**PROJECT REPORT**

On

**“Create A Custom Watson Speech To Text Model Using Specialized Domain Data**​**”**

*Submitted in partial fulfilment of the requirements for the award of*

**Bachelor of Technology (B. Tech)**

In the department of

**Computer Science & Engineering**

****

*Submitted by​*:

**Hunnan Hussain (ET16BTHCS023)**

**Hafiz Hussain (ET16BTHCS063)**

**Nawed Zaman (ET16BTHCS066)**

**Dipak Gogoi (ET16BTHCS064)**

**Engan Panggeng (ET16BTHEE022L)**

*Under the Supervision and Guidance of*

**Mr. AADITYA MAHESHWARI**

**(External Mentor)**

**Ms. NILAKSHI GOGOI**

**(Internal Mentor)**

**School of Engineering & Technology**

**The Assam Kaziranga University, Jorhat, Assam**

**( Nov –2019 )**

**CERTIFICATE**

This is to certify that the project report entitled ***“****​***CREATE A CUSTOM WATSON SPEECH TO TEXT MODEL USING SPECIALIZED DOMAIN DATA*”***​*​,​*submitted to the School of Engineering & Technology(SET), ​**THE ASSAM KAZIRANGA UNIVERSITY, JORHAT, ASSAM,** in​ partial fulfilment for the completion of ​**Semester – 7**​**th**​of the degree of ​**Bachelor ofTechnology** ​in the department of​**Computer Science & Engineering**​, is a recordof bonafide work carried out by ​**Mr. Nawed Zaman**, ​**Roll No**​.​**ET16BTHCS066, Mr. Hunnan Hussain, Roll No. ET16BTHCS023, Mr. Hafiz Hussain, Roll No. ET16BTHCS063, Mr. Dipak Gogoi, Roll No. ET16BTHCS064, Mr. Engan Panggeng , Roll No. ET16BTHEE022L**​, under myguidance.

All help received by us from various sources have been duly acknowledged.

No part of this report has been submitted elsewhere for award of any other degree.

-----------------------------​

​

-----------------------------​

**H**​**.O.D CSE Dept.**

**Mrs. Mousoomi Borah**

**External Mentor**

**Mr. Aaditya Maheshwari** ​**(IBM)**

-----------------------------

**Internal Mentor**

**Ms. Nilakshi Gogoi**

**ACKNOWLEDGEMENT**

The Project opportunity we had with ​**IBM** at ​**Kaziranga University Jorhat**​was a great chance for learning and gaining practical knowledge. Therefore, we consider ourselves very lucky as we were provided with an opportunity to be a part of it.

Our indebtedness and gratitude is towards those individuals who have helped us shape this report. We must record our immense gratitude to those who have helped us to learn.

We would like to thank ​ **Ms. Nilakshi Gogoi (**​**Assistant Professor, Internal Mentor), Mr. Aaditya Maheshwari (External Mentor)**​,​under whose guidance we have completed ourproject . All these people were of immense importance regarding the knowledge and support for the well-furnished equipment. We greatly acknowledge the help and the mental strength provided by entire ​**Computer Science & Engineering** and **IBM** family for encouraging and providing knowledge & guidance related with ourproject.

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| Sl. No | Topic | Page no. |
| 1. | Introduction | 5 |
| 2. | Featured Technologies | 6-7 |
| 3. | Download and prepare the data | 8 |
| 4. | Procedure | 9-12 |
| 5. | Results | 13-14 |
| 6. | Future Scope | 15 |
| 7. | Conclusion | 16 |

**Introduction**

**Speech recognition** is an [interdisciplinary](https://en.wikipedia.org/wiki/Interdisciplinary) subfield of [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) that develops methodologies and technologies that enables the recognition and [translation](https://en.wikipedia.org/wiki/Translation) of spoken language into text by computers. It is also known as **automatic speech recognition** (**ASR**), **computer speech recognition** or **speech to text** (**STT**). It incorporates knowledge and research in the [linguistics](https://en.wikipedia.org/wiki/Linguistics), [computer science](https://en.wikipedia.org/wiki/Computer_science), and [electrical engineering](https://en.wikipedia.org/wiki/Electrical_engineering) fields.

