 'text/plain',
    vmDuration: 11,

```

```
    fileName: 'transcription'  
  } ],  
  direction: 'Inbound',  
  availability: 'Alive',  
  messageStatus: 'Received',  
  lastModifiedTime: '2019-06-16T22:17:41.584Z',  
  vmTranscriptionStatus: 'Completed'  
}
```

There is some key information you will be interested in — the “from” phone number and the voicemail attachments.

Who just left a voicemail message?

Obviously, the “from” phone number is the caller’s phone number. This is the key information for you to identify the caller’s identity. You can search for the phone number from your customer database to check if the caller is your customer or not. If the caller is a known customer, you can pull necessary information on that customer from your database to get ready for the conversation with the customer when you make a call back. What if the caller’s identity is unrecognized? It can be an existing customer who uses a different phone number; it can be a potential customer; or it can be some spammer or scammer whom you really want to flag and ignore.

Scam call detection

Like email, voicemail is also another popular channel for scammers and spammers to exploit. Sometimes you can recognize a scam phone number if you’ve seen it several times earlier. But most of the time you’ll find out if a voicemail is spammy or not only after you start listening to it.

That makes spam voicemails extremely disturbing when you are overwhelmed with real customers’ voicemails. So, you don’t want to see spam voicemails in your voicemail list, or do you?

Your virtual voicemail assistant should be able to identify and flag such non-business voicemails for you. The fastest way to identify a spammy voicemail is to look up the caller’s phone number from a blacklist if you have one. If you don’t have a blacklist or cannot detect from your own blacklist, you can use online services such as the phone reputation detection from [WhitePages](#), to enquire for a detailed report of a phone

number.

The response from WhitePages's Phone Reputation service contains detailed reputation information of a phone number. You might be interested in a couple of key attributes—the reputation level and the category, that help you make quick decision if a voicemail is a spam or not.






The reputation level is a score number ranging from 1 to 4; with 1 indicates high confidence that the phone number has not been associated with spam/risky behavior; and 4 indicates high confidence that this is a spammy/risky number. You can use the reputation score to classify a caller's number as follow:

1: Clean; 2: Likely 3: Highly 4: Risky

The category label specifies the type of spam or scam associated with the phone number. Here is a list of categories the Phone Reputation API could identify:

Not Spam — Debt Collector — Telemarketer — Political Call — Phone Survey — Phishing — Extortion — IRS Scam — Tax Scam — Tech Support Scam — Vacation Scam — Lucky Winner Scam — Scam — NonProfit — Robocaller — TollFree Pumping — Other Spam

You can use the categories above to indicate the source of a voicemail. Optionally, for those voicemails came from your customers, you can specify a value of 1 to the reputation score and a label "Customer" for the category.

<input type="checkbox"/>	Dur	From	#Source	Spam
<input type="checkbox"/>	10	Paco Vu 	Customer	Clean
<input type="checkbox"/>	11	(202) 573-0012 	Extortion	Highly
<input type="checkbox"/>	17	(415) 775-4436 	Unknown	Clean
<input type="checkbox"/>	14	(508) 433-0816 	VacationScam	Risky
<input type="checkbox"/>	9	James Bond 	Customer	Clean

You can also detect a spammy voicemail by analyzing the voicemail content using AI technology. But let's explore that feature later when we discuss about AI solutions.

Access the voicemail audio

The voicemail audio content can be accessed via the attachment URI from the notification body.

```
{  
  id: 1054xxxx,  
  uri: 'https://media.ringcentral.com/restapi/v1.0/account/~/  
extension/~/message-store/1054xxxx/content/1054xxxx',  
  type: 'AudioRecording',  
  contentType: 'audio/mpeg',  
  vmDuration: 11  
}
```

Now that you'd get notified when there is a new voicemail; you'd know the caller's identity so you could avoid spending time on those spam voicemails; and you'd be able to listen to the voicemail audio. But your virtual voicemail assistant should do more to help you organize and prioritize your voicemail list. The challenge is how to make it recognize human speeches and understand the message for making decisions like a real assistant.

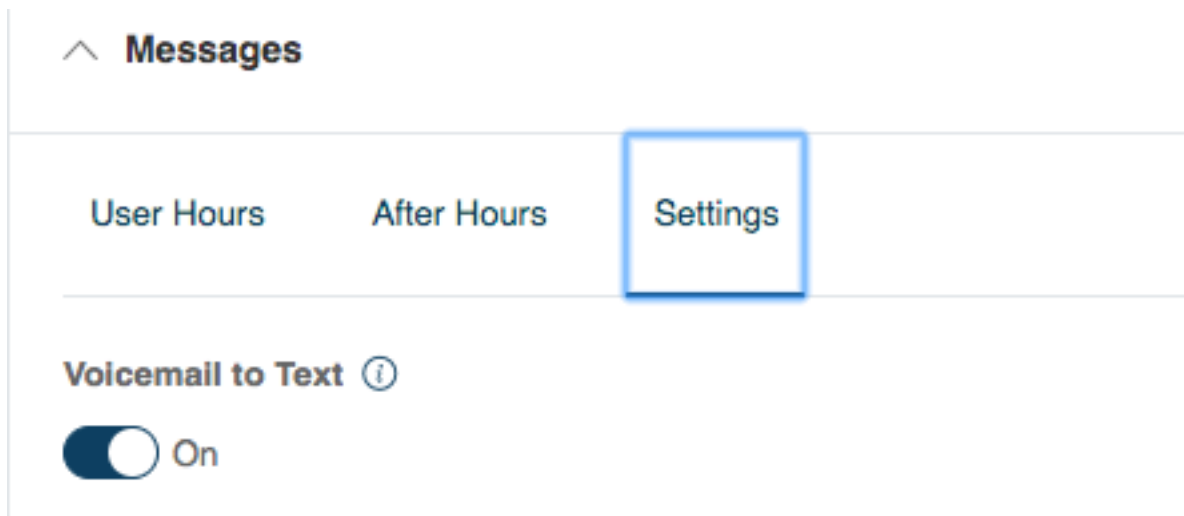
You are thinking about AI and machine learning right now, right? You have to develop NLP (Natural Language Processing) algorithms and statistical models to build the brain of your virtual assistant. It is a sophisticated project that normally it would take years to turn from technical concepts into an actual model in production.

Today, fortunately, you don't always need to develop your own machine learning algorithms for your AI solutions. Many companies provide AI services such as speech recognition, language detection, sentiment analysis, concept extraction etc. Most of the services are accessible on-demand via their cloud API platform. Your focus will be shifted from developing such complicated technology to creating great user experience and choosing the best AI solution available on the market for your application.

Now, let's move on to explore the AI services you would need for this project.

Voicemail transcription

Most of the RingCentral service plans include the Voicemail-to-Text feature. This means that all voicemails will be automatically transcribed. You can set the Voicemail to Text option from the RingCentral service dashboard mentioned earlier at the “Messages” section.



If the Voicemail to Text option is set, you can access the voicemail transcript via the attachment URI from the notification body.

