Speech to Text Generation with Sentimental Analysis for Physically handicapped people

Yagna Karthik Vaka
CSE
IIIT Sricity
Telangana, India
yagnakarthik.v17@iiits.in

Hemanth Phaneedra Varma
CSE
IIIT Sricity
Andhra Pradesh, India
hemanthphaneedravarma.g17@iiits.in

Sai santosh chirag
CSE
IIIT Sricity
Telangana, India
saisantosh.c17@iiits.in

Abstract: *In present industry, communication is the key element to progress. Passing on information, to the right person, and in the right manner is very important, not just on a corporate level, but also on a personal level. This paper basically gives an overview of major technological perspective and appreciation of the fundamental progress of speech to text conversion, basic process of text translation and also gives complete set of text generator using the long short term memory (LSTM) and concepts of NLP (natural language processing) and finally we explored an approach and performed sentimental analysis for detection of wrong words and implemented to remove those words instead of removing a whole comment or review. We finally built this whole application in django and better purpose for physically handicapped people.*

Keywords: *Speech to text, Language translation, Text generation, Sentimental analysis.*

1 Introduction

Over the past few years, Cell Phones have become an indispensable source of communication for the modern society. We can make calls and text messages from a source to a destination easily. It is known that verbal communication is the most appropriate modem of passing on and conceiving the correct information, avoiding misquotations. To fulfil the gap over a long distance, verbal communication can take place easily on phone calls. A path-breaking innovation has recently come to play in the SMS technology using the speech recognition technology, where voice messages are being converted to text messages. They can also be used for other applications, taking an example: Siri an intelligent automated assistant implemented on an electronic device, to facilitate user interaction with a device, and to help the user more effectively engage with local and/or remote services. The first component of speech recognition is, of course, speech. Speech must be converted from physical sound to an electrical signal with a microphone, and then to digital data with an analog-to-digital converter. Once digitized, several models can be used to transcribe the audio to text. Most mordern speech recognition systems rely on what is known as Hidden Markov Model(HMM). This approach works on the assumption that a speech signal, when viewed on a short enough timescale(say, ten millisecond), can be reasonably approximated as a stationary process– that is, a process in which statistical properties do not change over time.

Translation is a free multilingual neural machine translation service, to translate text and websites from one language into another. An application programming interface that helps developers build software applications. During a translation , it looks for patterns in millions of documents to help you decide on which words to choose and how to to arrange them in the target language.

Generated sentences seems to be quite right, with correct grammar and syntax, as if the neural network was understanding correctly the structure of a sentence.. But the whole next text does not have a great sense and sometimes has complete nonsense. This result would come from the approach itself, using only LSTM to generate text, word by word. In this paper, I will try to investigate a slightly different way to generate sentences in a text generator solution. Here I will create N-grams sequences from the sentences. An advancement of this will be to use single word or every combination words possible from the sentence to predict the next word. And this is lossely termed as n-gram sequences.

Sentiment analysis (or opinion mining) is a natural language processing techniques used to interpret and classify