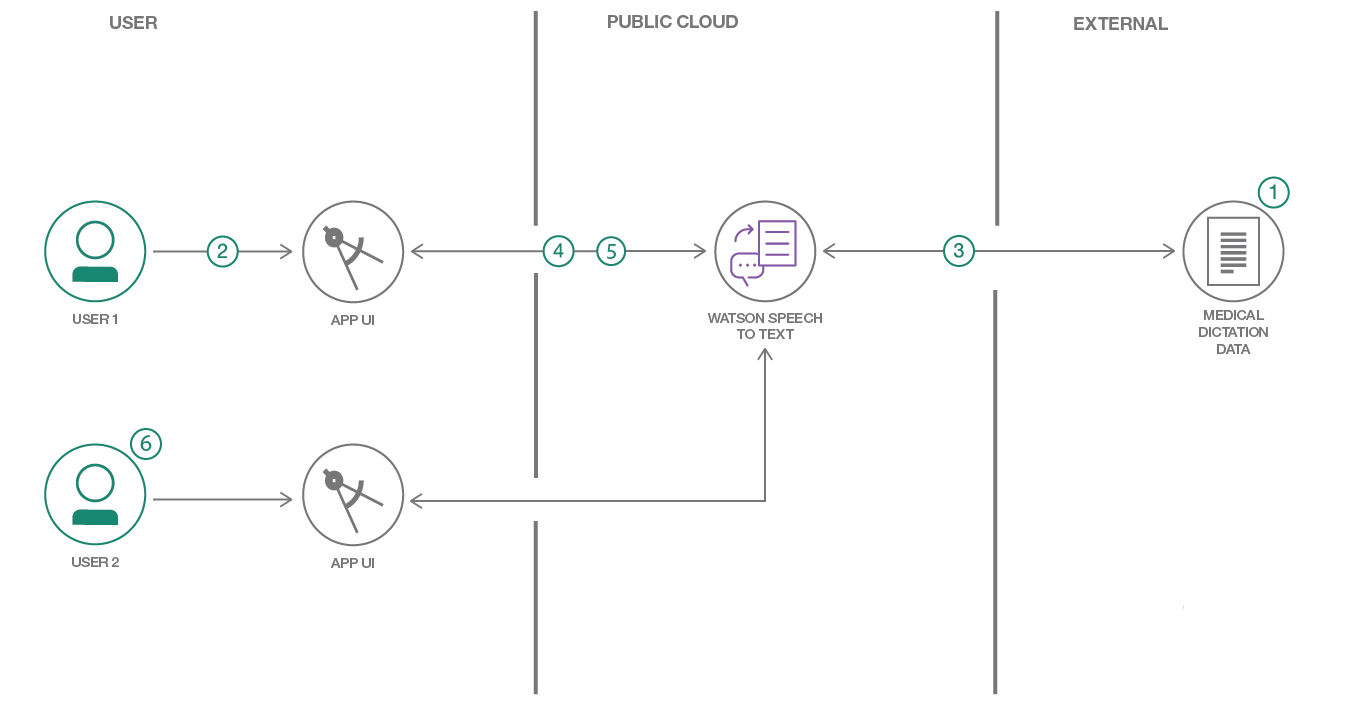
Some speech recognition systems require "training" (also called "enrollment") where an individual speaker reads text or isolated [vocabulary](https://en.wikipedia.org/wiki/Vocabulary) into the system. The system analyzes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. Systems that do not use training are called "speaker independent" systems. Systems that use training are called "speaker dependent".

We will create a custom speech to text model. The Watson Speech to Text service is among the best in the industry. However, like other Cloud speech services, it was trained with general conversational speech for general use; therefore it may not perform well in specialized domains such as medicine, law, sports, etc. To improve the accuracy of the speech-to-text service, you can leverage transfer learning by training the existing AI model with new data from your domain.

In this project, we will use a medical speech data set to illustrate the process. The data is provided by ezDI and includes 16 hours of medical dictation in both audio and text files.