```
{
  id: 2406yyyy,
  uri: 'https://media.ringcentral.com/restapi/v1.0/account/~/  
extension/~/message-store/1054xxxx/content/2406yyyy',
  type: 'AudioTranscription',
  contentType: 'text/plain',
  vmDuration: 11,
  fileName: 'transcription'
}
```

For testing on the sandbox environment, or for other reasons if you cannot use the transcription service from RingCentral (e.g. support other languages than English), you can use any Speech-to-Text service provided by many AI companies such as Google, AWS, IBM, Rev.ai, VoiceBase, AI Sense etc. to transcribe a voicemail.

The more accurate the voicemail transcript is, the better your virtual assistant would be able to process the customer’s message. Getting the voicemail transcript is just the first

step of preparation for the data analysis.

Define objectives

In customer services, sometimes the FIFS (First In First Serve) model is not preferable as you might want to give higher priority to a customer who needs urgent help. Assign a customer's case automatically to an agent based on competency and skills would also improve response time for your customer service, thus, increase your customer satisfaction.

So, your virtual voicemail assistant must be able to identify a voicemail urgency and to categorize the voicemail message then assign it to the right agent.

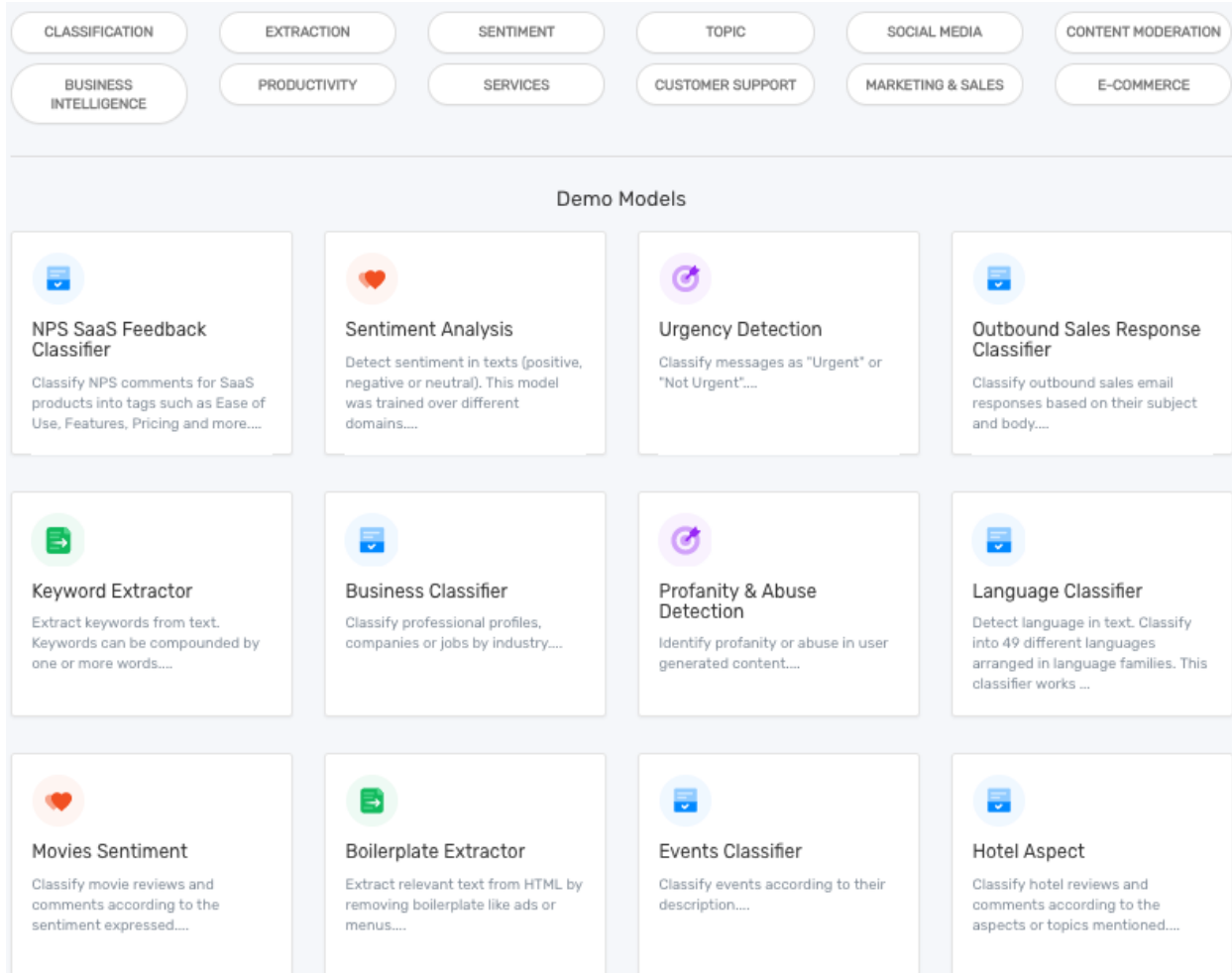
Train your dataset

Unfortunately, some generic AI services are not the perfect solutions for certain types of problems you are trying to solve. You can use general sentiment analysis to detect the common sense of customers satisfaction — how much they like or dislike something, but not necessarily about your product. You can also use the generic categorization from Google AI service to categorize your customer's problems. However, you will find it very hard to map their predefined categories into your own product or service categories.

Luckily, some leading AI companies let you customize the service to meet your expectation. In other words, they let you use their machine learning algorithms to train your own dataset.

To obtain a great data model that fits your customer service context, you must collect as many real sample messages as possible for your dataset, then train it with appropriate machine learning algorithms. Think of the dataset as a maths book with lots of addition and subtraction exercises. The more exercises you practice, the more knowledge about addition and subtraction you'll gain.

Different companies provide different methods to train custom dataset. Some allow you to choose algorithms, some automatically select the most suitable algorithms based on the type of model you selected. [MonkeyLearn](#) is one of the leading AI companies that provides easy ways to train custom dataset for different service models. They also provide lots of demo models for quick evaluations.



Auto reply with SMS message

To enhance the customers experience, you can instantly send a reply SMS message to notify the callers that you have received their message and will call them back as soon as you can. Of course, you are not going to reply to spammers nor try to send an SMS message if the caller's phone number is not a mobile phone number.

There are different ways to detect if the caller's number is a mobile number. If the phone number is found from your customer database, you can check the contact number type if that exists. Otherwise, you can use [WhitePages Phone Intelligence API](#) to detect if a number is a mobile phone number.

To send a reply SMS message, all you need is to compose a relevant message and send it using [RingCentral SMS API](#).

Try the virtual voicemail assistant demo

Click [here](#) to start the app and login with your own RingCentral user credentials. Open the Help page for detailed instructions.

Build Your RingCentral Virtual Voicemail Assistant for Business — Part 2

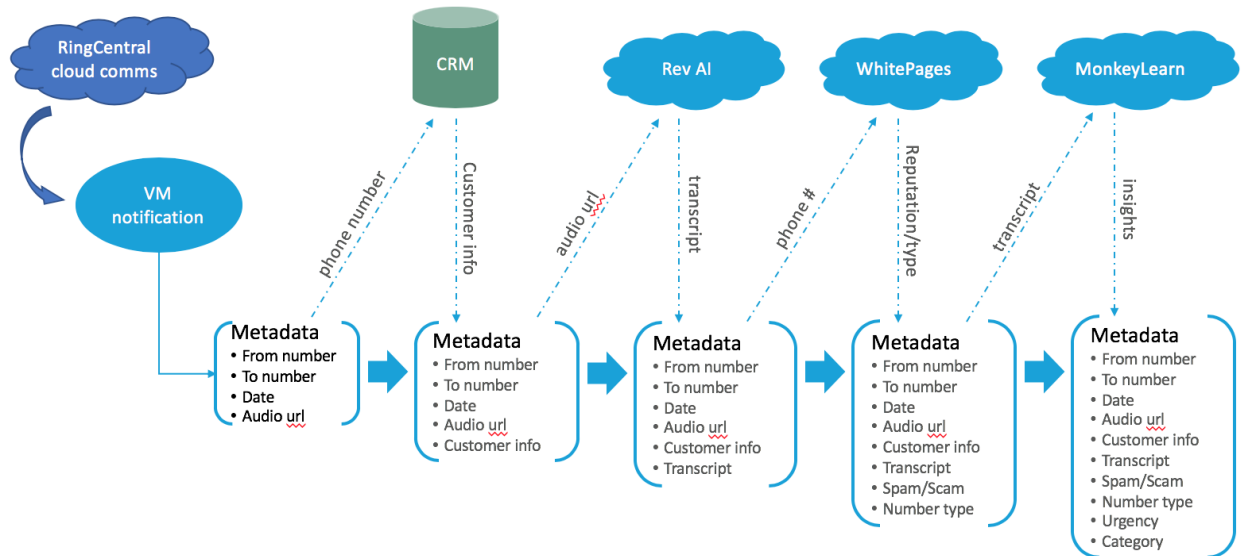


Paco Vu



In [part 1](#), I explained the voicemail capabilities of the RingCentral cloud communications system, and AI (Artificial Intelligence) solutions that can be employed to build an effective virtual voicemail assistant for your telephone customer services. I also showed you how to create and setup a dedicated extension for taking only voicemail messages, and the overall workflow of a virtual voicemail assistant.

In this article, I will walk through the essential steps to develop a Web app — a demo of virtual voicemail assistant for RingCentral Developers support, which can listen for new voicemail messages and perform the following tasks.



- Receive voicemail messages
- Detect spammy voicemails
- Auto reply to a caller with an SMS message
- Transcribe voicemail messages
- Detect urgency of a voicemail
- Categorize voicemail content
- Assign a support ticket (voicemail) to a designated support engineer (agent)
- Let an agent read the voicemail transcript or listen to the original voicemail message
- Allow agents to easily call back by click-to-dial or send an SMS to the caller

The associated demo application is built using Node JS, Express Web application framework. Thus, for conveniences, I will use the Node JS SDKs provided by RingCentral, Monkey Learn, and Rev AI to access their services. You can easily build this Web app backend in any programming language you like with the client libraries provided by the services providers mentioned above.

In order to build and run the demo app, you will need to setup the following accounts, get their API keys and login credentials:

- A RingCentral developer account. [Click here](#) to create a free developer account.
- A MonkeyLearn developer account. [Click here](#) to create.
- A Rev.ai developer account. [Click here](#) to create.
- A WhitePages Pro (a.k.a Ekata) developer account. [Click here](#) to create.

Note: The code snippets shown in this article are shorter and just for illustration. They may not work directly with copy/paste. I recommend you download the entire project [from here](#).

Get new voicemail notification

Let's get started with the critical function of a voicemail assistant — Listening for new voicemails arriving in the voice mailbox. This feature can be implemented using either [PubNub notification](#) or [Webhooks notification](#).

I will use the RingCentral webhooks notification method to subscribe for new voicemail event notification.

```
var eventFilters = ['/restapi/v1.0/account/~extension/~voicemail']
platform.post('/subscription',
{
  eventFilters: eventFilters,
  deliveryMode: {
    transportType: 'WebHook',
    address: 'https://[c1969441.ngrok.io]/webhookcallback'
  }
})
.then(function(response) {
  console.log("Ready to receive voicemail notification.")
})
```

See the **webhooks.js** module for complete code. I will skip the explanation on how RingCentral webhooks work, but if you are not familiar with RingCentral webhook notifications, please read [this blog](#) to learn more.

Since I expect to run the app on my local machine, I use the **ngrok** tool to get the callback address [https://c1969441.ngrok.io]. If the event notification subscription was subscribed successfully, I will receive notifications in a post request to the address specified above.

When I receive a notification, I parse the data to get the **body** JSON object, the **ownerId** (which is the extension Id of a user who subscribed for the notification) and the **subscriptionId**. Then I call the **processVoicemailNotification()** function to process the data. I use the **extensionId** to find the user who should process the voicemail notification and I use the **subscriptionId** to verify if it is the event I subscribed for.

```
app.post('/webhookcallback', function(req, res) {
  if(req.headers.hasOwnProperty("validation-token")) {
```

```

    res.setHeader('Validation-Token', req.headers['validation-
token']);
    res.statusCode = 200;
    res.end();
  }else{
    var data = []
    req.on('data', function(chunk) {
      data.push(chunk);
    })
    .on('end', function() {
      data = Buffer.concat(data).toString();
      var jsonObj = JSON.parse(data)
      var body = jsonObj.body
      var extensionId = jsonObj.ownerId
      var subscriptionId = jsonObj.subscriptionId

```

```

processVoicemailNotification(body,extensionId,subscriptionId)
    });
  }
})
function processVoicemailNotification(body, extId,
subscriptionId){
  var index = getUserIndexByExtensionId(extId)
  if (index < 0)
    return
  if (users[index].getSubscriptionId() == subscriptionId)
    users[index].processVoicemailNotification(body)
  else
    console.log("not my subscription")
}

```

Let's move on to parse the body of a voicemail notification to extract some essential data:


```

{
  "body": {
    id: 1054xxxx,
    from: {
      phoneNumber: '+1650224xxxx',
      location: 'Mountain View, CA'
    },
    type: 'VoiceMail',
    readStatus: 'Unread',
    attachments: [
      {
        id: 1054xxxx,
        uri:
'https://media.ringcentral.com/restapi/v1.0/account/~extension/
~/message-store/1054xxxx/content/1054xxxx',
        type: 'AudioRecording',
        contentType: 'audio/mpeg',
        vmDuration: 11
      }, {
        id: 2406yyyy,
        uri: 'https://media.ringcentral.com/restapi/v1.0/
account/~extension/~message-store/1054xxxx/content/2406yyyy',
        type: 'AudioTranscription',
        contentType: 'text/plain',
        vmDuration: 11,
        fileName: 'transcription'
      } ],
    lastModifiedTime: '2019-06-16T22:17:41.584Z',
    vmTranscriptionStatus: 'Completed'
  }
}

```

I extract the data and keep them in the **item** object as shown below:

```

var item = {}
if (body.from.hasOwnProperty('phoneNumber')){
  item['fromNumber'] = body.from.phoneNumber
  if (body.from.hasOwnProperty('name'))
    item['fromName'] = body.from.name
  else

```

```

        item['fromName'] = "Unknown"
    }else{
        // Just for demo purpose. Use predefined scam numbers
        if (index >= samplePhoneNumber.length)
            index = 0
        item['fromNumber'] = samplePhoneNumber[index]
        item['fromName'] = "Unknown"
        index++
    }
    item['toNumber'] = body.to[0].phoneNumber
    item['toName'] = body.to[0].name
    var timestamp = new Date(body.lastModifiedTime).getTime()
    item['date'] = timestamp
    item['id'] = body.id

```

I need to detect if the caller's phone number exists. This is necessary because the "from.phoneNumber" could be missing if a call is anonymous. In this demo, I predefined a list of scam phone numbers and use it for simulating scam calls if a call is anonymous. In a real application, we still can proceed the voicemail analysis for anonymous calls and try to detect if a caller leaves a call back number in the voicemail message.

Identify a caller

Using the caller's phone number, I make a query to my customer database to find a customer and retrieve the customer's information such as the first and last name, and the type of phone number (i.e. mobile). In a real application, you can connect to your CRM database and pull any necessary customer information that an agent should be aware of when he or she makes a callback to the customer.