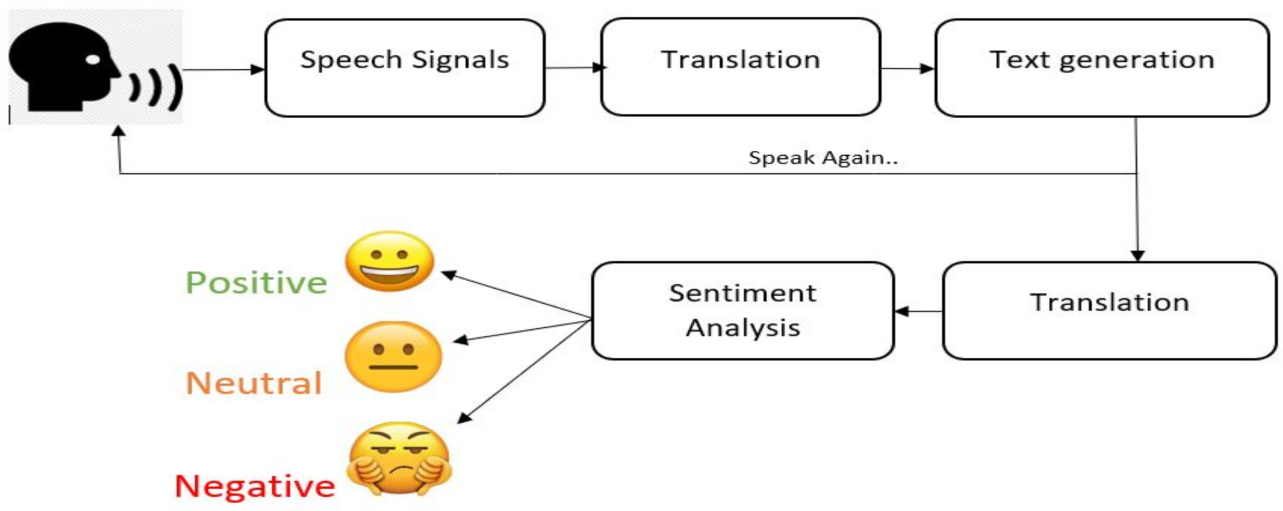

Fig. 1. Workflow of Project

emotions in subjective data. Sentiment analysis is often performed on textual data to detect sentiment in emails, survey responses, social media data and beyond. It focus on polarity (positive, negative, neutral) but also on feelings and emotions(angry, happy, sad etc), urgency(urgent, not urgent) and even intentions. If polarity precision is important to your business, you might consider expanding your polarity categories to include: positive, neutral, negative.

1.1 Motivation

We have motivated this application by seeing Google products and Natural language processing (NLP) teachings from my college. Especially we have inspired and motivated by Natural Language Processing (NLP) teachings from coursera. At the beginning of the project, We worked very slow and we did some research in 8 different research papers regarding speech to text, language translation, text generation and sentimental analysis. From reference [1], I learned some information about their work and got inspired regarding speech to text and language translation. From reference [4], We learned about text generation using Recurrent neural networks (RNN) and referred to some videos related to it. After that we got motivated by reading other research papers and some literature work finally we came up with the solution and this solution is unique idea. I have searched my idea and methodology in many resources but we did not get complete similar idea and this project will really help the people who are handicapped and lazy. Especially some people who are weak in english language writing.

2 State of the Art

From paper ”How accurately can the Google Web Speech API recognize and transcribe Japanese L2 English learners’ oral production?” In this we learned about how google speech API recognition works and procedure. Unfortunately even with the hugely powerful systems that are used widely nowadays on smartphones and PCs, recognition accuracy is not perfect. In this paper they were interested in exploring whether the Google Speech recognition package. So finally we explored some information on internet and how to make use of google speech recognition package in our project.

From paper ”Evaluating Google Speech-to-Text API’s Performance for Romanian e-Learning Resources”, In this paper they spoke about a way of performing Automated speech recognition (ASR) on multimedia e-learning resources available and with the usage of Google Cloud speech-To-Text API. They also did some literature review on applying the Google Cloud Speech-to-Text API on various video e-learning resources available online the Youtube. So we thought of doing something like this in our project.

From paper ”Real Time Speech Translation (Google TV and Chat)”, In this paper they have established a tool that can be used to watch a television or any video with speech and text translation in real time using Google Cloud Translation API. So we thought of taking this procedure and steps to add in our project.

From paper ”Generating Sequences With Recurrent Neural Networks”, In this paper shows how Long Short-term Memory recurrent neural networks can be used to generate complex sentences with long range structure, simply by predicting one data point at a time. Here they have used Long Short-Term Memory recurrent neural networks but we also used bidirectional Long Short Term Memory recurrent neural networks. Let’s take it a step further by allowing that context to flow in both directions. To implement this, we train two Recurrent Neural Networks (RNN) layers simultaneously. The first layer is fed the input sequence as-is and the second is fed a reversed copy.

From paper ”CONTEXT BASED TEXT-GENERATION USING LSTM NETWORKS”, In this paper they have used data pre-processing steps and all the words are vectorized using one-hot encoding, thus each word is represented by a vector of dimension V, where V is the size of vocabulary. They also implemented in Bidirectional LSTM but here in addition to it we also implemented N-grams for storing and to get better word formation. This model is still capable of producing some new text that can make some sense for a reader.

From paper ”Analysis of foul language usage in social media text conversation”, In this paper they have done some research in social networking sites offer today’s teenagers a platform for communication and entertainment. In some cases, the impact of the negative influences of social media on teenage users increases with an increase in the use of offensive language in social conversations. This increase could lead to frustration, depression and a large change in their behaviour. Hence, we propose a novel approach to classify bad language usage in text conversations. We have considered the English language as the medium for textual conversation. We have developed our system based on a foul or swear words language classification approach; it detects offensive language usage in a conversation and removes them from a textual conversation.

3 Proposed System / Methodology

Our system is all about helping handicapped people and lazy people. Here we speak some information using speech recognition API and we take that output text from speech recognition API. That information text will be translated into respective language of user’s choice. That information text will be used for text generation module and will be converted into respective language sentences. Finally if user likes any sentence that sentence will be sent to sentimental analysis module, there it will give some feedback like positive, negative and neutral and also it detects bad words and we implemented an algorithm to highlight them.