The detailed information of the dataset is given in the link below:

<https://www.ezdi.com/open-datasets/>



**Featured Technologies**

* **Machine Learning**

Machine Learning is a sub-area of artificial intelligence, whereby the term refers to the ability of IT systems to independently find solutions to problems by recognizing patterns in databases. In other words: Machine Learning enables IT systems to recognize patterns on the basis of existing algorithms and data sets and to develop adequate solution concepts. Therefore, in Machine Learning, artificial knowledge is generated on the basis of experience. Following are the types of machine learning:

1. **Unsupervised learning**: In unsupervised learning, artificial intelligence learns without predefined target values and without rewards. It is mainly used for learning segmentation (clustering).
2. **Supervised learning**: In the course of monitored learning, example models are defined in advance. In order to ensure an adequate allocation of the information to the respective model groups of the algorithms, these then have to be specified.
3. **Partially supervised learning**: Partially supervised learning is a combination of supervised and unsupervised learning.
4. **Encouraging learning**: Reinforcing learning is based on rewards and punishments. The algorithm is taught by a positive or negative interaction which reaction to a certain situation should take place.

* **Python 3.0**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python, as a high level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

* **Node.js**

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [runtime environment](https://en.wikipedia.org/wiki/Runtime_system) that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting)—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm unifying [web-application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server- and client-side scripts.

* **React.js**

React (also known as React.js or ReactJS) is a [JavaScript library](https://en.wikipedia.org/wiki/JavaScript_library) for building [user interfaces](https://en.wikipedia.org/wiki/User_interfaces). It is maintained by [Facebook](https://en.wikipedia.org/wiki/Facebook) and a community of individual developers and companies.React can be used as a base in the development of [single-page](https://en.wikipedia.org/wiki/Single-page_application) or mobile applications, as it is optimal for fetching rapidly changing data that needs to be recorded. However, fetching data is only the beginning of what happens on a web page, which is why complex React applications usually require the use of additional libraries for [state management](https://en.wikipedia.org/wiki/State_management), [routing](https://en.wikipedia.org/w/index.php?title=Web_library&action=edit&redlink=1), and interaction with an [API](https://en.wikipedia.org/wiki/API): [Redux](https://en.wikipedia.org/wiki/Redux_(JavaScript_library)), React Router and axiosare examples of such libraries.

* **IBM Watson Speech-to-Text**

IBM Watson Speech-to-Text (STT) — Automated Speech Recognition (ASR) service that converts an audio stream into text for other text-based APIs services. \

* **HTML**

Hypertext Markup Language (**HTML**) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

* **Bootstrap**

Bootstrap is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [CSS framework](https://en.wikipedia.org/wiki/CSS_framework) directed at responsive, mobile-first [front-end web development](https://en.wikipedia.org/wiki/Front-end_web_development). It contains [CSS](https://en.wikipedia.org/wiki/CSS)- and (optionally) [JavaScript](https://en.wikipedia.org/wiki/JavaScript)-based design templates for [typography](https://en.wikipedia.org/wiki/Web_design#Typography), [forms](https://en.wikipedia.org/wiki/Form_(HTML)), [buttons](https://en.wikipedia.org/wiki/Button_(computing)#HTML), [navigation](https://en.wikipedia.org/wiki/Web_navigation#Local_website_navigation) and other interface components.

**Download and prepare the data**

We downloaded both the medical dictation audio files and the transcribed text files from ezDI website. The downloaded files will be contained in zip files.

Create both an Audio and Documents subdirectory inside the data directory and then extract the downloaded zip files into their respective locations.

The transcription files stored in the Documents directory will be in **rtf** format, and need to be converted to plain text. Use the following bash script to convert them all to **txt** files.

If you have Python 3 installed run the following :

python convert\_rtf\_python3.py

The data needs careful preparation since our deep learning model will only be as good as the data used in the training. Preparation may include steps such as removing erroneous words in the text, bad audio recordings, etc. These steps are typically very time-consuming when dealing with large datasets.

Although the dataset from ezDI is already curated, a quick scan of the text transcription files will reveal some filler text that would not help the training. These unwanted text strings have been collected in the file [data/fixup.sed](https://github.com/IBM/Train-Custom-Speech-Model/blob/master/data/fixup.sed) and can be removed from the text files by using the *sed* utility.