```

var query = "SELECT first_name, last_name, phone_number_type
FROM customers WHERE phone_number='" + phoneNumber + "'"

```

For the demo purpose, I created a small customer database with just basic customer information below:

customer_id — first_name — last_name — phone_number — phone_number_type

If the caller is a registered customer, I move on to analyze the voicemail message. Otherwise, I will proceed to detect if it is a spammer or not.

Scam voicemail detection

As I mentioned earlier, I use WhitePages service to detect a phone number reputation.

```
var url = "https://proapi.whitepages.com/3.0/phone_reputation?"
url += "api_key=[WHITEPAGES_PHONE_REPUTATION_APIKEY]";
url += "&phone=[phoneNumber]"
request.get(url, function(err, res, body){
  if(res.statusCode == 200 ){
    // parse the body to get phone reputation info
    var jsonObj = JSON.parse(body)
    var numberInfo = {}
    if (jsonObj.hasOwnProperty("reputation_level"))
      numberInfo['reputation_level'] = jsonObj.reputation_level
    ...
    if (jsonObj.hasOwnProperty("reputation_details"))
      numberInfo['reputation_details'] =
jsonObj.reputation_details
    ...
  }
});
```

See the **number_analysis.js** module for complete code.

At this point, I should have enough information to identify if a caller is a spammer or not, based on the reputation level. I also classify a voicemail source using the “category” tag from the “reputation_details” returned from WhitePages. And for a registered customer’s voicemail, I use the “Customer” tag.

If the reputation level value is greater than 1, I flag the voicemail as spam with a label (Likely, Highly or Risky) based on the reputation level, and I will stop analyzing the voicemail.

If the reputation level value is 1, I will label the voicemail as “Clean”, and move on to get the voicemail transcript.

Get voicemail transcript

If the voicemail was automatically transcribed by RingCentral, I can access the voicemail transcript via the attachment URI from the notification event body. However, I need to check whether the voicemail transcription status is completed and if the attachment type is "AudioTranscription" before getting the transcript.

```
if (body.vmTranscriptionStatus == "Completed"){
  for (var attachment of body.attachments){
    if (attachment.type == "AudioTranscription"){
      platform.get(attachment.uri)
        .then(function(res) {
          return res.response().buffer();
        })
        .then(function(buffer) {
          var transcript = buffer.toString()
          item['transcript'] = transcript
        })
      break
    }
  }
}
```

For testing on the sandbox environment, I use the [Rev.ai speech recognition](#) service to transcribe the voicemail. I use [my own SDK](#) to call the Rev AI transcription service. However, you can use Rev AI's official [SDKs](#) to access their service. Either way, we must attach a valid access token to the voicemail attachment URL to make it publicly accessible (as long as the access token is valid).

```
for (var attachment of body.attachments){
  if (attachment.type == "AudioRecording"){
    var vmUri = platform.createUrl(attachment.uri, {addToken:
true})
    // Call Rev.ai transcription service API
    transcriptionist.transcribe(vmUri, function(err, transcript)
{
  if (err){
    console.log(err)
  }
})
}
```

```

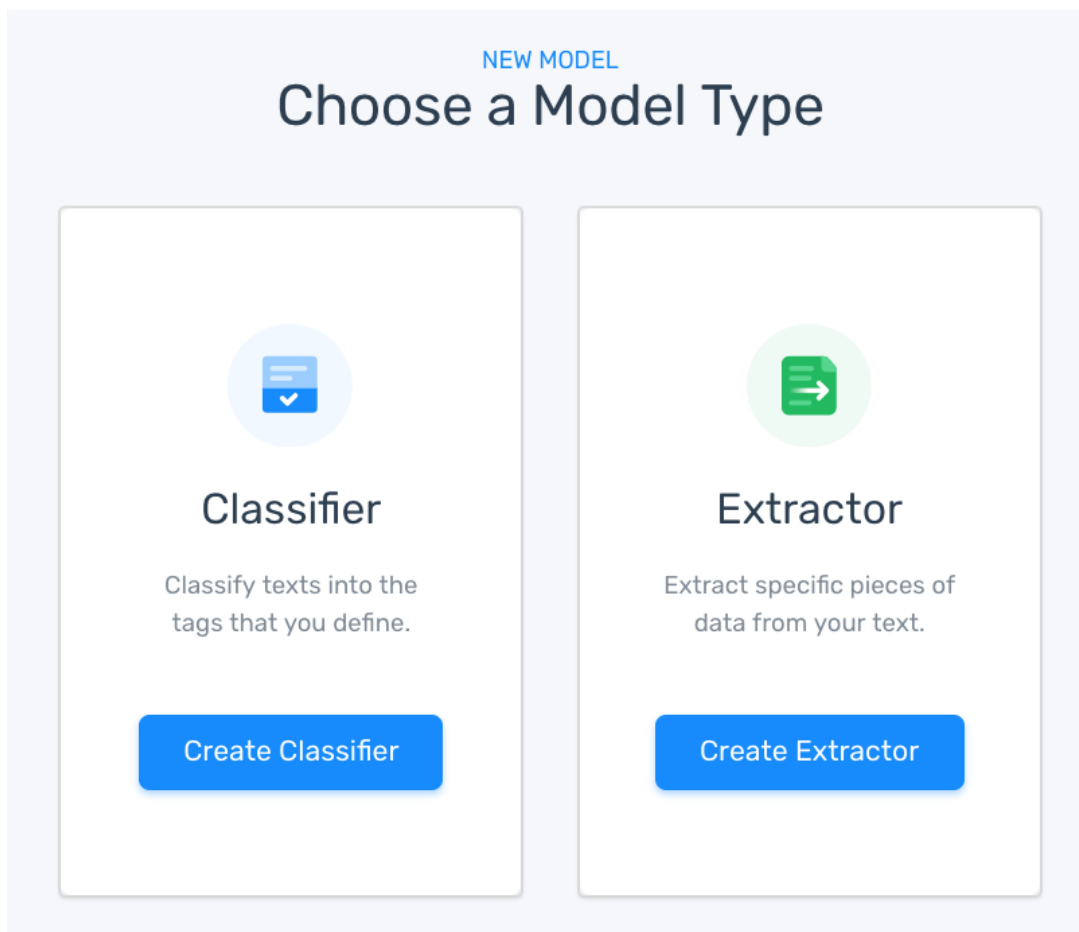
    }else{
        item['transcript'] = transcript
        ...
    }
    break
}
}

```

See the **transcription_engine.js** module for complete code.

Getting ready for voicemail analysis

I decided to use MonkeyLearn AI services for urgency detection and categorization. It is very easy to train your own dataset with MonkeyLearn's data model creation tools.



For urgency detection, I used their demo data model for the demo purpose, as it can detect the urgency based on keywords like “as soon as possible”, “as soon as you can” “this is urgent” etc.

For categorization, I trained my own dataset to build a data model for detecting the following predefined categories:

Messaging — Voice — Meeting — Data — Authentication — Configuration — Notification — Integration

First, I collected a couple hundred technical questions from the [RingCentral Developers Forum](#) and put them in an Excel sheet and tagged them appropriately, then saved them in a .CSV file. It's worth noting that, the more data samples you provide, the better your results will be for categorization.

	A	B
1	Text	Tag
	Meetings in sandbox	
	Is the meetings feature available on sandbox accounts? I created a developer account to test the API integration. I don't see meetings under the Tools menu. I can't apply for the	
2	Production until all the requirements are met, and I haven't been able to make a successful call to the meetings endpoint because the user doesn't have the proper permissions.	Meeting
	Can we play RC call recording?	
	Is it possible to playback a RC call recording in a web application.	
	Suppose we built a web application and want to play the call recording within the web pages directly and I don't want to keep the recording hosted in my application server. Can I	
3	play the recording from my web pages?	Data
	Finding missed call in call log api	
	I was going through call log api and I was figuring out how to get missed call value in the API.	
4	How can I find from the response if a call was a miss call?	Data
	Verify different ways of Fax receipt	
	When I use ringcentral Fax api to send fax to a different number, what are the different ways to test and verify Fax receipt?	
5	How can I make sure the Fax was received on other end? I know one way is to check call logs, but really want to know if other ways are available to confirm?	Messaging

Then I uploaded the file to MonkeyLearn to start the training process.

PREVIEW

Select Texts

Select the columns with your texts. Multiple selected columns will be concatenated

☒ Discard first row

Advanced ☒

	Use as Text	Use as Tag
1	Text	Tag
2	Meetings in sandbox Is the meetings feature available on san...	Meeting
3	Can we play RC call recording? Is it possible to playback...	Data
4	Finding missed call in call log api I was going through call...	Data
5	Verify different ways of Fax receipt When I use ringcentral ...	Messaging
6		
7		
8		
9		
10		

Showing 10 of 161 rows

Continue

Below are the main steps to train a data model with MonkeyLearn:

- Choose a Model Type
- Select a type of classification
- Import sample data from a file
- Build the model
- Test the model
- Copy the model Id and use it in your code

```
const MonkeyLearn = require('monkeylearn')
const ml = new MonkeyLearn(process.env.MONKEYLEARN_APIKEY)
let urgency_model_id = 'cl_Aiu8dfYF'
let categorization_model_id = 'cl_zBbUZ6dU'
```

Analyze the voicemail message

Now I have the voicemail transcript and my own data model, the next task is to detect the urgency and categorize the voicemail. But before analyzing the voicemail content, I do a quick check to see if I have enough information from the voicemail for analyzing it. I decided to check the word count from the voicemail transcript and set the threshold at 10 words. I also make sure that the voicemail is not a spammy one.

```
var wordArr = transcript.split(" ")
if (wordArr.length > 10 && reputation_level == 1){
  analyzeVoicemail(transcript)
}
```

To detect the urgency, I call the MonkeyLearn classifiers API, passing the “urgency_model_id” and the “transcript”, then I parse the API response to extract the urgency status (“Urgent”, “NotUrgent”), convert the confidence scale from 1 to 10.

```
let data = [transcript]
ml.classifiers.classify(urgency_model_id, data).then(res => {
  var body = res.body[0]
  var result = {}
  if (body.error == false){
    var classification = body.classifications[0]
    result['status'] = classification.tag_name
    var scaled = (classification.confidence * 10)
```



```

    if (classification.tag_name == "Urgent"){
      result['confidence'] = Math.ceil((scaled / 2) + 5)
    }else{
      result['confidence'] = Math.ceil(scaled / 2)
    }
  }else{
    console.log("Error: " + JSON.stringify(body))
  }
})

```

To categorize the voicemail, I call the MonkeyLearn classifiers API, passing the “categorization_model_id” and the “transcript”, then I parse the API response to read the categories. There could be multiple categories, and each category is associated with a confidence score. I iterate through the “classifications” array and pick the category with the highest confidence score.

