IBM Demos Watson Hands-On Lab

Using Watson Speech to Text with Customization and Grammars

Abhishek Shrivastava
Digital Technical Engagement - Data Science and AI
abhia@ibm.com

Bijo Thomas
Digital Technical Engagement - Data Science and AI
bijo.thomas1@ibm.com

Jason Brown
Digital Technical Engagement - Data Science and AI
jasonbro@us.ibm.com

We Value Your Feedback

Your feedback is very important to us – we use it to continually improve the content and experience.

Please contact Bijo Thomas or Jason Brown if you have feedback or questions.

Bijo Thomas
bijo.thomas1@ibm.com

Jason Brown
jasonbro@us.ibm.com

Table of Contents

1 Introduction

Watson Speech to Text service is part of AI services available in IBM Cloud.

The service provides speech transcription capabilities for your applications. The service leverages machine learning to combine knowledge of grammar, language structure, and the composition of audio and voice signals to accurately transcribe the human voice. It continuously updates and refines its transcription as it receives more speech.

The service can be used in applications such as voice-automated chatbots, analytic tools for customer-service call centers, and multi-media transcription, among many others. The service is ideal for clients who need to extract high-quality speech transcripts from call center audio. Clients in industries such as financial services, healthcare, insurance, and telecommunication can develop cloud-native applications for customer care, customer voice, agent assistance, and other solutions.

1.1 Key Capabilities

Watson Speech to Text provides out of the box speech to text capability in several languages using audio in most common audio formats. Audio files are broadly classified as Broadband and Narrowband. Audio that are sampled at 16 kHz or higher are supported using Broadband models and audio that are sampled at 8 kHz are supported using Narrowband models. Check out our documentation to see our latest list of supported languages and model types.

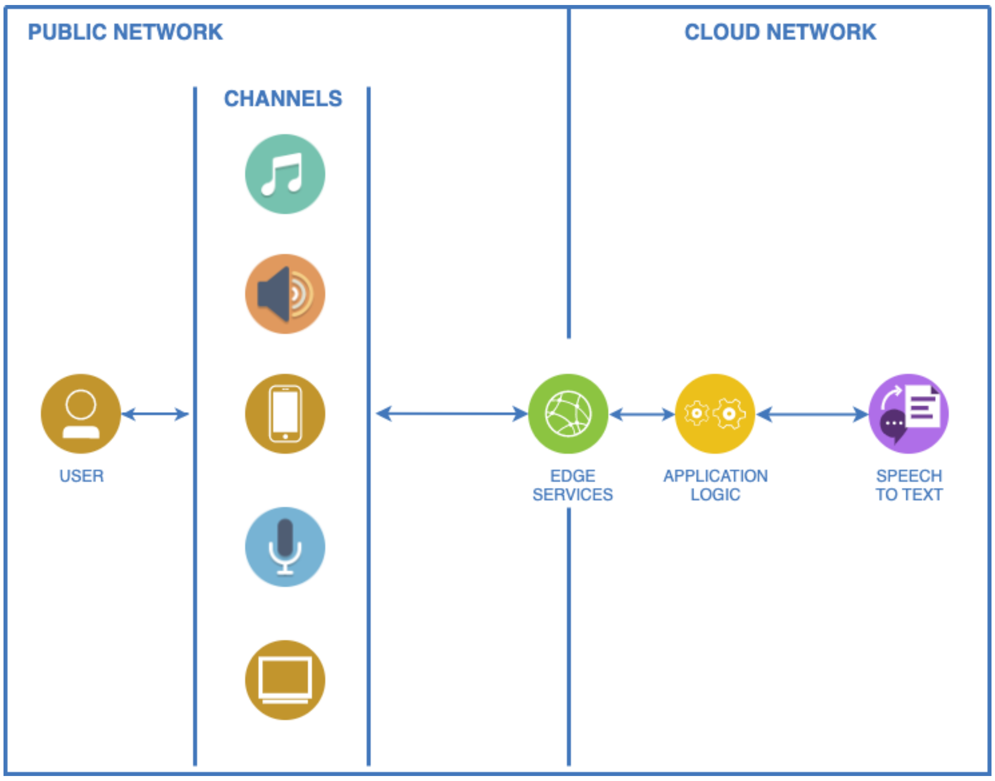
The service provides significant transcription features as well as powerful customization capabilities that can be leveraged for various types of audio.

- **Language Model Training**
  Improve speech recognition accuracy by providing corpora files representing your unique business terminology, acronyms, product names, jargons, and expressions.
- **Acoustic Model Training**
  Improve speech recognition accuracy by providing audio samples representing your unique regional dialect(s) and domain-specific acoustical environments.
- **Speech Grammar**
  Deepen your Language Model Training by applying rules to recognize specific phrases, words, letters, numbers or lists.
- **Speaker Diarization**
  Recognize who said what in a multi-participant voice exchange. Currently optimized for two-way call center conversation but can detect up to 6 different speakers.
- **Word Alternatives, Confidence and Timestamp**
  Response includes alternate words that are acoustically similar to the primary word detected, confidence level of each word detected and the timestamps in the audio input at which the words were detected.
- Check out our Documentation to see our full suite of features and functionalities.

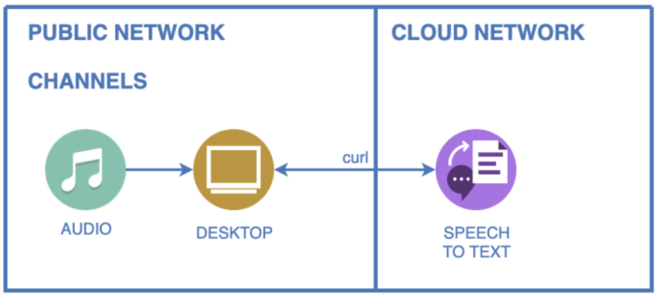
This lab explores the customization capabilities and grammar support in Watson Speech to Text service.

1.2 High Level Runtime Architecture

Speech to Text deployments can have varied architecture. The following diagram illustrates an example in which audio input from user is transcribed using Speech to Text service. The audio can be from various channels such as mobile devices, internet browser, live microphone, stored audio files etc. The application logic layer can be used invoke the speech to text service and process the transcription response.



This lab uses a much simplified architecture, which can be outlined as the following diagram.



Speech to Text is powered by several machine learning models that are provided out of the box and are customizable. The models are provided for several natural languages.

1.3 User Training Capabilities

The IBM Speech to Text service offers a model training interface that you can use to augment and customize its speech recognition capabilities. You can train the service to improve the accuracy of speech recognition requests by customizing a base model for your domain and audio. Customization is available for all supported languages at various levels.

The model training interface supports both the creation and training of custom language models and custom acoustic models. The interfaces for both types of custom models are similar and straightforward to use. Using either type of custom model with a recognition request is also straightforward, where you simply specify the customization ID of the model(s) that you created in your recognition request.

Speech recognition works the same with or without a custom model. When you use a custom model for speech recognition, you can use all of the input and output parameters that are normally available with a recognition request.

1.3.1 Language Training

The service was developed with a broad, general audience in mind. The service's base vocabulary contains many words that are used in everyday conversation. Its models provide sufficiently accurate recognition for many applications. But they can lack knowledge of specific terms that are associated with particular domains.

The language model customization interface can improve the accuracy of speech recognition for domains such as medicine, law, information technology, and others. By using language model customization, you can expand and tailor the vocabulary of a base model to include domain-specific terminology.

You can create a custom language model and add corpora and words specific to your domain. Once you train the custom language model on your enhanced vocabulary, you can use it for customized speech recognition. The service can typically train any custom model in a matter of minutes. The level of effort that it takes to create a model depends on the data that you have available for the model and how well it's organized before training. This lab will take you through the steps for creating a custom language model.

Check out our documentation to read more about Language Model Customization.

1.3.2 Acoustic Training

Similarly, the service was developed with base acoustic models that work well for various audio characteristics. But in cases like the following, adapting a base model to suit your audio can improve speech recognition:

- Your acoustic channel environment is unique. For example, the environment is noisy, microphone quality or positioning are suboptimal, or the audio suffers from far-field effects.
- Your speakers' speech patterns are atypical. For example, a speaker talks unusually fast or the audio includes very casual conversation.
- Your speakers' accents are pronounced. For example, the audio includes speakers who are talking in a non-native or second language.

The acoustic model customization interface can adapt a base model to your environment and speakers. You create a custom acoustic model and add audio data (audio resources) that closely match the acoustic signature of the audio that you want to transcribe. Once you train the custom acoustic model with your audio resources, you can use it for customized speech recognition.