Also, for the purpose of training, we will need to combine all text files into a single package, called a corpus file.

To remove the unwanted text strings and to combine all of the text files into a single corpus file, perform the following command:

sed -f fixup.sed Documents/\*.txt > corpus-1.txt

For the audio files, we can archive them as zip or tar files. Since the Watson Speech to Text API has a limit of 100MB per archive file, we will need to split up the audio files into 3 zip files. We will also set aside the first 5 audio files for testing.

zip audio-set1.zip -xi Audio/[6-9].wav Audio/[1-7][0-9].wav

zip audio-set2.zip -xi Audio/[8-9][0-9].wav Audio/1[0-6][0-9].wav

zip audio-set3.zip -xi Audio/1[7-9][0-9].wav Audio/2[0-4][0-9].wav

**Procedure**

1. Clone the repo.

(<https://github.com/IBM/Train-Custom-Speech-Model#4-download-and-prepare-the-data>)

2. Create IBM Cloud services.

Create the following services:

* [**Watson Speech To Text**](https://cloud.ibm.com/catalog/services/speech-to-text)

Note: In order to perform customization, you will need to select the Standard paid plan.

3. Configure credentials.

From your  **Watson Speech to Text** service instance, select the Service Credentials tab.

If no credentials exist, select the New Credential button to create a new set of credentials.

Save off the apikey and url values as they will be needed in future steps.

4. Download and prepare the data.

5. Train the models.

To train the language and acoustic models, you can either run the application or use the command line interface. Or you can mix as desired, since both are working with the same data files and services.

**a. Run the application**

The application is a *nodejs* web service running locally with a GUI implemented in *React*.

* Install [Node.js](https://nodejs.org/en/) runtime or NPM.

To allow the web service to connect to your **Watson Speech to Text** service, create in the root directory a file named services.json by copying the sample file services.sample.json. Update the apikey and 'url' fields in the newly created file with your own values that were retrieved in [Step 3](https://github.com/IBM/Train-Custom-Speech-Model#3-configure-credentials).

{

"services": {

"code-pattern-custom-language-model": [

{

"credentials": {

"apikey": "<your api key>",

"url": "<your api url>"

},

"label": "speech\_to\_text",

"name": "code-pattern-custom-language-model"

}

]

}

}

The application will require a local login. The local user accounts are defined in the file [model/user.json](https://github.com/IBM/Train-Custom-Speech-Model/blob/master/model/user.json). The pre-defined user/passwords are user1/user1 and user2/user2.

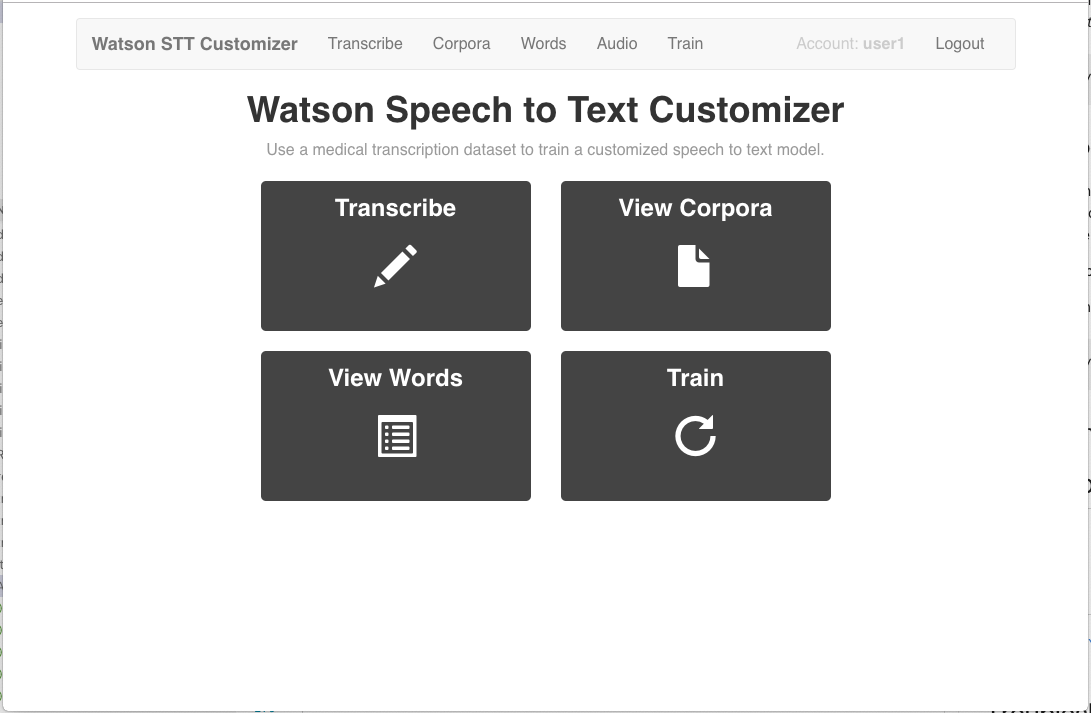
The langModel and acousticModel fields are the names of your custom language and acoustic models which will be created upon logging in if they do not already exist. You can change the baseModel field if the base model you are working with is different from our default.