```

let data = [transcript]
ml.classifiers.classify(categorization_model_id, data).then(res
=> {
  var body = res.body[0]
  var result = null
  if (body.error == false){
    var confidence_score = 0
    for (var item of body.classifications){
      if (item.confidence > confidence_score){
        result = {
          category: item.tag_name,
          confidence: item.confidence
        }
        confidence_score = item.confidence
      }
    }
  }else{
    console.log("Error: " + JSON.stringify(body))
  }
})

```

See the **content_analysis.js** module for complete code.

Assign support ticket to a designated support engineer

Suppose there are several support engineers (agents) in a developer support team, I assign their duty based on their technical skills that match the categories I defined above (the assignment can be done from the settings page of this demo app).

Delegation

Agents

Colin Arlington ▼

Categories

Nothing selected ▼

Assign

Assigned agents

Colin Arlington: Authentication,Notification

Bill Williams: Voice,Messaging,Meeting

Brad Barclay: Configuration,Integration,Data

```
var agentName = "Unassigned"
for (agent of settings.assigned_agents){
  for (var cat of agent.category){
    if (cat == result.category){
      var table = "voicemail_" + agent.id
      agentName = agent.name
      item['assigned'] = agent.name
      addSupportCaseToAgentDB(table, item)
      if (item.status == "Urgent" && item.confidence > 6){
        var text = "You have an urgent callback request from "
        text += item['fromNumber'] + "\n"
        text += (item['transcript'].length < 150) ?
```

```

        item['transcript'] :
        item['transcript'].substr(0, 150)
        notifyAgentBySmsMessage(thisUser, agent.id, text)
    }else{
        console.log("Not urgent. No need to alert an agent")
    }
    break;
}
}
}
}

```

In the code block above, I iterate through the “assigned_agents” list, compare the voicemail category with each of the agent’s assigned category. If there is a match, I assign a support ticket to that agent then add the ticket information to a database. I also send an SMS to notify the agent if the voicemail is detected as urgent.

Auto reply SMS message to a customer

To enhance the customers experience, we can instantly send a reply SMS message to notify the callers that we are working on their questions and will call them back as soon as we can. Of course, we are not going to reply to scammers nor try to send an SMS message if the caller’s phone number is not a mobile phone number.

To send a reply SMS message, all I need is to make sure that the caller’s phone number is a mobile phone number, then compose a relevant message and send an SMS message using RingCentral SMS API.

```

var text = "Hi, thank you for your voice message! We will get
back to you as soon as possible. For your reference, here is
your case number 1234567890"
var params = {
    from: { 'phoneNumber': ourServicePhoneNumber },
    to: [{ 'phoneNumber': item['fromNumber'] }],
    text: text
}
platform.post('/account/~extension~/sms', params)
    .then(function (response) {

```