The length of time that it takes the service to train the custom model depends on how much audio data the model contains. The level of effort that it takes to create a model depends on the audio data that you have available for the model. It also depends on whether you use transcriptions of the audio.

Check out our documentation to read more about Acoustic Model Customization.

1.3.3 Grammars

Custom language models allow you to expand the service's base vocabulary. Grammars enable you to restrict the words that the service can recognize from that vocabulary. When you use a grammar with a custom language model for speech recognition, the service can recognize only words, phrases, and strings that are recognized by the grammar. Because the grammar defines a limited search space for valid matches, the service can deliver results faster and more accurately.

You add a grammar to a custom language model and train the model just as you do for a corpus. Unlike a corpus, however, you must explicitly specify that a grammar is to be used with a custom model during speech recognition. This lab will take you through the steps for creating a grammar along with your custom language model.

Check out our documentation to read more about Grammar support in IBM Speech to Text service.

2 Getting Started

This lab is based on Watson Speech to Text service available in IBM public cloud. You need a Plus plan instance of Speech to Text to perform customizations. To create a Plus plan instance, you will need either a Pay-As-You-Go or Subscription account with IBM Cloud. Next section provides detailed instruction to create IBM Cloud account and Speech to Text instance.

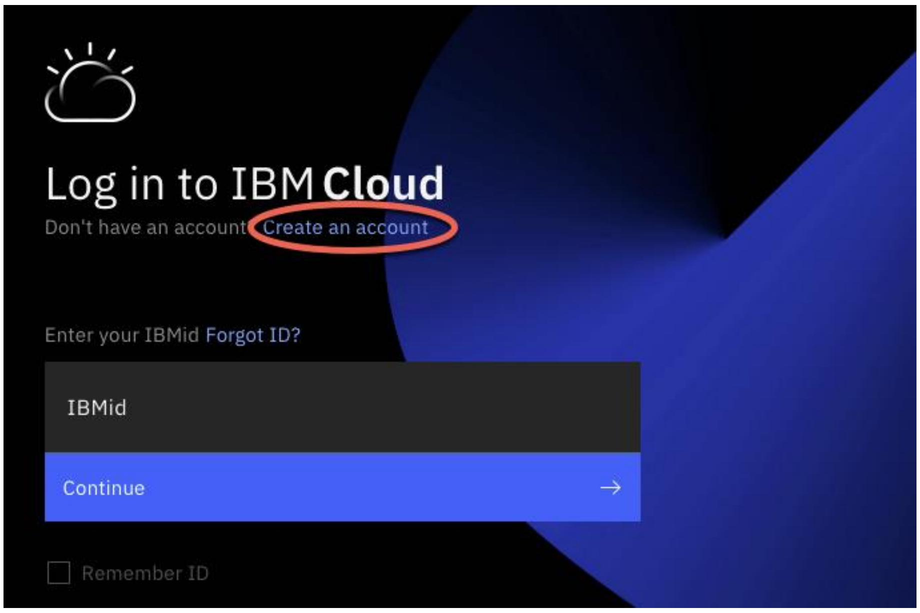
If you have existing IBM Cloud account with necessary account type or Speech to Text instance in Plus plan, you can use that. In that case, the steps irrelevant to you from following sections about creating IBM Cloud account and Speech to Text instance can be skipped.

You can read more about IBM Cloud account types and Plus plan from following pages

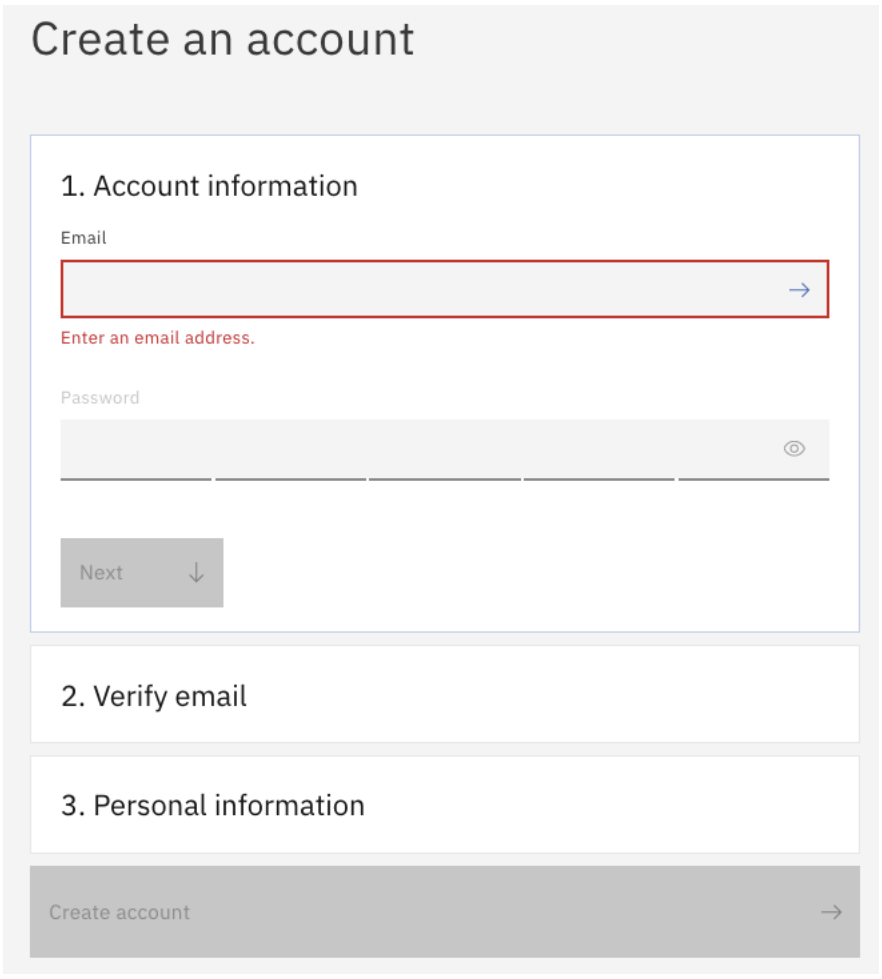
https://cloud.ibm.com/docs/account?topic=account-accounts#accounts
https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-faq-pricing

2.1 Set up your IBM Cloud account

Go to https://cloud.ibm.com and click "Create an Account"


Log in to IBM Cloud
Don't have an account? Create an account
Enter your IBMid Forgot ID?
IBMid
Continue
Remember ID

Enter your email address, verify it, and provide your information.



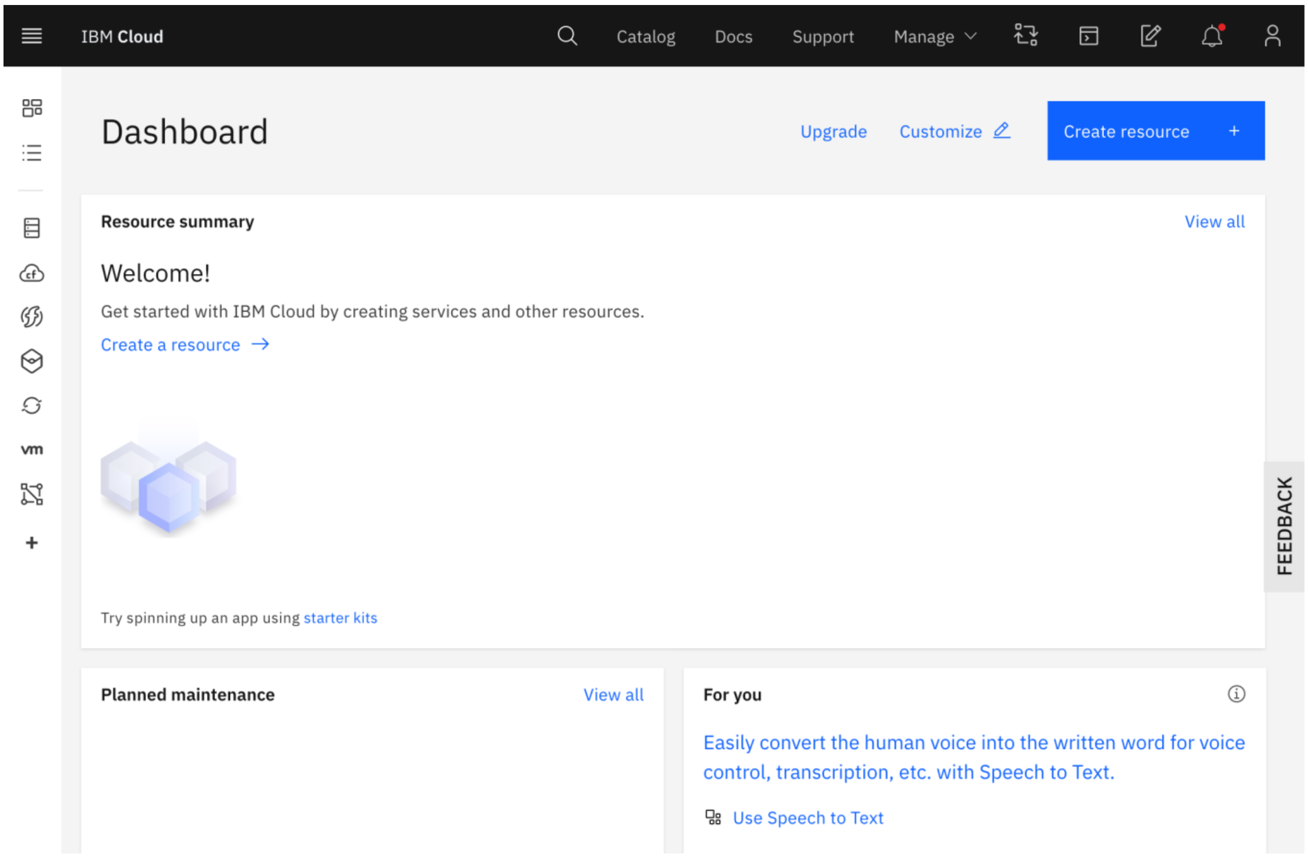
Once you have verified your email address and filled in the required fields, the “Create account” button will become active and you can click it. Within few minutes, you will receive an email that your account has been created.