Install and start the application by running the following commands in the root directory:

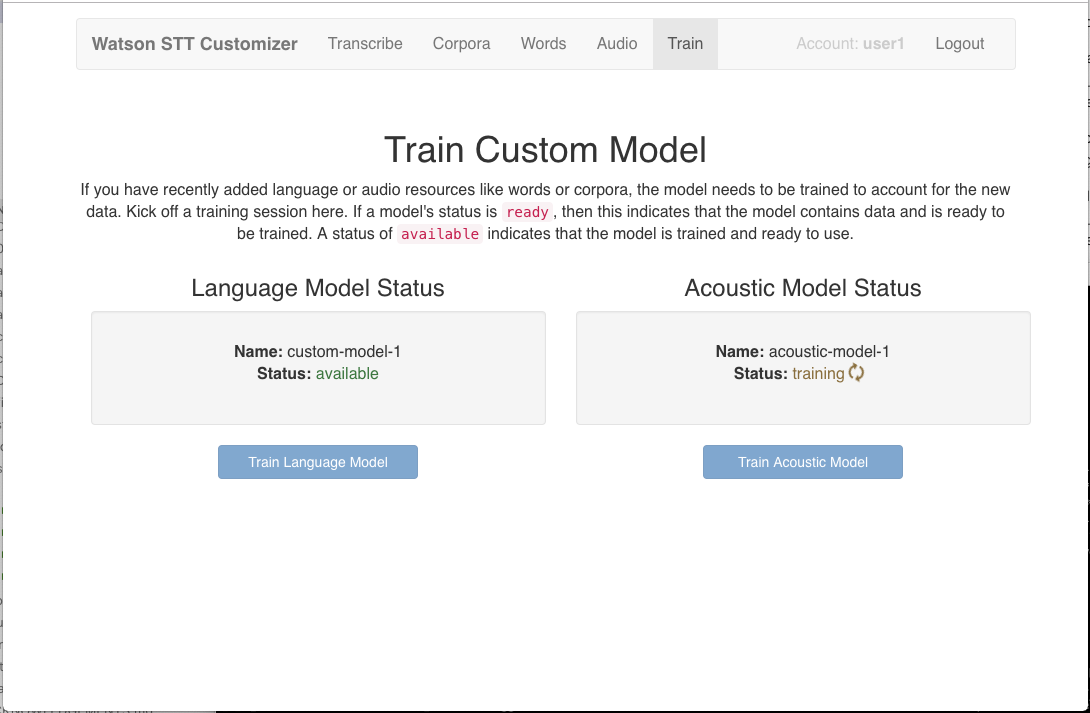
npm install

npm run dev

The local nodejs web server will automatically open your browser to [http://localhost:3000](http://localhost:3000/).



Before training the model, you must add the corpus and audio files. The files can be uploaded using the panels displayed in the Corpora and Audio tabs of the application UI. Then select the Train tab to show the training options. Train both the Language Model and Acoustic Model.