```

    console.log("sent SMS")
  })
  .catch(function(e){
    console.log(e.message)
  })
}

```

0:00 / 0:00

Virtual Assistant +1 (205) 206-7011 Help | Logout |

Delete

Search case number

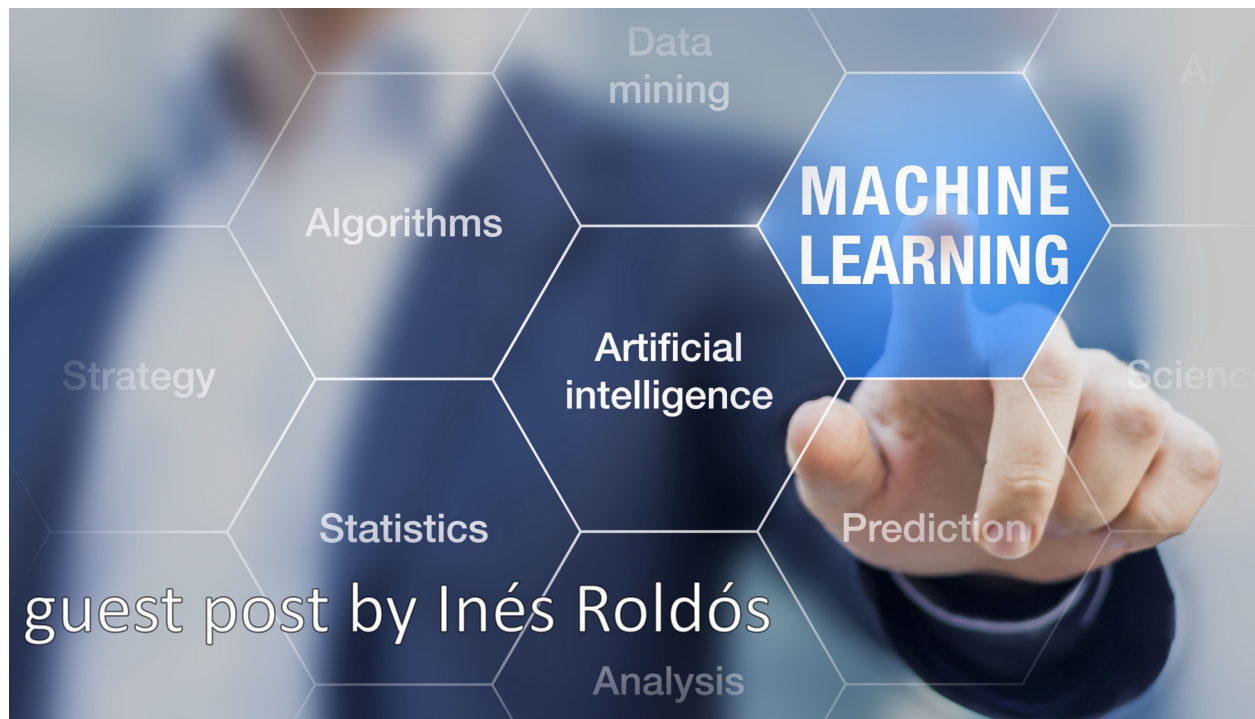
	Dur	From	#Source	Spam	Urgency	Date/Time	Age	Voice	Transcript	Category	Agent	Respond
	32	(929) 474-9128	Telemarketer	Risky	0	Aug 11, 2019, 10:12 AM	29d5h22m		N/A	N/A	Unassigned	
	13	Paco Vu	Customer	Clean	10	Aug 11, 2019, 08:39 AM	29d6h54m		Hello. I have a problem to access ring central platform today. I cannot get authenticated, so please call me back as soon as possible. Thank you very much.	Messaging	Colin Arlington	
	12	Paco Vu	Customer	Clean	10	Aug 6, 2019, 10:48 AM	34d4h46m		Hello. I have a problem with accessing ring central platform. It used to work before, but today I couldn't get it work, so please call me back as soon as you can. Thank you.	Authentication	Colin Arlington	
	24	Paco Vu	Customer	Clean	8	Aug 6, 2019, 08:15 AM	34d7h18m		Hello. I tried to use API to send SMS, MMS and fax messages. Um, but I cannot do that. Um, I don't know how to use the API. Um, I am a customer of RingCentral, so please call me back as soon as possible. This is very important for me. Thank you.	Messaging	Bill Williams	

That's it for the demo for now — you should be able to get started building your virtual voicemail assistant the way you want and feel free to further develop this app to make it useful for your business.

Building Machine Learning Models with MonkeyLearn



Paco Vu



Communication is an integral part of businesses, not only internally, but also externally, in how they communicate with the customers and partners. Consequently, it's essential to work with a communication system in place to achieve this successfully. Having the correct communication system will consequently create effective communication between employees, clients, and stakeholders, improving customer service and as a result, customer engagement.

However, with time and growth of the business comes new challenges. Customer queries start piling up and even having a successful communication system sometimes is not enough to manage the new flood of enquiries. Not only support teams need to handle this growth while delivering a quality service, but customers are becoming more and more demanding, and want answers right away. To illustrate, 80% of business buyers said they expect companies to respond to and interact with them in real-time.

But don't panic, artificial intelligence, most specifically machine learning is here to help. By using this technology, you're going to be able to automate certain processes so that

your customer service team can do more.

For example, you can use machine learning to build an effective virtual voicemail assistant for your telephone customer services. You can train a model to detect spammy voicemails, auto-reply to a caller with an SMS message, and even detect urgency of a voicemail, saving your customer service team countless of hours and making them more efficient.

After reading this article, you'll not only learn about what machine learning can do for your business, you'll also be able to build a machine learning model using MonkeyLearn. Go ahead and keep reading to learn how to do it — below are the sections if you want to go to something specific:

- Getting started with MonkeyLearn
- How to build an accurate model?
- How accurate a model can be?
- What algorithms are used for training models?
- Use cases and applications
- How MonkeyLearn handle data security?

Let's get started!

Getting Started with MonkeyLearn

MonkeyLearn is a platform that makes text analysis with machine learning easy and accessible for everyone, not only for data scientists. It's built to analyze huge amounts of data automatically and efficiently, saving businesses time and resources to do it manually.

With MonkeyLearn, you can use two types of models to analyze your data automatically: classifiers and extractors. On the one hand, Text classifiers are used to group data into a defined tag or category (by sentiment, topic, urgency, etc.). On the other hand, text extractors are used to identify and retrieve pieces of information present in text (for example keywords, entities, prices, dates, etc). By combining classifiers and extractors companies can automate processes, get insights from data, and save time processing data.

NEW MODEL

Choose a Model Type



Classifier

Classify texts into the tags that you define.

Create Classifier



Extractor

Extract specific pieces of data from your text.

Create Extractor

To illustrate how businesses are making use of MonkeyLearn, here are some of the most popular use cases:

- **Customer service:** automatically tag your support tickets based on topic, issue, sentiment, or intent. By doing this, you can automatically route the ticket to the right person, prioritize what to answer first, and improve reporting.
- **Customer feedback:** automatically tag feedback based on topic, aspect, intent or sentiment. This will allow you to analyze huge amounts of feedback, get key insights from data, and improve decision making.

Now you know what Machine Learning can do, but how does it work?

Machine learning algorithms learn by experience, so in order to perform a certain task, they need to be trained how to do it with data. For example, if you want a machine

learning model to detect spammy voicemails, you'll need to give examples of both 'regular' and 'spammy' voicemails to the machine. Once it has seen a certain amount of examples, the model will be able to effectively discern spam content from the regular content and begin making predictions on new voicemails.

How to Build an Accurate Machine Learning Model

If you want to create a custom model in MonkeyLearn, you'll have to train it to be able to perform its predictions. Here are some of the best practices to follow in order to train an accurate custom model:

1. The more data you use for training a model, the smarter the model will be

The amount of data you'll need to create an accurate model depends on each particular case. But as a general rule, the more training samples, the better. Machine learning algorithms learn from the data you feed, so naturally, the more information you give to the model, the smarter it will be.

For instance, to obtain accurate results in topic detection you'll need about 250 examples per category or tag, whereas in sentiment analysis you'll need around 500 examples per tag (e.g. positive, neutral, and negative).

In the case you want to create a model that detects spammy voicemail or a model that detects the urgency of it you'll need about 100 to 300 examples per tag to start seeing good results.

2. Quality of the data is more important than volume

Even though the quantity of the data is relevant, keep in mind that, in this case, the *quality* is even more important. It's preferable to feed the algorithm with less, but high-quality training samples, than feeding the model with thousands of examples that have no valuable information for the model.

For example, if you're creating a model to detect the urgency of voicemails, you should feed the machine with the different ways to express urgency by customers. If you only manage to train the model with examples that just mention things like 'ASAP' or 'Please Help Me Now', the model may miss other situations that you'll also consider urgent (for

example, a customer gently asking for a refund or reporting a bug).

3. Define tags that can be used consistently

Tags should always have a unique and specific definition. Define each tag with clear guidelines and make sure there are no overlapping concepts between two of them.

Tagging your data inconsistently for training your model causes confusion and significantly affect the accuracy of predictions.

4. Keep tags to a minimum. Remove tags that are too small or too niche

Aim to stick to a maximum of 10–15 tags per model. Having more tags than that will not only make the tagging more inconsistent over time, but also it will be more time-consuming to tag data for training the model.

As an example, if you're categorizing voicemails based on the topic their content, don't create niche tags like "mobile performance", "app speed", or "desktop loading times" that only apply for a small amount of voicemails. Instead create a single broader tag like "performance" that can group these kinds of voicemails. With niche tags, the machine won't be able to learn correctly, as there wouldn't be enough examples or information to learn from.

5. Use a single classification criterion per model

Create one classifier per challenge you want to solve. If you are tagging voicemails based on different criteria, just separate them into two different custom fields.

For example, if you want to tag voicemails based on their sentiment (positive, negative, or neutral) and on their urgency (urgent, not urgent), make two different models for each task. Combining both tasks into a single model will confuse the model and affect its prediction capabilities.

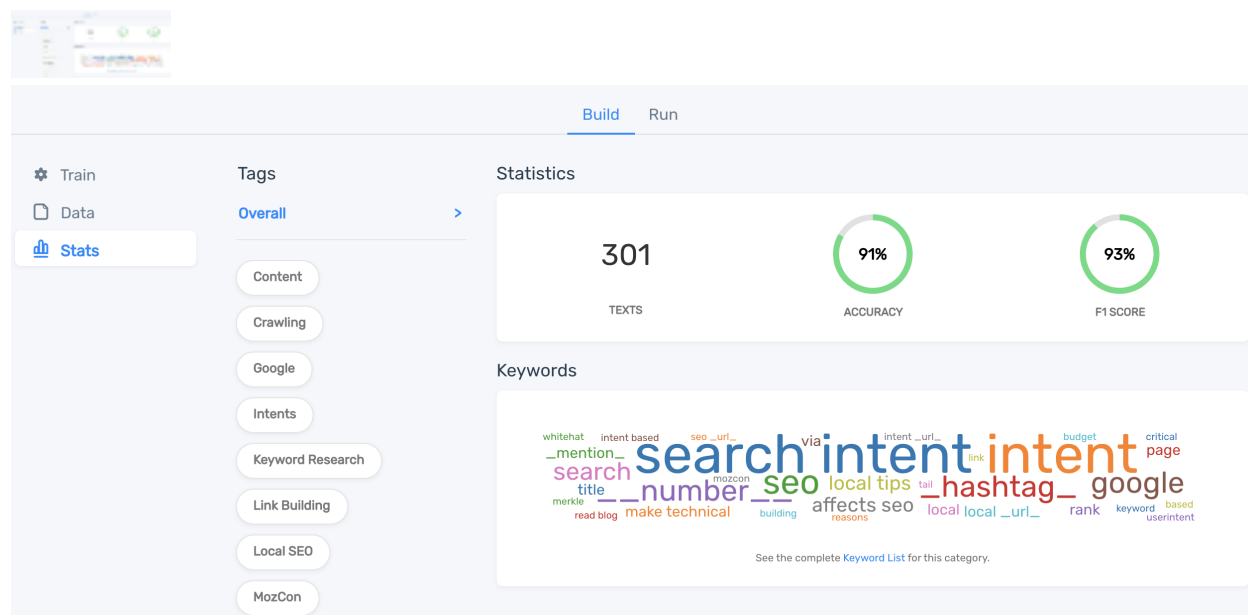
6. It's an iterative process

Creating a reliable and accurate machine learning model is an iterative process. You start with a small model that only 'understands' particular type of voicemails. Then, you add more examples to improve certain predictions. Afterward, you detect certain edge cases that the model makes mistakes and you work with the existing training data to improve these predictions. Next, you adjust the parameters and start fine-tuning the model for specific situations. And so on.

Keep in mind that a machine learning model can always be improved. You should continuously keep feeding the model with more and better examples to get the best results. If you just stick to the data you initially fed the machine, the learning process will end there and the model will not become more accurate or even learn from a new type of voicemail that you might receive over time.

How Accurate a Model Can Be?

If you follow these best practices, you're probably wondering exactly how accurate the model can become. Well... there is no simple answer to this question. It depends on each particular case, however, provided the dataset is clean, criteria is well-defined and the tagging of the data is consistent, you might get to F1 scores over 90%. The F1 score is the statistical accuracy of the model, so naturally the higher the F1 score, the better.



The accuracy of the model will also depend on which algorithm was used to create the model. In the following section, we'll go through the different options you'll have.

What Algorithms Are Used for Training the Models?

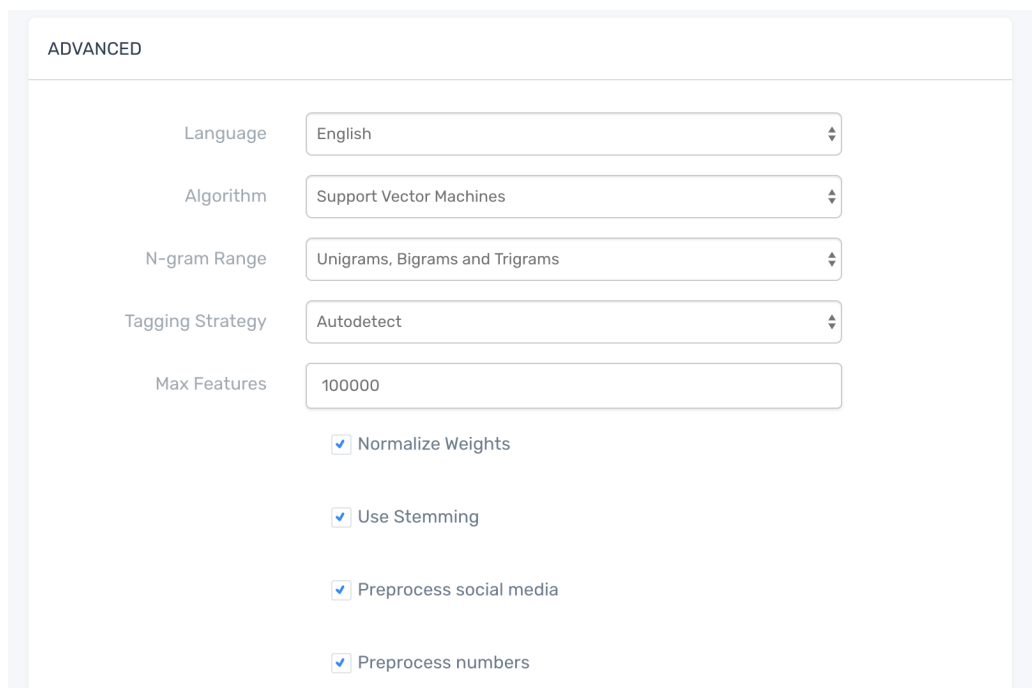
When creating a custom model in MonkeyLearn you'll be able to choose which algorithm to use to train the model. There is no right or wrong way to go here, each algorithm works better in different situations. Depending on whether you're creating a classifier or extraction model, your options will be the following.

For custom classifiers, you can choose between training your model with Naive Bayes or Support Vector Machines (SVM) algorithms.

Naive Bayes is a simple, fast, accurate, and reliable solution, that works especially well with natural language processing problems. Naive Bayes takes advantage of Bayes' Theorem and probability theory to predict the tag of a text. It is a family of probabilistic algorithms that for a given text (input) calculates the probability of each tag (output), and decides the outcome based on the highest probability.

Support Vector Machines is an algorithm that works particularly well with a limited number of data, being faster, and having better performance than other algorithms. We recommend using this algorithm when the data is linearly separable so you can quickly classify data.

For custom extractors, you can only use the default algorithm which is Conditional Random Fields (CRF), an algorithm with a statistical approach that contemplates the context and relationship to make predictions. This algorithm can create really complex patterns between words and data than a REGEX and has the ability to generalize from a small amount of information.



The image shows a screenshot of the 'ADVANCED' settings panel in MonkeyLearn. The panel is titled 'ADVANCED' and contains several configuration options for a custom classifier. The options are as follows:

- Language:** A dropdown menu set to 'English'.
- Algorithm:** A dropdown menu set to 'Support Vector Machines'.
- N-gram Range:** A dropdown menu set to 'Unigrams, Bigrams and Trigrams'.
- Tagging Strategy:** A dropdown menu set to 'Autodetect'.
- Max Features:** A text input field set to '100000'.
- Normalize Weights:** A checkbox that is checked.
- Use Stemming:** A checkbox that is checked.
- Preprocess social media:** A checkbox that is checked.
- Preprocess numbers:** A checkbox that is checked.

Use Cases & Applications

By now, you have learned how to train a model and picked up some best practices on what to do to get accurate predictions. But how can all of this be useful for your business?

Machine learning can help get you to get key insights from your data and automate all kinds of processes. For example, you can use machine learning to create a smart voicemail assistant that can do some of your work for you, such as:

- Automatically tag new voicemails based on topic, issue, sentiment, or intent.
- Routing the voicemails to the right team member.
- Detect the urgency of a voicemail, so you'll be able to prioritize, answering the most urgent ones first.
- Analyze your voicemail to discover insights on what people are talking about, which can be used as a resource for decision making.
- Create AI-based auto-responses and response suggestions for voicemails to save time when giving an answer.

How MonkeyLearn Handles Data Security?

At MonkeyLearn, we are aware that businesses are not only trusting us to add value to their data but also to keep it secure. We take extreme measures to maintain that security, including:

- **Physical security:** Our services are hosted in Microsoft Azure which keep state-of-the-art physical security, including 24x7x365 surveillance, environmental protections, and extensive secure access policies.
- **System security:** our servers run in recent Linux OS releases with long term support policies and are regularly updated. Our web servers communicate over HTTPS (TLS 1.2) to protect requests from eavesdrop and man-in-the-middle attacks. We use 2048 bit RSA SSL certificates, signed with SHA256. Our engineering team monitors and logs errors using top-notch tools like Datadog and Sentry. We also have strict privacy policies and a testing infrastructure to ensure data privacy within our applications. Moreover, we run and securely store daily data backups.
- **Operational security:** The development and maintenance of our platform is run by our highly trained engineering team. User data can only be accessed by a small number of authorized employees. Accessing user's accounts by MonkeyLearn employees is only allowed in exceptional cases, always with your

prior permission and for the purpose of resolving a specific issue only.

Wrap-up

Companies grow and evolve over time. As soon as the number of customer queries start to scale, and the expectations start to arise, you'll need to provide your customer service team with the right tools to stay on top of their workloads.

That's where machine learning can play a crucial role. By automating processes that are time-consuming, customer service teams can focus on the important things. Instead of wasting time routing voicemails or support tickets to the right team or monitoring incoming customer queries to detect urgent issues, agents can work on what's more important; solving issues and delighting customers.

Voice Communications Analytics



Paco Vu

Wouldn't it be useful to make communication content, in the form of audio, searchable for what is said in it, analyze it and extract actionable insights which help us quickly understand and easily navigate to critical moments within a conversation?

In this article, I will walk through the necessary steps to build a Web app that can analyze call recordings and voicemail content to extract text and actionable insights. It's about automating the process of deriving meaning from vast quantities of content, which would be impossible with purely human involvement — but possible by using artificial intelligence and applied machine learning technology available on the market like IBM Watson, Google Cloud Platform or Haven OnDemand platform. Once we're finished we'll have an app which will:

- Transcribe speech-to-text from call recordings and voicemail messages.
- Classify call content into predefined set of classification categories.
- Index call content with extracted metadata to enable advanced search.
- Allow users to search for any spoken word or phrases from call recordings and voicemail messages.
- Allow users to fetch call recordings and voicemail by a caller's number, a callee's number or by an extension number.
- Allow users to list call recordings and voicemail under the same categories.
- Allow users to search for call recording and voicemail with sentiment as positive, negative or neutral.
- Play back a call recording or a voicemail with stylish text synchronization.
- Allow users to interact with a call recording or a voicemail from the transcript by clicking on any word from the text to fast-forward or rewind to the selected word.
- Highlight positive and negative human opinions in the speech.
- Extract meaningful entities from a call recording or from a voicemail and allow users to see summaries of famous people, famous places and famous companies mentioned in the transcript. And easily navigate to related information from Wikipedia.
- Allow users easily reply to a voicemail message by click-to-dial.

This demo application is built using Node JS, Express Web application framework. Thus, for conveniences, I will use the Node JS SDKs provided by [RingCentral](#), [IBM Watson](#), [Google Cloud Platform](#) and [Haven OnDemand Platform](#) to access their services. You can easily build this Web app backend in any programming language you like with the client libraries provided by the services providers mentioned above.

Project's source code

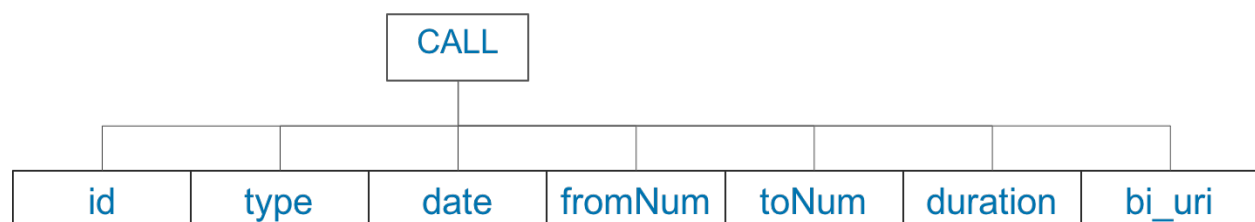
Code snippets in this blog are just for illustration purpose. They are shortened and incomplete. In order to follow the course of the application development, you may want to [download the entire project source code from our GitHub repository](#).

If you want to build your own app using the source code, remember to use your own service access credentials to access [RingCentral platform](#), [IBM Watson](#), [Google Cloud Platform](#) and [Haven OnDemand platform](#) and replace them from the `.env` configuration file.

Prepare for the content

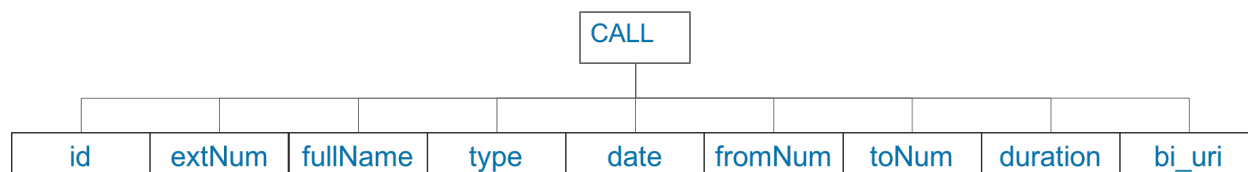
Before you start, make sure that you have some call recordings and voicemail content with good quality of speech. If you don't have real content, which is the best data to play with, you can make a few phone calls to RingCentral phone numbers under your RingCentral account, pick up the call and start to record the conversation. You should also make a few calls and just leave a voice message instead of picking up the call. By-the-way, if you are not a RingCentral customer, you can [sign up for a free developer account](#) and [download the RingCentral soft-phone app](#) for generating contents.

Now let's have a look at the RingCentral account's call log database, read call recording and voicemail logs, then collect a few essential call's metadata to create our own dataset. In this demo, I am interested at the date and time when the call was made, the duration of the call, the phone number or name of the caller, the phone number or name of the callee, and the most important piece of data is the URI of the binary content. Because I am interested at both call recordings and voicemail, I also want to specify the type of the content as 'CR' for call recording and 'VM' for voicemail.



To make it possible also for a supervisor to analyze company-wide voice communication, I allow a user with the *admin* role to access the company call log. This means that the supervisor can read call recordings and voicemail of any extension (any user) under the same account. That is why I want to add the extension number (extNum) and the full name (fullName) of any user to the dataset. Finally, I save the dataset into a local database (in this project I use SQLite).

Up to this point, we have some structured data with the metadata retrieved from RingCentral call log database. After saving the dataset to our SQLite database, we can search for calls made on a certain date, or calls from the same caller's number etc.



The most valuable information of a voice communication is still hidden in the audio binary content, which is the dialogue of a call recording or the monologue of a voicemail. The data in the wave form is human data — an unstructured data which could be understood only when we're listening to it. Now, without listening to every single call recording or voicemail, a time-consuming task which would take us hours or days to complete — how can we find out what was said in the conversation? How can we identify calls with happy or unhappy conversation? How do we know if our customers left a message complaining about our product or asking us to call them back?

Extract text from speech communication content

The mystery in a wave-form content could be uncovered by using some level of artificial intelligence. And the first necessary step to tackle is to transcribe the speech using Speech-to-Text technology. This sounds complicated and requires a lot of engineering works, right? Fortunately, speech recognition technology is matured and can be easily accessed via on-demand service from many different leading service providers such as IBM Watson, Google Cloud Platform, AWS etc.

There are pros and cons for considerations while choosing a speech recognition service from a wide range of providers on the market. For instance, in this demo, I chose Watson Speech-to-Text instead of Google Cloud Speech-to-Text just because Watson gives the transcript with the timestamp of each spoken word — which can be used to enable the feature of playing back a call recording or a voicemail with stylish text synchronization and letting users interact with the binary content. Otherwise, it would be great to use Google Cloud Speech-to-Text for its great features of providing transcript with punctuations, auto-detect language, which are quite critical for data analytics (please note that at the time this blog is written, I don't see Google Cloud speech-to-text API supports timestamp extraction but it may support that in the future).

To use Watson Speech-to-Text API, we specify the query parameters for our expecting result as follows:

```

var prams = {
  model: 'en-US_NarrowbandModel',
  audio: bufferStream,
  content_type: 'audio/mp3',
  timestamps: true,
  interim_results: false,
  profanity_filter: false,
  smart_formatting: true,
  speaker_labels: true
};

```

Where, **bufferStream** is the audio data stream read from the binary content URI from RingCentral call log. Then we call the API as shown below:

```

var watson = require('watson-developer-cloud');
var speechToText = new watson.SpeechToTextV1({
  username: process.env.WATSON_USERNAME,
  password: process.env.WATSON_PWD,
  url: 'https://stream.watsonplatform.net/speech-to-text/api/'
});
speechToText.recognize(params, function(err, res) {
  if (err)
    console.log(err);
  else
    console.log(JSON.stringify(res))
})

```

Upon receiving the response from Watson Speech-to-Text API, we parse the result and iterate thru the alternatives array to create an array to keep all the spoken words and the *start_time* timestamp of each word.

```

▼ results [22]
  ▼ 0 {2}
    ▼ alternatives [1]
      ▼ 0 {3}
        ▼ timestamps [2]
          ▼ 0 [3]
            0 : hello
            1 : 2.03
            2 : 2.55
          ▼ 1 [3]
            0 : then
            1 : 2.58
            2 : 2.92
          confidence : 0.736
          transcript : hello then
          final : true
        ▼ 1 {2}
          ▼ alternatives [1]

```

var wwoArr = [{"word": "hello", "offset": 2.03},
{"word": "then", "offset": 2.58},
{"word": "...", "offset": x.xx}]

Because we specified the **speaker_labels** parameter, Watson Speech-to-Text API result will contain an array of speaker labels. To create a conversation flow identified by the returned speaker labels from the response, we need to match a speaker label with the transcript. This is not straightforward as there is no word associated with a speaker label. Instead, we have to match the *start_time* timestamps from the **speaker_labels** array with the *start_time* timestamps from the **alternatives** arrays to create a new array containing spoken words of that speaker.

```

▼ results [22]
  ▼ 0 {2}
    ▼ alternatives [1]
      ▼ 0 {3}
        ▼ timestamps [2]
          ▼ 0 [3]
            0 : hello
            1 : 2.03
            2 : 2.55
          ▼ 1 [3]
            0 : then
            1 : 2.58
            2 : 2.92
          confidence : 0.736
          transcript : hello then
          final : true
        ▼ 1 {2}
          ▼ alternatives [1]

```

```

▼ speaker_labels [100]
  ▼ 0 {5}
    from : 2.03
    to : 2.55
    speaker : 2
    confidence : 0.469
    final : false
  ▼ 1 {5}
    from : 2.58
    to : 2.92
    speaker : 2
    confidence : 0.4
    final : false
  ▼ 2 {5}
    from : 3.67
    to : 4.34
    speaker : 1
    confidence : 0.445
    final : false

```

```

var conversations = [{"speakerid": 2, "words": ["hello", "then"], "timestamp": [2.03, 2.58],
{"speakerid": 1, "words": ["hi"], "timestamp": [3.67]}]

```

As Watson Speech-to-Text API does not support transcript with punctuations, we will need some mechanism to break a large chunk of text into sentences or paragraphs. In this demo project, for voicemail transcript, I simply rely on the transcript sentence of each **alternatives** in the **results** array. And for call recording transcript, I detect when a speaker id is changed to define a sentence. This approach is good enough as long as the speech is fluent and punctuated. For real application, I recommend you use some 3rd party service or better algorithm to create accurate punctuations for the transcript. Or perhaps, drop the stylish text synchronization feature and use Google Cloud Speech-to-Text service, which supports punctuations.

We've solved the first problem, a vital important step to transform audio content into text content which is required for data analytics. We also generated some new metadata such as the timestamp of every spoken word and the speaker labels. Our dataset is getting richer now with a few more useful data fields. In fact, we can save the dataset to our database and be able to search for any spoken word or phrase from the "audio" content. We can also implement a user interface to display the conversation separated by different speakers and display stylish text synchronized while playing back the audio content. I will discuss in more details about how to implement that feature later in this blog.



The next step is to apply data analytics for extracting actionable insights from the content and further transform unstructured data into structured data so that we can operate upon the data. This is a very critical step as you need to ask yourself a question of what information do you need? The answer will depend on the nature of the content and your expectation of operating the data. Let's say you are planning to analyze your customer's feedback about your products, you may want to use sentiment analysis to analyze how your customers think and talk about the product they purchased. What they like or dislike and what is the level of their opinions.

In this project, I want to categorize the content, extract keywords from it so that I can enhance the search engine — allow users to search for content with similar category and rank the search result. I also want to highlight meaningful entities such as people, places, companies or phone numbers if they are mentioned in a call recording or a voicemail message. One of my favorite data analytics features is to use sentiment analysis to measure human opinions and classify the content based on the polarized sentiments as positive or negative.

IBM Watson includes the [Natural Language Understanding API](#) which can be used to identify actionable insights from a document. The Watson NLU API is capable of extracting insights such as keywords with confidence score, meaningful entities, key concepts of the content etc. It can also classify the content into predefined categories. Alternatively, [Google Cloud Natural Language](#) or [Haven Ondemand Text Analysis](#) services could do the same thing. It's hard to say which platform is better in terms of quality, performance and price. You can always try them out and choose the one which works well for your data. In this project, I chose the Watson NLU for extracting keywords from the content and categorize the content. My choice is not based on the quality nor the pricing factor, but it is based on the performance in term of convenience because I can specify several features and make just one API call to get the result. While with Google Cloud Platform or Haven Ondemand Platform, I must make separate API calls for different features.

Let's call the Watson NLU API to extract keywords from the content and classify the content with predefined categories to enrich our dataset.