Open the “Welcome to IBM Cloud” email and click “Log in”

When you login to IBM Cloud for the first time, you will be prompted to review IBM’s privacy policy. Proceed if you agree.

2.2 Provision Speech to Text instance

Once logged in to IBM Cloud, your landing page should look like below



Resource summary

View all

Welcome!

Get started with IBM Cloud by creating services and other resources.

Create a resource →

Try spinning up an app using starter kits

Planned maintenance

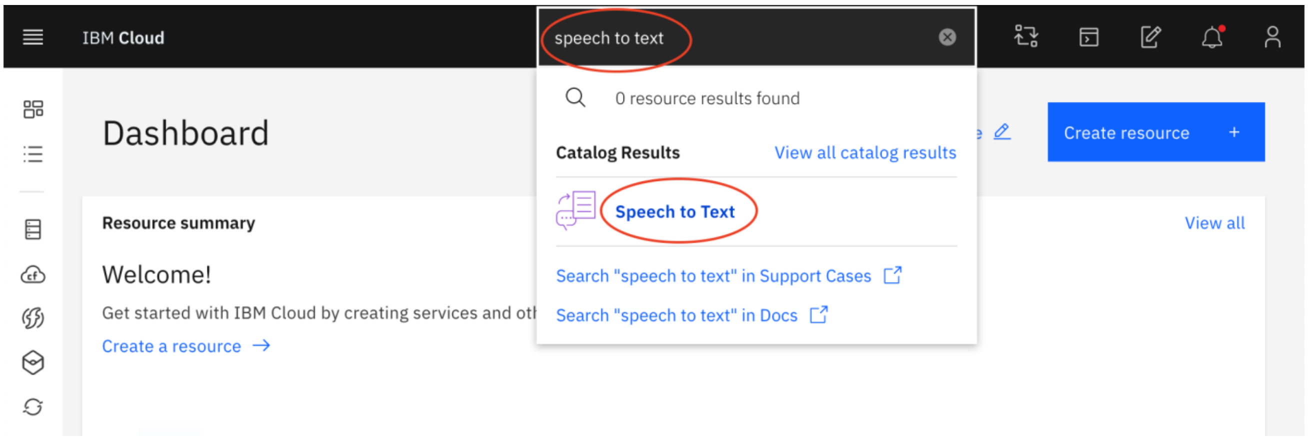
View all

For you

Easily convert the human voice into the written word for voice control, transcription, etc. with Speech to Text.

Use Speech to Text

If you don't already have a Plus plan Speech to Text instance, search for "speech to text" in the top search bar and click "Speech to Text" from the Catalog Results.



speech to text

0 resource results found

Catalog Results

View all catalog results

Speech to Text

Search "speech to text" in Support Cases

Search "speech to text" in Docs

Resource summary

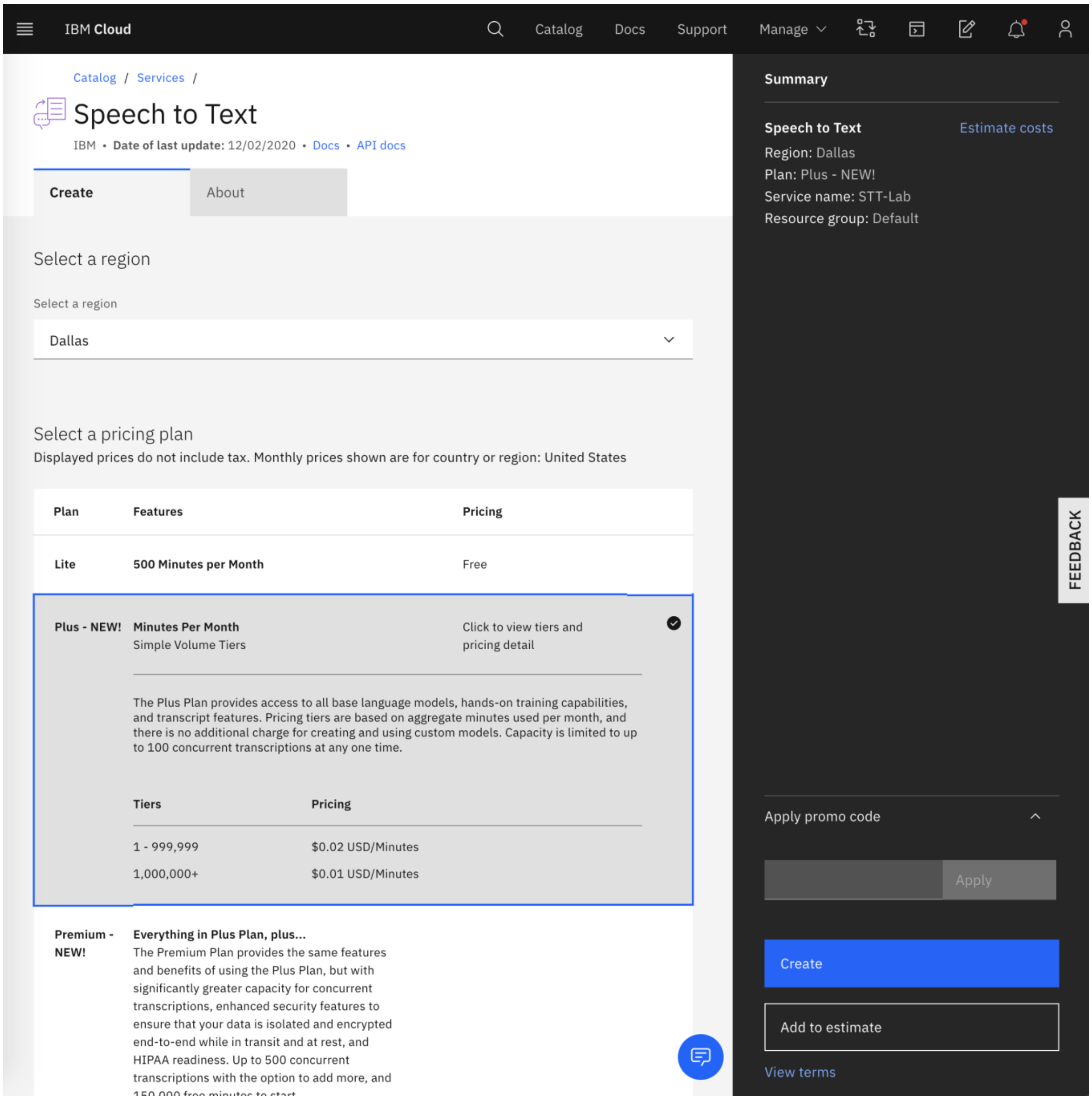
View all

Welcome!

Get started with IBM Cloud by creating services and oth

Create a resource →

Speech to Text can be created with Lite, Plus and Premium plans. Features addressed in this lab are available in Plus or higher plan. Choose a region closer to you, review the costs associated with the plan you select and click Create.