6. Transcribe your dictation

To try out the model, either create your own recorded medical dictation in wav format (use 8KHz sampling rate), or use one of the first 5 test wav files located in /data/Audio (remember, we left those out of the data set used to train the model).

If running the application, click on the Transcribe tab and then browse to your wav file. You can select any combination of base or custom model for language and acoustic. Using custom model

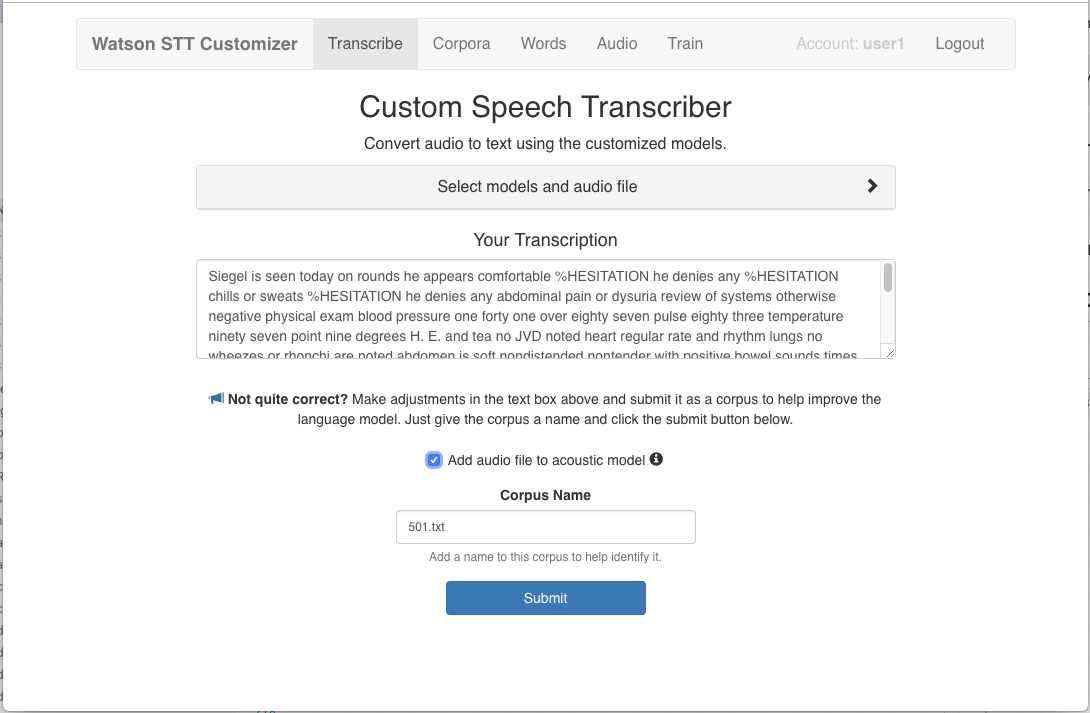
for both should give the best result.

If using the command line, enter the following:

python ../cmd/transcribe.py <my\_dictation.wav>

Similarly to the application, you can set or unset the environment variables LANGUAGE\_ID and ACOUSTIC\_ID to select any combination of base or custom model for language and acoustic. If the corresponding variable is unset, the base model will be used. The transcription will be displayed on the terminal as well as written to a file with the same name as the audio file but with the file extension .transcript.

7. Correct the transcription



If you detect errors in the transcribed text, you can re-train the models by submitting corrected transcriptions. From the Transcribe panel, correct the transribed text.