```
var params = {
  'text': text,
  'features': {
    'categories': {},
    'keywords': {
      'limit': 100
    }
  }
}

nlu.analyze(params, function(err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(response)
});
```

The response of a success API call above will contain an array of categories, which were classified for the provided content, and an array of extracted keywords with confidence score.

Alternatively, for categorization, you can use Google Cloud Natural Language API to classify the content.

```

const document = {
  content: text,
  type: 'PLAIN_TEXT',
};
language_client.classifyText({document: document})
  .then(results => {
    console.log(JSON.stringify(classification.categories))
  })
  .catch(err => {
    console.error('ERROR:', err);
  });

```

Remember that when classifying your content using Watson NLU or using Google Cloud Natural Language, your content will be classified based on their predefined set of categorizations. Thus, the results from each will be different!

Let's move on to analyze sentiments of the content. Both Watson NLU and Google Cloud Platform support sentiment analysis API. However, they are different and both give too simple result. For Watson sentiment analysis, you must define a set of target words (max 20 targets). This is suitable for content that you knew it might contain subjects that you want to analyze the sentiment. For instance, if the content is about customer's feedback of your products and you have a list of products' names, you can specify your products' names in the target array then add the **sentiment** keyword to the features list in the query parameter. Let's have a look at how it analyzes the following sample sentence:

```

var text = "The bananas were fresh and sweet. But the grapes
were rotten and bitter."
var params = {
  'text': text,
  'features': {
    'sentiment': {'target': ['bananas', 'grapes']}
  }
}
nlu.analyze(parameters, function(err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(JSON.stringify(response))
}

```

```
});
```

If sentiment is found relating to the specified targets, the result will be an array of targets with each object containing the sentiment information in the example response below:

```
{
  ...
  "sentiment": {
    "targets": [
      {
        "text": "bananas",
        "score": 0.740724,
        "label": "positive"
      }, {
        "text": "grapes",
        "score": -0.616188,
        "label": "negative"
      }
    ],
    "document": {
      "score": 0.0875428,
      "label": "positive"
    }
  },
  ...
}
```

If you want to use Google Cloud sentiment analysis API, you don't need to predefine a list of target words. The API analyzes sentiment of each sentence in the content. This is one of the reasons why I mentioned earlier that the recognized text should be accurately punctuated. If sentiment is found in sentences, the result will be an array of sentences with each object containing the data in the example response below:

```
"sentences": [
  {
    "text": {
      "content": "The bananas were fresh and sweet.",
      "beginOffset": -1
```

```

    },
    "sentiment": {
      "magnitude":0.8999999761581421,
      "score":0.8999999761581421
    }
  },{
    "text": {
      "content": "But the grapes were rotten and bitter.",
      "beginOffset": -1
    },
    "sentiment": {
      "magnitude": 0.20000000298023224,
      "score": -0.20000000298023224
    }
  }
]

```

You can compare the pros and cons of each API's capabilities and result, then decide which API you want to use. As for me, I need more than just the overall polarized sentiment of the content and the sentiment score of each predefined target if I use Watson NLU, or the sentiment score of a sentence if I use Google Cloud Sentiment Analysis API. That is why I am considering Haven On-demand platform because its [Sentiment Analysis API](#) result gives me more useful information. Let's have a look at how it analyzes the same sample sentence above:

```

var hod = require('havenondemand')
var hodClient = new hod.HODClient(process.env.HOD_APIKEY, "v2")
var request = {'text' : text}
hodClient.get('analyzesentiment', request, false,
  function(err, resp, body) {
    if (!err) {
      console.log(resp)
    }
  })
// RESPONSE
{
  "sentiment_analysis": [
    {

```



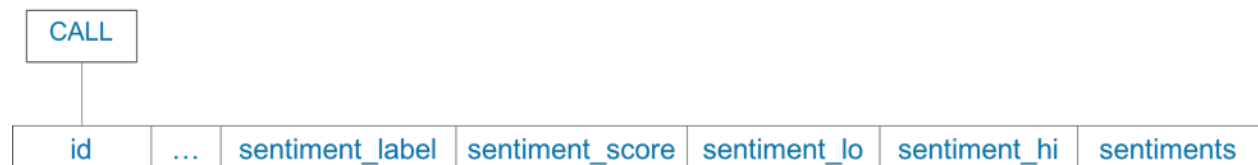
```

    "positive": [
      {
        "sentiment": "fresh and sweet",
        "topic": "The bananas",
        "score": 0.9203650635837769,
        "original_text": "The bananas were fresh and sweet",
        "original_length": 32,
        "normalized_text": "The bananas were fresh and sweet",
        "normalized_length": 32,
        "offset": 0
      }
    ],
    "negative": [
      {
        "sentiment": "rotten and bitter",
        "topic": "the grapes",
        "score": -0.8532963042204732,
        "original_text": "But the grapes were rotten and bitter",
        "original_length": 37,
        "normalized_text": "But the grapes were rotten and
bitter",
        "normalized_length": 37,
        "offset": 34
      }
    ],
    "aggregate": {
      "sentiment": "slightly positive",
      "score": 0.03353437968165185
    }
  }
]
}

```

As you can see, this API gives me much more insights. Besides the polarized scores, I can capture the topic and the sentiment in a sentence represented by the original text. Thus, I would easily find out more how people talks about a topic.

Let's consider what information we want to include in our dataset and how do we use them later. First, I want to add the aggregate sentiment label and the score to the dataset. With the sentiment label in the database, I can search for content with positive, negative or neutral sentiment. And with the sentiment score, I can set the threshold to limit search results or to rank the result based on the high or the low score. Second, I want to find out the highest positive sentiment score and the lowest negative sentiment score, compare them with predefined thresholds (one for positive and one for negative), add them to the dataset so that I can read and display alerts if there is any statement with very positive sentiment or some statement with very negative sentiment in the content. Finally, I want to add the positive and the negative sentiment objects which contain the sentiment, the topic, the score and the original text. With these detailed information, I can highlight positive and negative statements when displaying the text content.



Let's further extract meaningful entities from the content. All Watson NLU, Google Cloud Natural Language and Haven Ondemand Text Analysis support entities extraction feature. Like our previous consideration of choosing Sentiment Analysis API from different platforms, we should consider which one is more suitable for our use case. I will let you run your own tests with your real content and make your own judgement. For now, I choose Haven On-demand Entity Extraction API over the others because of its wide-range of entity types and providing reference to Wikipedia information source.

```

var entityType =
['people_eng', 'places_eng', 'companies_eng', 'number_phone_us']
var request = {
  'text': transcript,
  'entity_type': entityType,
  'show_alternatives': false
}
hodClient.get('extractentities', request, false,
function(err, response, body) {
  if (!err) {
    console.log(response)
  }
}
  
```

```
})
```

From the code snippet above, I specify the entity type to extract famous people, places, companies and U.S formatted phone numbers, then call the API to extract those entities from the transcript. On success, I will add the response containing a list of identified entities to my dataset. The detailed information from an entity differs from each type of entity. For example, a person entity object contains a quick profile of that person such as the name of that person, a list of professions, the date of birth, the image and the link to a person's Wikipedia page. And a place entity object contains essential information of that place such as the name of the location, the longitude and latitude, the type of the place (e.g. city or country) etcetera. You can learn more about entities information from [here](#).

Create actionable items

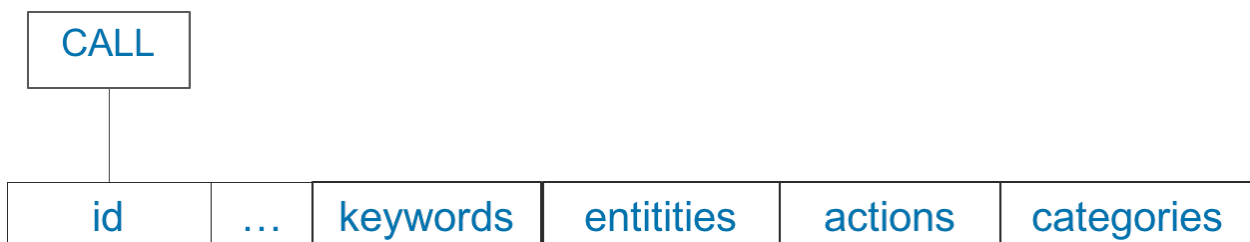
This demo project is about analyzing voice communication content, which includes voicemail messages. Some voicemail could contain a message just “for your information”, some voicemail could contain critical requests for taking actions. For example a voicemail might contain a “call back request”. Let's implement a simple technique to detect a call back request and make it easier for a user to reply such a voicemail.

```
var callActionDictionary = ['my number is', 'my cell phone is',  
'my cell number is', 'my phone number is', 'call me', 'call me  
back', 'give me a call', 'reach me at']  
var callHighlight = transcript  
for (var term of callActionDictionary){  
    var regExp = new RegExp("\\b" + term + "\\b", "ig");  
    if (callHighlight.match(regExp) != null){  
        var text = '<span class="call_highlight">  
        text += term + "</span>"  
        callHighlight = callHighlight.replace(regExp, text)  
    }  
}  
for (var number of phoneNumbers){  
    var regExp = new RegExp("\\b" + number + "\\b", "ig")  
    if (callHighlight.match(regExp) != null){  
        var call = '<a href="rcmobile://call?number=' + number + '">  
        call += number + '</a>'  
        callHighlight = callHighlight.replace(regExp, call)
```

```
}
}
```

First, I define a simple dictionary of call back request phrases. Then I detect if those phrases are found from the voicemail transcript. If a phrase is found, I highlight it by wrapping a CSS class `call_highlight` around the phrase. Then finally, I go through the list of phone numbers returned from the Entity Extraction API, and enable the click-to-dial on that number. The reason I implement the click-to-dial manually, is because I want to display the **callHighlight** text dynamically (using jQuery to show or hide) and force the browser to launch the RingCentral soft-phone to make a phone call. One extra thing to consider is that what if there is a call back request but there is no phone number detected? In that case, maybe we can presume that the caller expected a call back on the same number he/she was calling from. So in that case, we can use the “from number” extracted from the call metadata discussed earlier.

It's time to finalize our dataset with the rest of content metadata.



We are done with building the structured dataset and the process of metadata generation. Below is some code to create a SQLite database and a user table with defined columns. We use the unique **extensionId** of a user as the name of a user table.