Speech to Text

IBM • **Date of last update:** 12/02/2020 • Docs • API docs

Select a region

Select a region: Dallas

Select a pricing plan

Displayed prices do not include tax. Monthly prices shown are for country or region: United States

Plan	Features	Pricing
Lite	500 Minutes per Month	Free
Plus - NEW! ✓	Minutes Per Month Simple Volume Tiers	Click to view tiers and pricing detail

The Plus Plan provides access to all base language models, hands-on training capabilities, and transcript features. Pricing tiers are based on aggregate minutes used per month, and there is no additional charge for creating and using custom models. Capacity is limited to up to 100 concurrent transcriptions at any one time.

Tiers	Pricing
1 - 999,999	$0.02 USD/Minutes
1,000,000+	$0.01 USD/Minutes

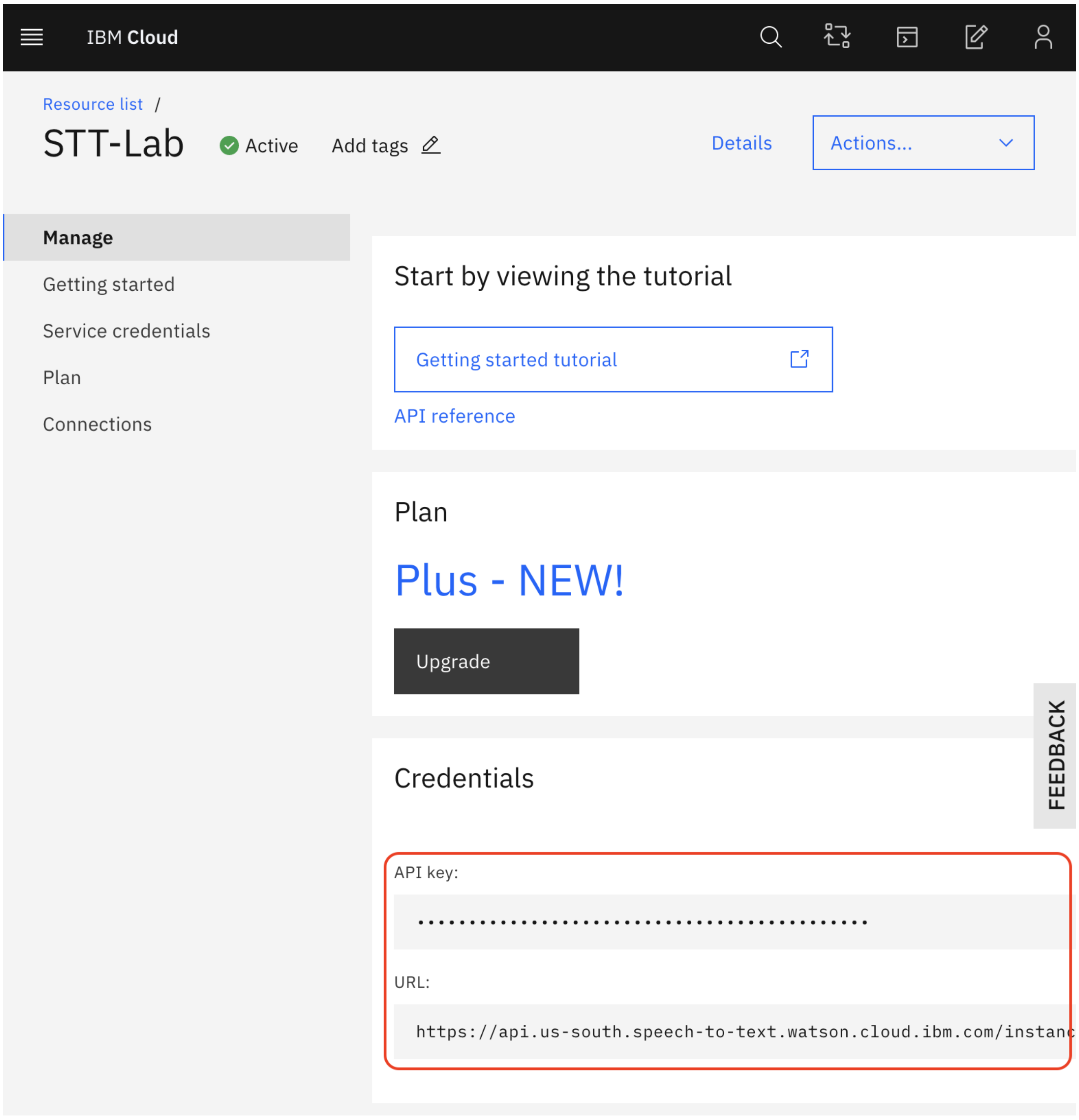
Premium - NEW! **Everything in Plus Plan, plus...** The Premium Plan provides the same features and benefits of using the Plus Plan, but with significantly greater capacity for concurrent transcriptions, enhanced security features to ensure that your data is isolated and encrypted end-to-end while in transit and at rest, and HIPAA readiness. Up to 500 concurrent transcriptions with the option to add more, and

Summary

Speech to Text Estimate costs

Region: Dallas
Plan: Plus - NEW!
Service name: STT-Lab
Resource group: Default

Note the API key and URL from the following start page of the service instance



API key:

```
............................................
```

URL:

```
https://api.us-south.speech-to-text.watson.cloud.ibm.com/instanc
```

This API key and URL can be used with the curl commands to do your own customization.

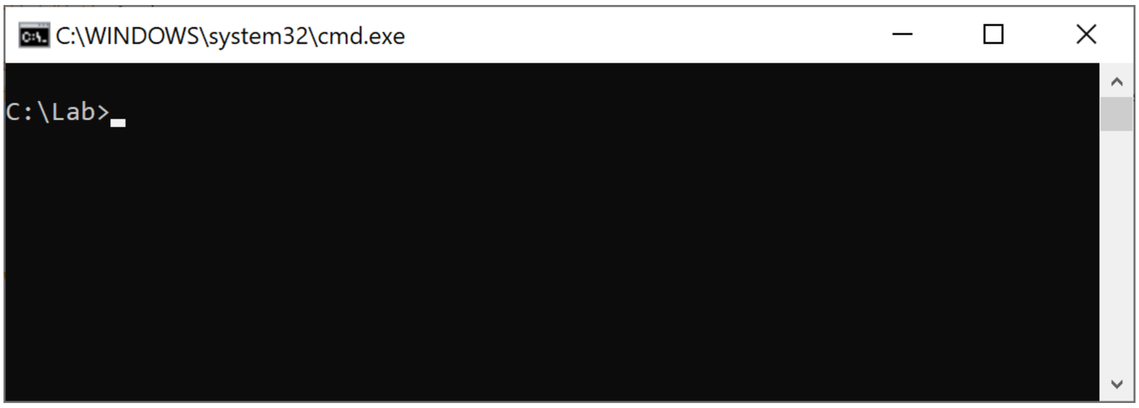
1. API key – Access key to use the API
2. URL – Service location

2.3 Environment Setup

Designate a directory in your desktop to keep the lab materials such as audio files, data files and scripts. For example, `C:\Lab`, if you use Windows or `~/Lab`, if you use Mac. Create the directory, if you do not have it already. This lab guide refers to this directory as **work directory**.

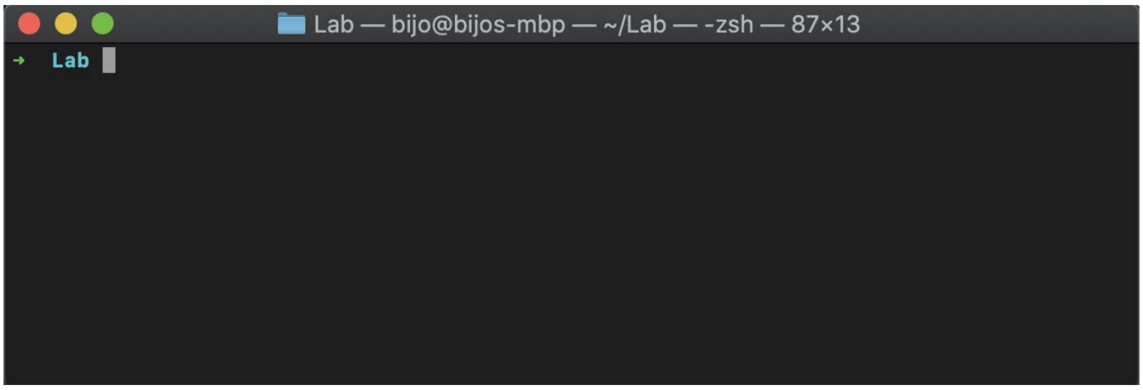
All terminal commands and scripts listed in this lab are defined to be executed from the work directory. If you execute from a different directory, you will need to make necessary changes to the commands.