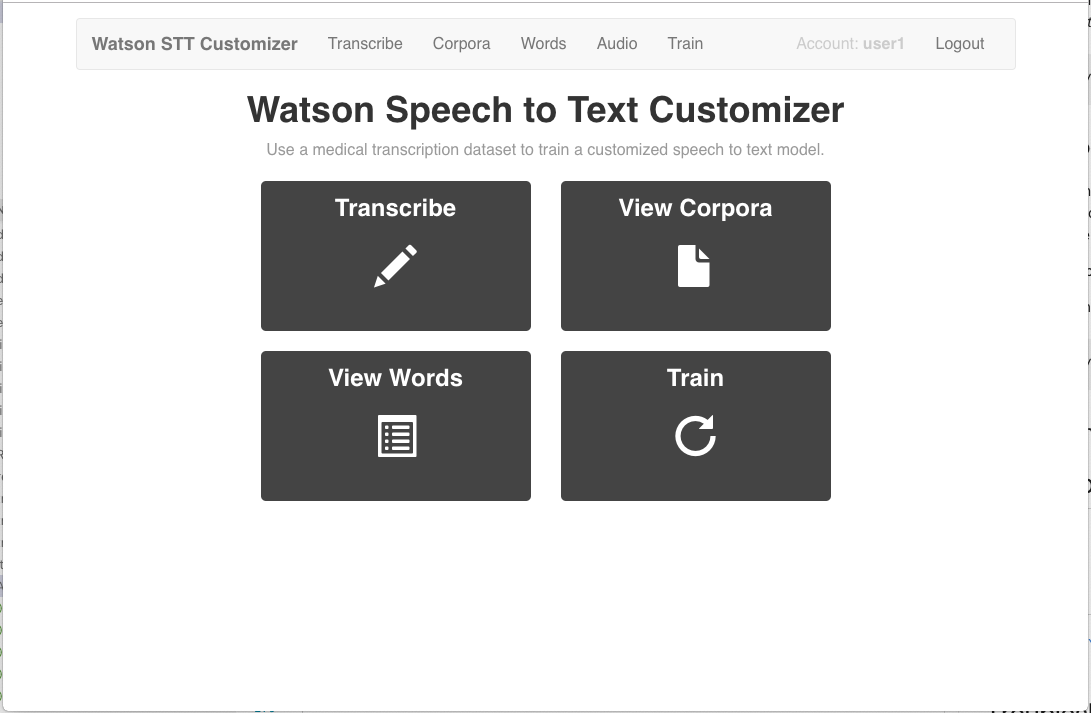
If the audio file being transcribed is not already included in the acoustic model, check the Add audio file to acoustic model checkbox.

Enter a corpus name, and hit Submit.

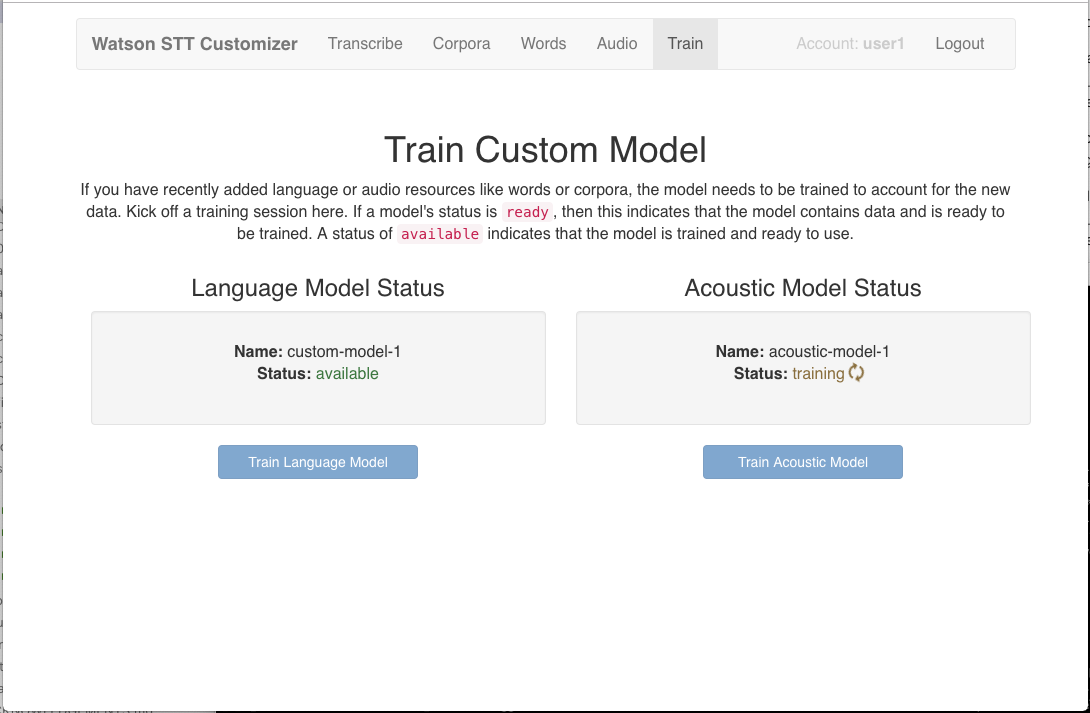
The language and acoustic models will be re-trained with the new files.