3.1 Speech to Text

In a typical HMM, the speech signal is divided into 10-millisecond fragments.To decode the speech into text, groups of vectors are matched to one or more phonemes—a fundamental unit of speech. This calculation requires training, since the sound of a phoneme varies from speaker to speaker, and even varies from one utterance to another by the same speaker. A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes.

One can imagine that this whole process may be computationally expensive. In many modern speech recognition systems, neural networks are used to simplify the speech signal using techniques for feature transformation and dimensionality reduction before HMM recognition. Voice activity detectors (VADs) are also used to reduce an audio signal to only the portions that are likely to contain speech. This prevents the recognizer from wasting time analyzing unnecessary parts of the signal.Fortunately, as a Python programmer, we don’t have to worry about any of this. A number of speech recognition services are available for use online through an API, and many of these services offer Python SDKs.

A handful of packages for speech recognition exist on PyPI. A few of them include:

1. google-cloud-speech
2. Speech Recognition

We are using Speech Recognition library.It acts as a wrapper for several popular speech APIs and is thus extremely flexible. One of these—the Google Web Speech API—supports a default API key that is hard-coded into the Speech Recognition library. That means you can get off your feet without having to sign up for a service. The flexibility and ease-of-use of the Speech Recognition package make it an excellent choice for any Python project.

3.2 Language translation

We are using Cloud Translation API, can dynamically translate text between thousands of language pairs. The Cloud Translation API lets websites and programs integrate with the translation service programmatically. The Cloud Translation API is part of the larger Cloud Machine Learning API family. Translation API instantly translates texts into more than one hundred languages for your website and apps. Translation API advanced offers the same fast, dynamic results you get with basic and additional customization features. Customization matters for domain and context specific terms or phrases.

Benefits of language translation are:-

1. Translate many languages
2. Language detection
3. Highly scalable
4. Simple, affordable pricing

3.2.1 Translate many languages

Translation API’s pre-trained model supports more than one hundred languages.

3.2.2 Language detection

When you don’t know your source text language – for instance, in user-generated content that doesn’t include a language code – our translation products automatically identify languages with high accuracy.

3.2.3 Highly scalable

We provide a generous daily quota and let you set lower limits. And you can use batch translation with Google Cloud Storage to reduce the workflow complexity of translating long or multiple text files.

3.3 Text generation

We have implemented the process, text generation using LSTM. There are two types of text generation:

1. Character based text generation
2. Word based text generation

Character based text generation is each character of the text is used to train the model and prediction will also result in generation of new characters. Word based text generation is Words are converted into tokens which are used to train the model. The model will generate words instead of characters in the prediction stage.

Mechanics of the text generation model:

1. The next word of the sequence is predicted using the words that are already present in the sequence.
2. It is a simple model where splitting the data into training and testing sets is not required. This is because the model will use all the words in the sequence to predict the next word. Just like forecasting.

The flow of program:

1. Loading data.
2. Preprocessing the data and Tokenizing.
3. Building and fitting the model on data.
4. Evaluate the model.
5. Predicting(Generating the text)
6. Saving the model for future applications

3.3.1 Load and Examine data

We loaded our data from a local file and then we ran a word count, we saw that vocubulary for dataset is quite small. This was by design of this project. This allows us train the models in reasonable time.

3.3.2 Cleaning process

We performed our data by replacing the useless text with space. We converted the whole data into lower case to get uniformity. Splitting the data to get every line seperately but this will give the list of uncleaned data. We also removed HTML tags, stop words, punctuation markings.

3.3.3 Tokenization

We need to tokenize the data – i.e., convert the text to numerical values. This allows the neural network to perform operations on the input data. For this project, each word and punctuation mark will be given a unique ID. When we run the tokenizer, it creates a word index, which is then used to convert each sentence to a vector.

3.3.4 Creating n-gram sequences

The main idea of generating text using N-grams is to assume that the last word of the n-gram can be inferred from the other words that appear in the same n-gram, which I call context. This is the basic concept of forecasting which can be applied here to generate text. An advancement of this will be to use single word or every combination words possible from the sentence to predict the next word. And this is loosely termed as n-gram sequences. So we will use combinations of words to make our model better. So we will turn the sentences to sequences line by line and create n-gram sequences.

3.3.5 Padding

When we feed our sequences of word IDs into model, each sequence needs to be the same length. To achieve this, padding is added to my sequence that is shorter than the max length(i.e. shorter than the longest sentence).