Windows:



```
C:\Lab>
```

Mac:



```
→  Lab
```

Since API Key and URL are part of almost all commands you use this lab, for convenience, we store them as environment variables and the commands refer to them as variables. If you open a new terminal window, you will need to set them again in the new window.

Windows:

```
C:\Lab> set APIKEY=[Your API Key with Double Quotes]
C:\Lab> set URL=[Your Service URL without Double Quotes]
```

Mac:

```
~/Lab$ export APIKEY=[Your API Key with Double Quotes]
~/Lab$ export URL=[Your Service URL without Double Quotes]
```

For example, if your API Key is b1DzrXLeZWnuRdylhacJTA4e-_Y4Vk_2bZoO5TEDErt1 and URL is https://api.us-south.speech-to-text.watson.cloud.ibm.com/instances/cb07aa2a-e12e-4ae3-a8e1-6c2b31ef385e, the commands will be

Windows:

```
C:\Lab> set APIKEY="b1DzrXLeZWnuRdylhacJTA4e-_Y4Vk_2bZoO5TEDErt1"
C:\Lab> set URL=https://api.us-south.speech-to-
text.watson.cloud.ibm.com/instances/cb07aa2a-e12e-4ae3-a8e1-6c2b31ef385e
```

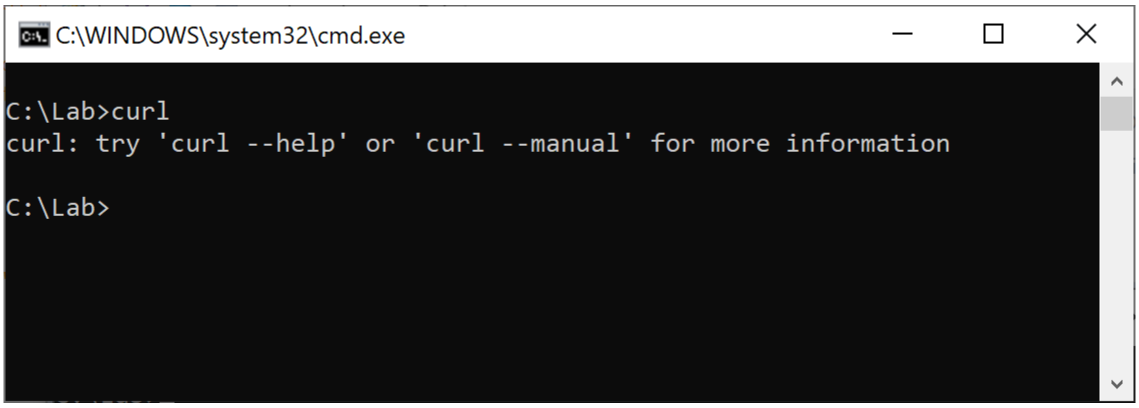
Mac:

```
~/Lab$ export APIKEY="b1DzrXLeZWnuRdylhacJTA4e-_Y4Vk_2bZoO5TEDErt1"
~/Lab$ export URL=https://api.us-south.speech-to-
text.watson.cloud.ibm.com/instances/cb07aa2a-e12e-4ae3-a8e1-6c2b31ef385e
```

Most commands use curl to execute. So, let's make sure you have curl in your desktop. Run following command in your terminal

Windows:

```
C:\Lab> curl
```

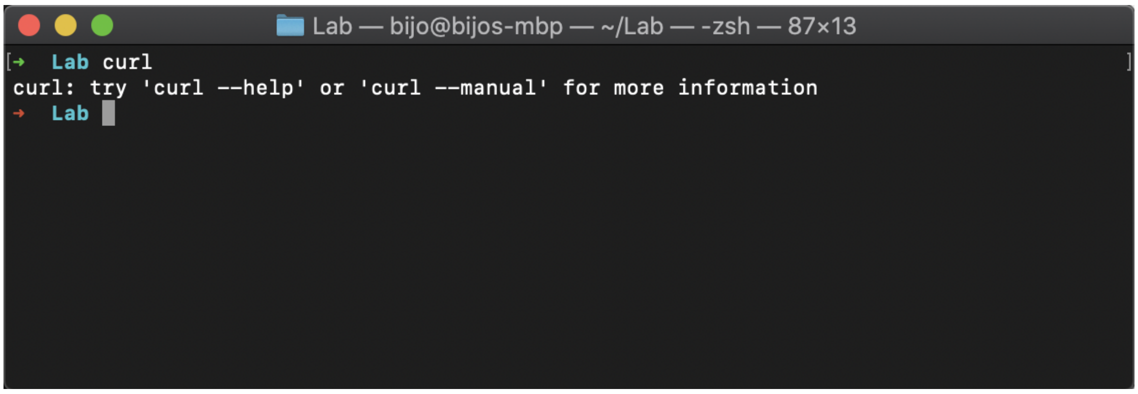


```
C:\Lab>curl
curl: try 'curl --help' or 'curl --manual' for more information

C:\Lab>
```

Mac:

```
~/Lab$ curl
```



```
➜  Lab curl
curl: try 'curl --help' or 'curl --manual' for more information
➜  Lab
```

If you see the same response as in the above screenshot, you have curl installed and proceed.

> In case if you don't have curl in your desktop, download curl binary for your operating system from following URL, install or extract it as the case may be.
>
> https://curl.haxx.se/download.html
>
> If not already, all curl to your execution path so that it's available in your terminal.
>
> Once the path is set, try `curl` command in terminal again to verify you get a response similar to what you see in above screenshot.

All audio, data and scripts used in this lab can be downloaded as a zip file from following link.

https://www.ibm.com/demos/live/content/watson/stt/lab/lab-materials.zip

Download and extract the content to your work directory

2.4 Warming Up

To request speech recognition with the IBM Speech to Text service, you need to provide only the audio that is to be transcribed. The service offers the same basic transcription capabilities with each of its interfaces: the WebSocket interface, the synchronous HTTP interface, and the asynchronous HTTP interface.

In this lab, we use synchronous HTTP interface using curl.

The following sections show basic transcription requests, with no optional input or output parameters.

You have an audio file audio-file.flac in your work directory. Let's transcribe that audio using your Speech to Text service instance. You can play and hear the audio locally in your desktop first if you would like to.

Issue the following command to call the service's /v1/recognize method for basic transcription with no parameters. It uses the Content-Type header to indicate the type of the audio, audio/flac and uses the default language model, en-US_BroadbandModel, for transcription

Windows:

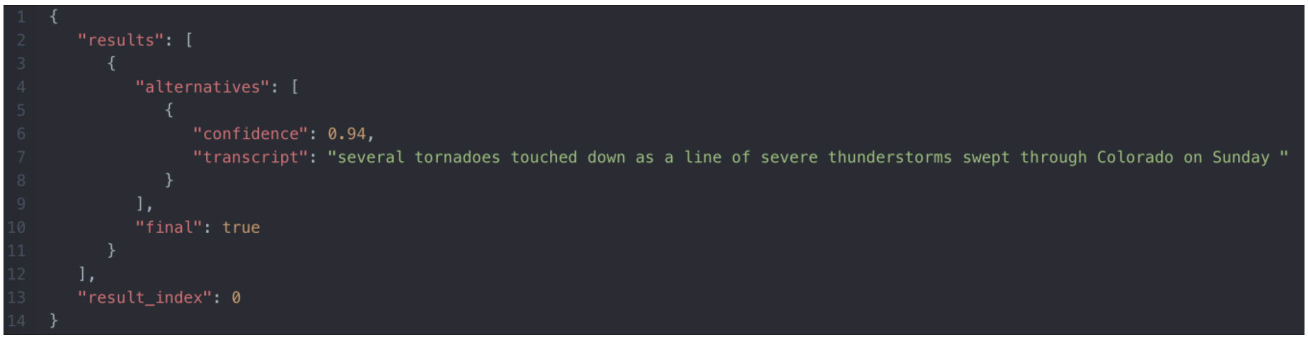
```
C:\Lab> warmup.bat
```

Mac:

```
~/Lab$ ./warmup.sh
```

> If the script fails, please check
>
> 1. If you have the environment variables APIKEY and URL defined as described in previous section.
> 2. If the warmup script has sufficient execute permissions in your operating system.
>
> If insufficient execute permission is the cause of failure, please remember to add execute permission to every script file you have in your work directory to avoid further problems in upcoming lab sections.