**Results**

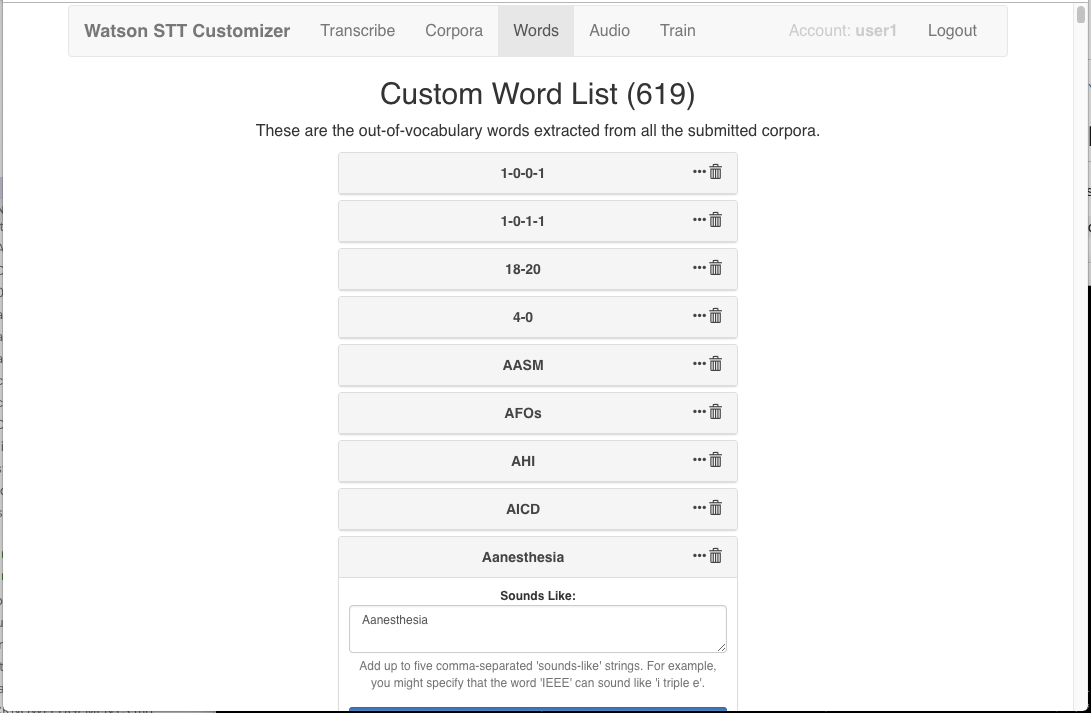
* The main GUI screen:

[](https://github.com/nawedzaman/Custom-speech-to-text-web-application/blob/master/doc/source/images/main-page.png)

* Status for training of the models:

[](https://github.com/nawedzaman/Custom-speech-to-text-web-application/blob/master/doc/source/images/training-panel.png)

* List of "Out of Vocabulary" words determined during training:

[](https://github.com/nawedzaman/Custom-speech-to-text-web-application/blob/master/doc/source/images/custom-word-list.png)

Note: These are the words that are not a part of the base Watson Speech to Text service, but will be added to the language model.

**Future Scope**

1. Accuracy will become better and better
2. Dictation speech recognition will gradually become accepted
3. Microphone and sound systems will be designed to adapt more quickly to changing background noise levels, different environments, with better recognition of extraneous material to be discarded.

**Conclusion**

This Web Application gives us the ability to create custom speech to text service using IBM Watson.

This makes our application:-

* Easy to Transcribe
* Increased Accuracy and
* Self Improvement