```
function createUserTable(extensionId) {
  let db = new sqlite3.Database(USERS_DATABASE)
  var query = 'CREATE TABLE '+ extensionId + ' (id DOUBLE PRIMARY
KEY, rec_id VARCHAR(16) NOT NULL, date INT(11) NOT NULL, type
VARCHAR(12) NOT NULL, extensionNum VARCHAR(6) NOT NULL, fullName
VARCHAR(32) NOT NULL, fromRecipient VARCHAR(12) NOT NULL,
toRecipient VARCHAR(12) NOT NULL, recordingUrl VARCHAR(256) NOT
NULL, duration INT DEFAULT 0, processed BOOLEAN NOT NULL,
wordswithoffsets TEXT NOT NULL, transcript TEXT NOT NULL,
conversations TEXT NOT NULL, sentiment TEXT NOT NULL,
sentiment_label VARCHAR(8) NOT NULL, sentiment_score DOUBLE NOT
NULL, sentiment_score_hi DOUBLE NOT NULL, sentiment_score_low
```

```
DOUBLE NOT NULL, actions TEXT NOT NULL, keywords TEXT NOT NULL,
entities TEXT NOT NULL, categories TEXT NOT NULL)'
db.run(query, function(err, result) {
  if (err){
    console.error(err.message);
  }else{
    console.log("table created")
  }
});
}
```

Now let's learn how to read RingCentral call log database. I will skip the login to RingCentral account and get authenticated steps. But if you are interested, please read [this tutorial](#) or self-study the login part from the code of this project. RingCentral call log can be accessed programmatically using the call-log API. An account user with the *admin* role, can read the call log of any extension under the account. Any user with the standard *user* role can only read his own call log. This why after a user logs in his RingCentral account, I read the user information and detect his role to decide if he can read all extensions' call log or just his own extension call log.

```
platform.get('/account/~extension/~/')
  .then(function(response) {
    var jsonObj = response.json();
    if (jsonObj.permissions.admin.enabled){
      engine.getAccountExtensions(userIndex)
    }else{
      var item = {}
      var extensionList = []
      item['id'] = jsonObj.id
      item['extNum'] = jsonObj.extensionNumber.toString()
      item['fullName'] = jsonObj.contact.firstName + " "
      item['fullName'] += jsonObj.contact.lastName
      extensionList.push(item)
      ...
    }
  })
```

To read calls' information from the call log, I let a user choose a time range when calls were made. Then I iterate through the extension list to read the call log of each

extension and detect if there is a voicemail message or a call recording in that call, parse and extract metadata then add them to the database.

```
var endpoint = '/account/~extension/' + ext.id + '/call-log'
var params = {
  view: "Detailed",
  dateFrom: req.body.dateFrom,
  dateTo: req.body.dateTo,
  showBlocked: true,
  type: "Voice",
  perPage: 1000
}
platform.get(endpoint, params)
  .then(function(resp){
    var json = resp.json()
    if (json.records.length == 0){
      console.log("EMPTY")
    }else {
      let db = new sqlite3.Database(USERS_DATABASE);
      async.each(json.records,
        function(record, callback0){
          var item = {}
          if (record.hasOwnProperty("message") &&
            record.message.type == "VoiceMail"){
            // extract voice mail metadata
          }else if (record.hasOwnProperty("recording")){
            // extract call recording metadata
          }else {
            // call does not have CR nor Voicemail message
            return
          }
          var query = "INSERT into "+extId+" VALUES (...)"
          db.run(query, function(err, result) {
            if (err){
              console.error(err.message);
            }else{


```

```







        callback0(null, result)
    }
}
},
...

```

At this point, I might have call recordings and voicemails with metadata stored in the local database so I can read the data to display them on a dashboard as shown in the picture below:

Call Recordings ANALYSIS [Read](#) [List](#) [Logout](#) [About](#) Powered by 

Field: All Type: All Sentiment: All Pos: 0.5 Neg: -0.5 Total: 90 items

User	Call info	Type	Audio	Sentiment	Transcript	Analyze	Rem	Del
Ext.#: 103 Name: Anina	Fr: +165051 To: +14134183745 Date: Aug 23, 2018, 10:14 AM			--	<input type="button" value="Transcribe"/>		<input type="button" value="Rem"/>	<input type="button" value="Del"/>
Ext.#: 103 Name: Anina	Fr: +165051 To: +14134183745 Date: Aug 23, 2018, 10:11 AM			--	<input type="button" value="Transcribe"/>		<input type="button" value="Rem"/>	<input type="button" value="Del"/>
Ext.#: 103 Name: Anina	Fr: +165051 To: +14134183745 Date: Aug 10, 2018, 11:39 AM			--	<input type="button" value="Transcribe"/>		<input type="button" value="Rem"/>	<input type="button" value="Del"/>

Now I will let the user to manually click the **Transcribe** button at each item on the list to start the data analytics for that call recording or voicemail message. Of course you can automate this step if you want to. This means that after reading the call log, you can automatically call the function to transcribe the call recording or the voicemail binary content.

After the binary content is transcribed and analyzed, I add the new metadata to the database and update the content list with the transcript, positive or negative sentiment indicator and sentiment alerts. And I also enable the **Open** button so that the user can click to open and see detailed analytics of that call item.

Call Recordings ANALYSIS

[Read](#)
[List](#)
[Logout](#)
[About](#)

Powered by RingCentral

Field

All

*

Type

All

Sentiment

All

Pos: 0.5

Neg: -0.5

Search

Total: 27 items

User	Call info	Type	Audio	Sentiment	Transcript	Analyze	Rem	Del
Ext.#: 101 Name: Demo Recording	Fr: +165022- To: +16505 Date: Aug 16, 2018, 9:26 PM		0:00 / 14:12		the trying to access my online account for my Verizon hot spot online. I was unable to create a account without the mailing me a physical password receive the password a week later. want to go online ...	Open	Rem	Del
Ext.#: 101 Name: Demo Recording	Fr: +165022- To: +16505 Date: Aug 16, 2018, 9:26 PM		0:05 / 5:28		thank you for calling ABC telecommunications Sam my name is Kathy may I have your phone or account number please. I'm sorry can you please tell. well okay I'm sorry for that can you hear me now yes. I ...	Open	Rem	Del

From the search bar on the dashboard, I add several options such as the Field (all, transcript, keywords, from, to, extension or categories), the Type (call recording or voicemail) and the Sentiment (all, neutral, positive or negative) dropdown list. And I also add the positive and negative sliders for users to specify advanced search. For example, I can select the Transcript field and write the word “account” to the search text field, then select the type as **Voicemail** and the Sentiment as **Positive**. I also set the positive score slider to 0.600 and click the Search button. This will search from the database for voicemail messages which contain the word “account” and the message must have the overall positive sentiment with the sentiment score is greater than 0.600.

Now let’s have a look at the detailed analysis view. On the details view, I display the conversations on the left-hand side, and on the right-hand side I display the analytics information based on what the user chooses from the menu bar. In the screenshot shown below, you can see the left-hand side is shown the transcript with speaker labels, and the texts displayed in different colors. Texts in green color were read, and a single word in yellow color is the current spoken word, then texts in gray color are unread. The right-hand side is shown with the transcript highlighted with positive and negative sentiment in green or red color, respectively.


```

    aPlayer = document.getElementById("audio_player");
    aPlayer.addEventListener("timeupdate", seektimeupdate, false);
    aPlayer.addEventListener('loadeddata', audioLoaded, false);
    aPlayer.addEventListener('seeked', seekEnded, false);
}
function audioLoaded() {
    mIndex = 0;
}
function seekEnded() {
    var pos = aPlayer.currentTime;
    resetReadWords(pos);
    var id = "word" + mIndex;
    wordElm = document.getElementById(id);
}
function seektimeupdate() {
    var pos = aPlayer.currentTime;
    if (mIndex < wwoArr.length) {
        var check = wwoArr[mIndex].offset;
        while (pos >= check) {
            wordElm.setAttribute("class", "readtext");
            wordElm = document.getElementById("word"+mIndex);
            wordElm.setAttribute("class", "word");
            mIndex++;
            check = wwoArr[mIndex].offset;
        }
    }
}
function resetReadWords(value) {
    var elm;
    for (var i=0; i<mIndex; i++) {
        var idee = "word" + i;
        elm = document.getElementById(idee);
        elm.setAttribute("class", "unreadtext");
    }
    mIndex = 0;
    var pos = offsetArr[mIndex];

```

```

while (pos < value) {
    var idee = "word" + mIndex;
    elm = document.getElementById(idee);
    elm.setAttribute("class", "readtext");
    mIndex++;
    pos = offsetArr[mIndex];
}
}

```

Interact with the audio player and instant search function

You can search for any spoken word from the transcript by entering a word into the search text field and click the **Search** button. If the word is found from the transcript, the media player will fast forward or rewind instantly to that word and continue to play the audio content from that moment. You can also click on any word on the transcript to fast forward or fast rewind to that selected moment. Also from the right-hand side, you can read the sentiment and click on the **Goto** link to jump instantly to the beginning of that sentence in the transcript. To implement this feature, I add the **onclick** event to every word then assign the *jumpTo()* function and passing along the offset timestamp of that word. Inside the *jumpTo()* function, I simply set the offset timestamp to the audio player's *currentTime*. To search for a spoken word and fast forward or rewind to that moment, I pick the word from the search field and find it from the **wwowArr** (word with offset array), if the word is found from the array, I read the offset timestamp and call the *jumpTo()* function with the timestamp.

// EJS page

```

<% for (var n = 0; n < conv[i].sentence.length; n++) { %>
<% var wId = "word" + index %>
<span onclick= "jumpTo(<%= conv [i].timestamp[n] %>)"
class="unread" id="<%= wId %>" ><%= conv[i].sentence[n] %></
span>
<% index += 1 %>
<% } %>

```

// JavaScript code

```

function searchForText(){
    var searchWord = $("#search").val()

```

```

    for (var i=mIndex; i<wwoArr.length; i++){
        var word = wwoArr[i].word
        if (word == searchWord){
            var timeStamp = wwoArr[i].offset
            jumpTo(timeStamp)
            break
        }
    }
    if (i >= wwoArr.length){
        for (var i=0; i<wwoArr.length; i++){
            var word = wwoArr[i].word
            if (word == searchWord){
                var timeStamp = wwoArr[i].offset
                jumpTo(timeStamp)
                break
            }
        }
    }
}

function jumpTo(timeStamp) {
    var value = timeStamp;
    aPlayer.pause();
    resetReadWords(timeStamp);
    var id = "word" + mIndex;
    wordElm = document.getElementById(id);
    aPlayer.currentTime = timeStamp;
    aPlayer.play();
}

```

Displaying analytics results

From the menu bar on the right-hand side, you can choose to display sentiment analysis, meaningful entities, or transcript with keywords highlighted, or actionable item (click to dial in this demo) or just the plain text content.

You can change the positive and negative thresholds from the sliders to adjust the sentiment score for displaying sentiment analysis.

All the features above are implemented on the front-end using JavaScript to process the transcript and the metadata extracted from the audio content.

Congratulations! Now you should be able to build and further develop the this project with more features if you want to. For example, you may want to extract the concepts of the content using concepts extraction service. Or you want to implement an advanced feature for finding similar content based on the set of keywords found from each content.

Node.js / JavaScript: <https://github.com/ringcentral-tutorials/voice-communication-analytics-nodejs-demo>



Learn more about RingCentral and earn great prizes and rewards in the process!

<https://gamechanging.dev>