Open the output file warmup.json in a text editor. The transcription result look like the following



```
{
   "results": [
      {
         "alternatives": [
            {
               "confidence": 0.94,
               "transcript": "several tornadoes touched down as a line of severe thunderstorms swept through Colorado on Sunday "
            }
         ],
         "final": true
      }
   ],
   "result_index": 0
}
```

Transcript text can be seen against the name "transcript". You can also see "confidence" value of this transcription.

For a full list of parameters and additional details you can get with the transcription, such as speaker labels, word alternatives etc., see the API specification at

https://cloud.ibm.com/apidocs/speech-to-text#recognize-audio

3 Exercise A: Train a Language Model

This exercise will show you how audio files containing domain terminology can be transcribed with higher accuracy with use of a custom language model.

To better appreciate this section of the lab, you may want to play the audio files in respective sections and listen to it.

3.1 Transcribe without a Trained Custom Language Model

First, let's try to transcribe an audio file automotive.flac from your work directory without using a trained custom language model. If you listen to the audio, you will notice the speakers mentioning certain domain terms such as Nzeckster, key escutcheon, Nzeckster Countlone etc.

Issue the following command to call the service's /v1/recognize method for basic transcription with no parameters. It uses the Content-Type header to indicate the type of the audio, audio/flac and uses the default language model, en-US_BroadbandModel, for transcription

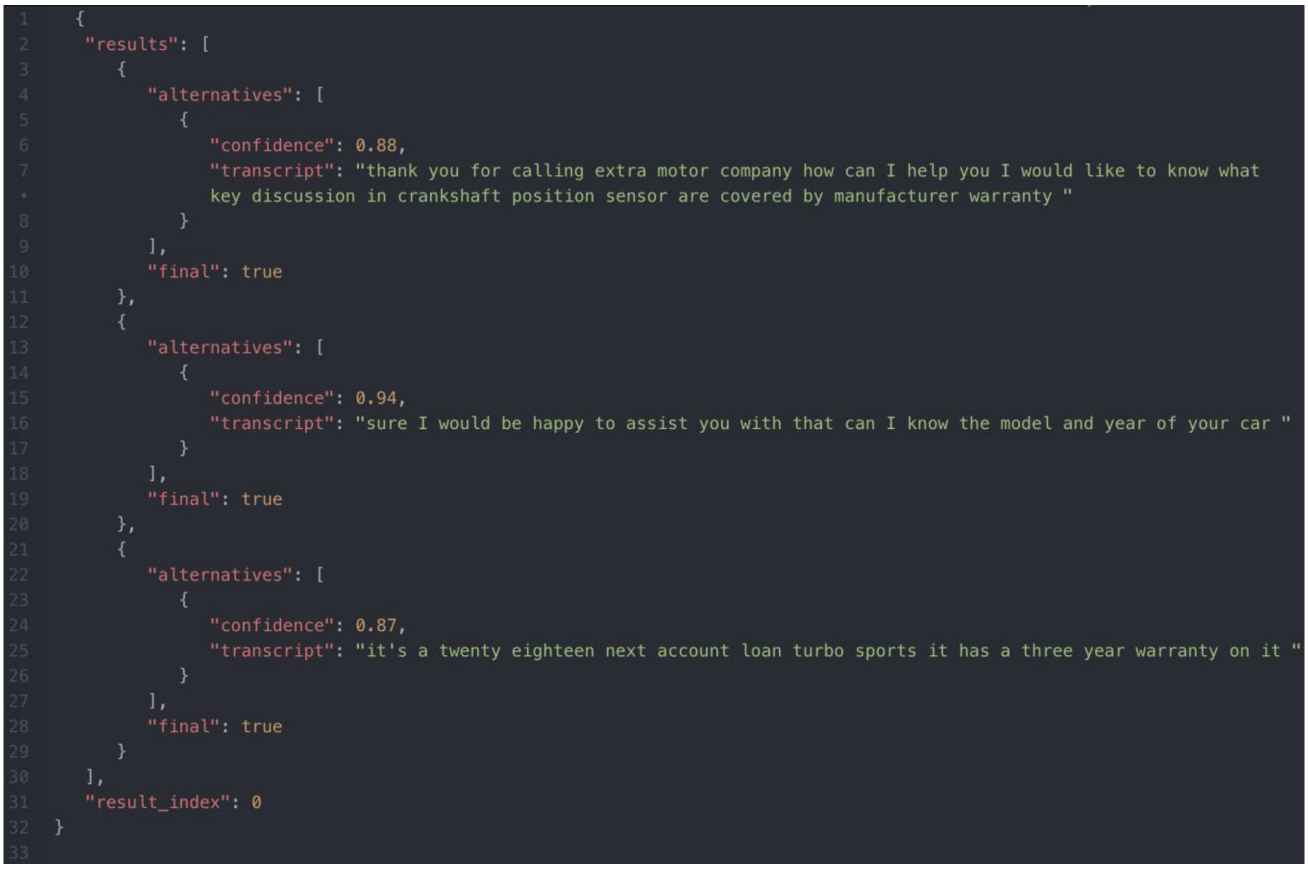
Windows:

```
C:\Lab> transcribe-automotive-flac.bat
```

Mac:

```
~/Lab$ ./transcribe-automotive-flac.sh
```

Open the output file automotive-flac.json in a text editor. The transcription result look like the following



```
{
 "results": [
    {
       "alternatives": [
          {
             "confidence": 0.88,
             "transcript": "thank you for calling extra motor company how can I help you I would like to know what
             key discussion in crankshaft position sensor are covered by manufacturer warranty "
          }
       ],
       "final": true
    },
    {
       "alternatives": [
          {
             "confidence": 0.94,
             "transcript": "sure I would be happy to assist you with that can I know the model and year of your car "
          }
       ],
       "final": true
    },
    {
       "alternatives": [
          {
             "confidence": 0.87,
             "transcript": "it's a twenty eighteen next account loan turbo sports it has a three year warranty on it "
          }
       ],
       "final": true
    }
 ],
 "result_index": 0
}
```

As you can see in the transcript, some of these domain terms Nzeckster, key escutcheon, Nzeckster Countlone etc. did not get transcribed correctly.

Now, let's try the same audio in a different audio format. You have the file automotive.wav in your work directory, which is the same conversation in a different audio format.

You can issue the following command to call the same API method with the audio file automotive.wav. The Content-Type header now indicates the type of the audio as audio/wav.

Windows:

```
C:\Lab> transcribe-automotive-wav.bat
```

Mac:

```
~/Lab$ ./transcribe-automotive-wav.sh
```

Output in automotive-wav.json will be similar to the previous output seen in automotive-flac.json.

3.2 Create and Train a Custom Language Model

Custom Language Model uses corpus as one of the ways you can customize a model. Language model corpus is a text file containing common utterances in your domain that you expect your users would say, the audio for which is intended to be transcribed with your Speech to Text instance. You can extrapolate the utterances with common variations. You can also repeat the utterances to improve the accuracy.

An example language model will look like this.

```
calling Nzeckster Motor Company
calling Nzeckster Motor Company
is key escutcheon covered by warranty
do you cover key escutcheon in manufacturer warranty
is crankshaft position sensor covered in warranty
does my warranty cover crankshaft position sensor
I use a Nzeckster Countlone turbo sports model
mine is a Nzeckster Countlone
it's a Nzeckster Countlone
```

For this lab, the corpus is provided to you in the automotive-lm.txt file that you have available in your work directory. Following steps describes how to create and train a custom language model using this corpus.

When you create a custom language model, you need to provide a name for your new model and also need to provide the name of the base language model your custom model is based on. You can also provide an optional description.

For a full list of supported language models, visit
https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-models#modelsList

In this lab, we are creating a custom model against US English Broadband base model.

Issue the following command to call the service's /v1/customizations method to create a new custom language model in your Speech to Text instance.

Windows:

```
C:\Lab> create-automotive-lm.bat
```

Mac:

```
~/Lab$ ./create-automotive-lm.sh
```

The service creates a new language customization id and it would be stored in the file automotive-customization-id.json in your work directory.

View the content of automotive-customization-id.json by opening it in a text editor, which looks similar to below

```
{"customization_id": "00f1b7134-23d1-4ad2-8b0d-a1218ad69abe"}
```

You will use the value against name "customization_id" in following commands. So please note that down. For convenience, you can set that an environment variable as below. Following sections in this lab uses this environment variable in commands. If you open a new terminal window, you will need to set them again in the new window.

Windows:

```
C:\Lab> set AUTO_LM=[Your Customization Id without Double Quotes]
```

Mac:

```
~/Lab$ export AUTO_LM=[Your Customization Id without Double Quotes]
```

Now you can add corpus to your custom language model you just created

Issue the following command to call the service's /v1/customizations/{customization_id}/corpora/{corpus_name} method to add the corpus to the new custom language model.

Windows:

```
C:\Lab> add-autocorpus.bat
```

Mac:

```
~/Lab$ ./add-autocorpus.sh
```

Once added, you can check the status of the corpus by issuing following command

Windows:

```
C:\Lab> check-autocorpus.bat
```

Mac:

```
~/Lab$ ./check-autocorpus.sh
```

The output will look similar to below

```
{
   "out_of_vocabulary_words": 2,
   "total_words": 53,
   "name": "autocorpus",
   "status": "being_processed"
}
```

Initial status will be **being_processed**. Reissue the command to see updated status. Wait until the status changes to **analyzed**.

Now you are ready to train your custom language model.

Issue the following command to call the service's /v1/customizations/{customization_id}/train method to initiate training of the new custom language model.

Windows:

```
C:\Lab> train-automotive-lm.bat
```

Mac:

```
~/Lab$ ./train-automotive-lm.sh
```

Once the training is initiated, you can check the status of the custom model by issuing following command.

Windows:

```
C:\Lab> check-automotive-lm.bat
```

Mac:

```
~/Lab$ ./check-automotive-lm.sh
```

The output will look similar to below

```
{
   "owner": "1a864ef2-711d-4a0a-19e2-31bf4b9bdef0",
   "base_model_name": "en-US_BroadbandModel",
   "customization_id": "00f1b7134-23d1-4ad2-8b0d-a1218ad69abe",
   "dialect": "en-US",
   "versions": ["en-US_BroadbandModel.v2020-01-16"],
   "created": "2020-05-04T02:26:29.495Z",
   "name": "autolm",
   "description": "Automotive Model",
   "progress": 0,
   "language": "en-US",
   "updated": "2020-05-05T17:01:20.214Z",
   "status": "training"
}
```

Initial status will be **training**. Reissue the command to see updated status. Wait until the status changes to **available**.

Now you are ready to use your custom language model.

3.3 Transcribe with Custom Language Model

Let's try to transcribe the same automotive.flac from your work directory using custom language model you just created and see if there is better detection of the domain terms.

Issue the following command to call the service's /v1/recognize. The customization id being passed with parameter language_customization_id. It uses the Content-Type header to indicate the type of the audio, audio/flac and uses the default language model, en-US_BroadbandModel, for transcription

Windows:

```
C:\Lab> transcribe-automotive-flac-customized.bat
```

Mac:

```
~/Lab$ ./transcribe-automotive-flac-customized.sh
```

Open the output file automotive-flac-customized.json in a text editor. The transcription result look like the following

```
 {
  "results": [
     {
        "alternatives": [
           {
              "confidence": 0.97,
              "transcript": "thank you for calling Nzeckster motor company how can I help you I would like to know what key escutcheon in crankshaft position sensor covered by manufacturer warranty "
           }
        ],
        "final": true
     },
     {
        "alternatives": [
           {
              "confidence": 0.93,
              "transcript": "sure I would be happy to assist you with that can I know the model and year of your car "
           }
        ],
        "final": true
     },
     {
        "alternatives": [
           {
              "confidence": 0.98,
              "transcript": "it's a twenty eighteen Nzeckster Countlone turbo sports it has a three year warranty on it "
           }
        ],
        "final": true
     }
  ],
  "result_index": 0
}
```

As you can see, the domain terms such as Nzeckster, key escutcheon or Nzeckster Countlone are now transcribed with improved accuracy.

Check out an interactive demo of Speech to Text customization, with side by side comparison of base model transcription and custom model transcription

This concludes the language model customization exercise. Now let’s move on to the next section for the grammars exercise.

4 Exercise B: Grammars

This exercise will show you how audio files containing specific formatted data can be transcribed with higher accuracy with use of grammar definition.

For example, when you need to recognize specific words or phrases, such as yes or no, individual letters or numbers, or a list of names, using grammars can be more effective than examining alternative words and transcripts. Moreover, by limiting the search space for valid strings, the service can deliver results faster and more accurately.

Grammar support is provided in Speech to Text service based on Speech Recognition Grammar Specification: https://www.w3.org/TR/speech-grammar/

The service currently does not support all features of the specification. To know more about what's not supported, see below page

https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-grammars#grammarSpecification

Grammar is supported in both Augmented Backus-Naur Form (ABNF) and XML Form.

For example, a yes or no grammar will look like below in ABNF

```
# ABNF 1.0 ISO-8859-1;
language en-US;
mode voice;
root $yesno;

$yesno = yes | no ;
```

In XML Form, it will look like below

```
<grammar version="1.0" xml:lang="en-US" root="yesno"
      xmlns="http://www.w3.org/2001/06/grammar">
      <rule id="yesno">
            <one-of>
                  <item>yes</item>
                  <item>no</item>
            </one-of>
      </rule>
</grammar>
```

In this lab, we use XML Form.

4.1 Defining Grammar

Let's imagine a situation when a user reads out loud a claim number into a voice-activated application. The claim number contains letters and numbers. Many of the letters or numbers can easily sound like common words to a speech transcription engine, as the examples below suggest:

T - tea
B - bee, be
C - see, sea
2 - to, too
K - okay
S - yes

Many of these alphanumerics can be challenging to transcribe correctly if one only used the out-of-the-box base language models.

However, for situations like this, we can use a grammar to limit the search space for the transcription engine. If the search is limited to just letters and numbers instead of all dictionary words, the transcription engine will know exactly how to transcribe the identified alphanumerics which will help to improve overall transcription accuracy.

In this lab, let's imagine a claim number which follows the pattern below:

Two letters followed by six numbers. First letter should be either K, Q or T.

An XML grammar definition of this pattern can look similar to following. Note that the section below has trimmed the list of letters and numbers for display purposes. To see the full definition, see the claim-number-grammar.xml which you have in your work directory.

```
<grammar version="1.0" xml:lang="en-US" root="patterns"
      xmlns="http://www.w3.org/2001/06/grammar">
      <rule id="patterns">
            <one-of>
                  <!-- pattern: 8 positions [KQT][A-Z][0-9]{6} -->
                  <item>
                        <ruleref uri="#firstletters" />
                        <ruleref uri="#letters" />
                        <ruleref uri="#numbers" />
                        <ruleref uri="#numbers" />
                        <ruleref uri="#numbers" />
                        <ruleref uri="#numbers" />
                        <ruleref uri="#numbers" />
                        <ruleref uri="#numbers" />
                  </item>
            </one-of>
      </rule>
      <rule id="firstletters">
            <one-of>
                  <item>K.</item>
                  <item>Q.</item>
                  <item>T.</item>
            </one-of>
      </rule>
      <rule id="letters">
            <one-of>
                  <item>A.</item>
                  <item>B.</item>
                  ...
                  ...
                  <item>Y.</item>
                  <item>Z.</item>
            </one-of>
      </rule>
      <rule id="numbers">
            <one-of>
                  <item>zero</item>
                  <item>one</item>
                  ...
                  ...
                  <item>eight</item>
                  <item>nine</item>
            </one-of>
      </rule>
</grammar>
```

4.2 Adding Grammar to Custom Language Model

A Grammar cannot be used by itself. Rather, it must be added to a Custom Language Model, even if that Custom Language Model only contains the Grammar. In this exercise, let's **add** this grammar definition to the automotive custom language model we created in Exercise A.

Issue the following command to call the service's /v1/customizations/{customization_id}/grammars/{grammar_name} method to add the grammar file to custom model.

Windows:
```
C:\Lab> add-grammar.bat
```

Mac:
```
~/Lab$ ./add-grammar.sh
```

Once added, you can check the status of the grammar by issuing following command.

Windows:
```
C:\Lab> check-grammar.bat
```

Mac:
```
~/Lab$ ./check-grammar.sh
```

The output will look similar to below.

```
{
   "out_of_vocabulary_words": 0,
   "name": "claimnumber,
   "status": "being_processed"
}
```

Initial status will be **being_processed**. Reissue the command to see updated status. Wait until the status changes to **analyzed**.

Now you are ready to train your custom language model.

Issue the following command to call the service's /v1/customizations/{customization_id}/train method to initiate training of the new custom language model.

Windows:
```
C:\Lab> train-automotive-lm.bat
```

Mac:
```
~/Lab$ ./train-automotive-lm.sh
```

Once the training is initiated, you can check the status of the custom model by issuing following command.

Windows:
```
C:\Lab> check-automotive-lm.bat
```

Mac:
```
~/Lab$ ./check-automotive-lm.sh
```

The output will look similar to below

```
{
   "owner": "1a864ef2-711d-4a0a-19e2-31bf4b9bdef0",
   "base_model_name": "en-US_BroadbandModel",
   "customization_id": "00f1b7134-23d1-4ad2-8b0d-a1218ad69abe",
   "dialect": "en-US",
   "versions": ["en-US_BroadbandModel.v2020-01-16"],
   "created": "2020-05-04T02:26:29.495Z",
   "name": "autolm",
   "description": "Automotive Model",
   "progress": 0,
   "language": "en-US",
   "updated": "2020-05-05T17:01:20.214Z",
   "status": "training"
}
```

Initial status will be **training**. Reissue the command to see updated status. Wait until the status changes to **available**.

Now you are ready to use new grammar.

4.3 Transcribe without Grammar

Let's try to transcribe the audio file claim-number.flac from your work directory without using grammar. Subsequently, we will transcribe using grammar to see the difference.

This file contains audio clip of a spoken claim number. To better appreciate this section of the lab, you may want to play the audio file claim-number.flac and listen to it.

Issue the following command to call the service's /v1/recognize without any parameters. It uses the Content-Type header to indicate the type of the audio, audio/flac and uses the default language model, en-US_BroadbandModel, for transcription.

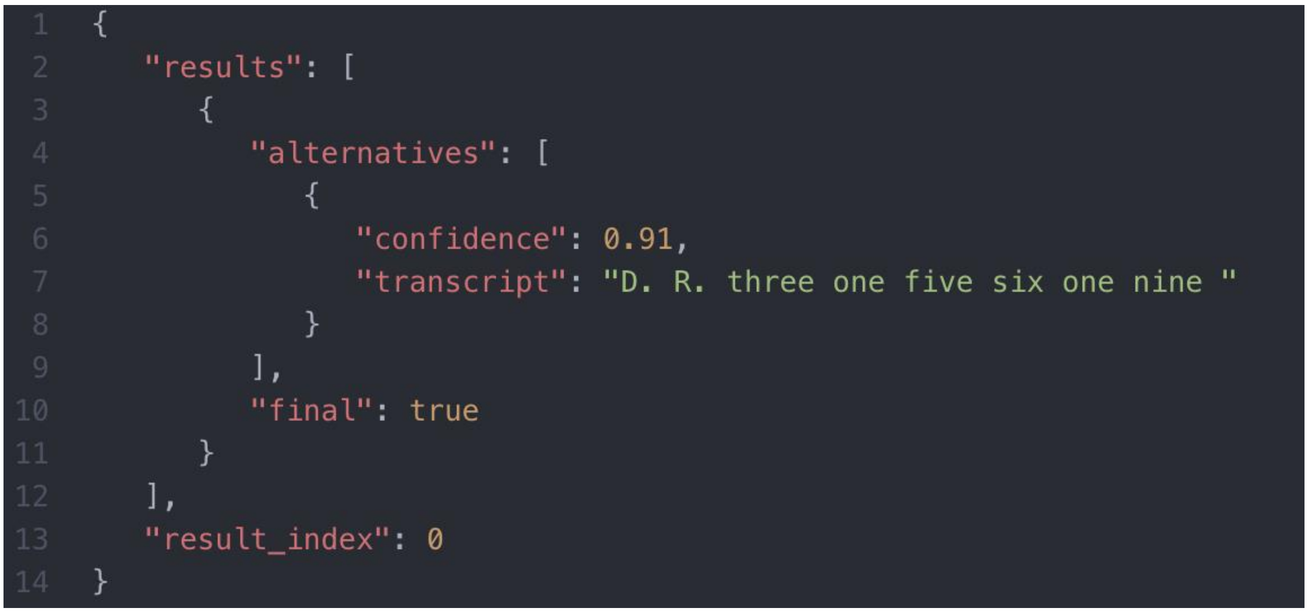
Windows:

```
C:\Lab> transcribe-claim-number-flac.bat
```

Mac:

```
~/Lab$ ./transcribe-claim-number-flac.sh
```

Service response in claim-number.json looks like below



```
{
   "results": [
      {
         "alternatives": [
            {
               "confidence": 0.91,
               "transcript": "D. R. three one five six one nine "
            }
         ],
         "final": true
      }
   ],
   "result_index": 0
}
```

If you had listened to the audio, you would have noticed that the first letter spoken, actually T, sounds very similar to D and gets transcribed incorrectly as D.

Now let's see if grammar makes a difference.

4.4 Transcribe with Grammar

We will use the same audio file claim-number.flac, which is already in your work directory.

Issue the following command to call the service's /v1/recognize. The customization id being passed with parameter language_customization_id and grammar name passed with parameter grammar_name. The command uses the Content-Type header to indicate the type of the audio, audio/flac and uses the default language model, en-US_BroadbandModel, for transcription.

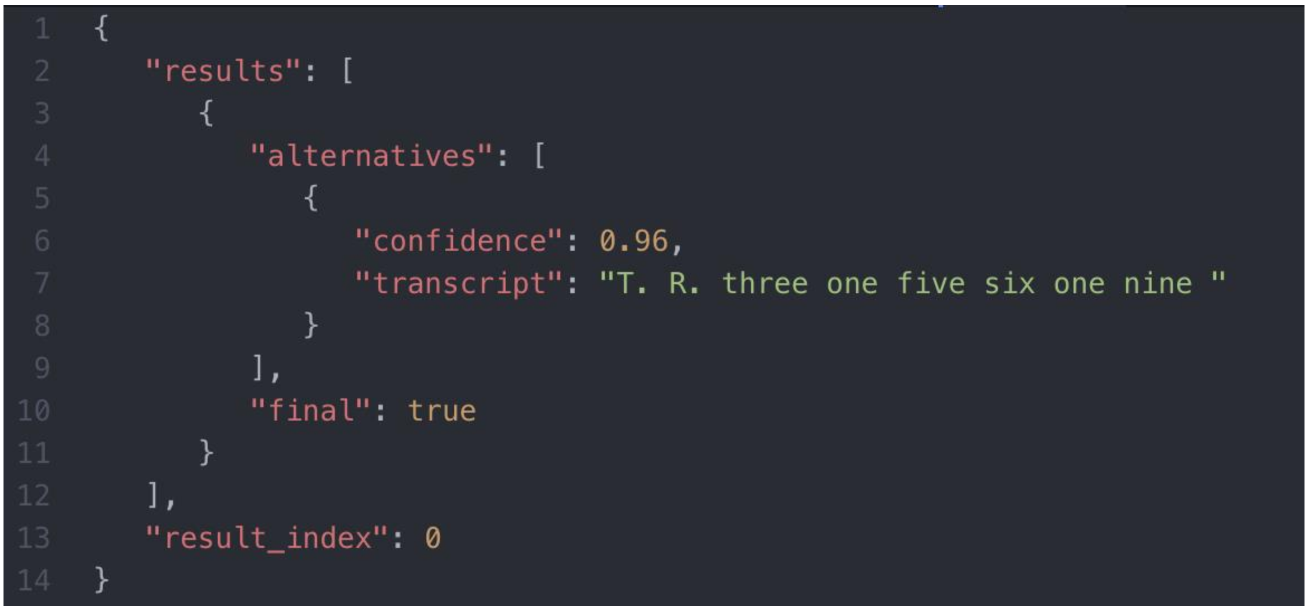
Windows:

```
C:\Lab> transcribe-claim-number-flac-grammar.bat
```

Mac:

```
~/Lab$ ./transcribe-claim-number-flac-grammar.sh
```

Service response in claim-number-grammar.json looks like below



```
{
    "results": [
        {
            "alternatives": [
                {
                    "confidence": 0.96,
                    "transcript": "T. R. three one five six one nine "
                }
            ],
            "final": true
        }
    ],
    "result_index": 0
}
```

You can see the first letter T in the audio is now correctly transcribed.

This concludes the Speech Grammar exercise.

5 Appendix I - Evaluating Accuracy

When you train custom models, Word Error Rate (WER) is a common measure of accuracy to determine if your custom model has improved. It's calculated using below formula.

$$WER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C}$$

where

S is the number of substitutions,
D is the number of deletions,
I is the number of insertions,
C is the number of correct words,
N is the number of words in the reference (N=S+D+C)

However, relying solely on WER could be counterproductive at times. For example, if you are trying to identify ID numbers, incorrect detection of a single character would make the whole ID incorrect, or if you are using the transcription with downstream services such as Watson Assistant, error in one word may not make any difference.

You can read more about it here

https://medium.com/ibm-watson/why-the-overall-voicebot-solution-and-user-experience-are-more-important-than-speech-accuracy-d229bf54aede