3.3.6 Training the model

Taking non-label(xs) and labels(y) to train the model. xs contains every word in sentence except the last one because we are using this value to predict the y value labels contains only the last word of the sentence which will help in hot encoding the y value in next step

3.3.7 One-Hot Encoding

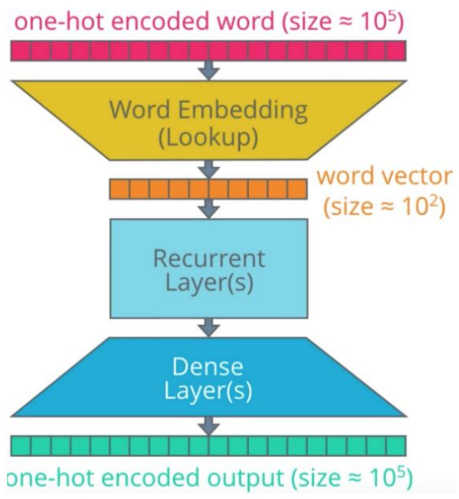
one-hot encoded word (size ≈ 10^5)
Word Embedding (Lookup)
word vector (size ≈ 10^2)
Recurrent Layer(s)
Dense Layer(s)
one-hot encoded output (size ≈ 10^5)

Fig. 2. Architecture of an RNN at a high level

One-hot encoding the labels according to the vocabulary size. The matrix is square matrix of the size of vocabulary size. Each row will denote a label and it will have a single positive value(i.e 1) for that label and other values will be zero.

One of the advantages of One-hot encoding (OHE) is efficiency since it can run at as faster clock rate than other encodings. The other advantage is that One-hot encoding (OHE) better represents categorical data where there is no ordinal relationship between different values. For example, let's say we're classifying animals as either a mammal, repitle, fish or bird. If we encode them as 1,2,3,4 respectively, our model may assume there is a natural ordering between them, which there isn't. It's not useful to structure our data such that mammal comes before reptile and so forth. This can mislead our model and cause poor results. However, if we then apply one-hot encoding to these integers, changing them to binary representations — 1000, 0100, 0010, 0001 respectively — then no ordinal relationship can be inferred by the model.

But, one of the drawbacks of OHE is that the vectors can get very long and sparse. The length of the vector is determined by the vocabulary, i.e. the number of unique words in your text corpus.

3.3.8 Modeling

First, let's breakdown the architecture of an RNN at a high level. Referring to the digram above, there are a few parts of the model to be aware of:

1. **Inputs:** Input sequences are fed into the model with one word for every time step. Each word is encoded as a unique integer or one-hot encoded vector that maps to the English dataset vocabulary
2. **Embedding Layers:** Embeddings are used to convert each word to a vector. The size of the vector depends on the complexity of the vocabulary.
3. **Recurrent Layers (Encoder):** This is where the context from word vectors in previous time steps is applied to the current word vector.
4. **Dense Layers (Decoder):** These are typical fully connected layers used to decode the encoded input into correct sequence.
5. **Outputs:** The outputs are returned as a sequence of integers or OHE vectors which can be mapped to vocabulary.

Bidirectional Layer Until now we understood how context flows through the network via the hidden state, let's take it a step further by allowing that context to flow in both directions. This is what a bidirectional layer does. Until now the encoder only has historical context. But, providing future context can result in better model performance. This may seem counter intuitive to the way humans process language since we only read in one direction. However, humans often require future context to interpret what is being said. In other words, sometimes we don't understand a sentence until an important word or phrase is provided at the end.

To implement this, we train two Recurrent Neural Networks (RNN) layers simultaneously. The first layer is fed the input sequence as-is and the second is fed a reversed copy.

3.3.9 Predicting

In this we defined a function to take input of seed text from user and number of words to be predicted. We tokenize the text into sequences, after that we pad the sequences. Then predict the before sequences and store into a variable. Finally we predict the word with previous word and generate the text.

3.3.10 Saving the model

After training the model, we get accuracy of almost 94% to 97% for different types of story books like Alice and it's Adventure, Harry potter and Twilight. We saved those models in HDF5 files without training again and again. It is better at computational time so that we can test the model using the saved HDF5 files. As it saves a lot of time.

3.4 Sentimental Analysis

Sentimental analysis is the process of detecting positive or negative sentiment in text. It's often used in businesses to detect sentiment in social data and customers. Since speaker express their thoughts and feelings more openly than ever before, sentimental analysis is becoming an essential tool to monitor and understand that sentiment. Automatically analyzing speaker feedback, such as opinions in survey responses and social media conversations, allows brands to learn what makes others happy or frustrated. For example, using our sentimental analysis to automatically analyze the review or message from a speaker we automatically detect it and if that review is very unhappy then we will just remove swear words from that review. So that both the speaker and viewer can be happy instead of deleting whole review.

How can we improve the accuracy of algorithm ? Usually, input data is presented in natural form, which is in format of text,sentences and paragraphs etc. Before such input can be passed to machine learning algorithms, it needs some clean-up or pre-processing so algorithms can focus on main words instead of words which adds minimal to no value. So pre-processing steps are :

1. Removing HTML tags
2. Removing special characters
3. Removing Stop words
4. Lemmatization
5. normalization
6. Removing swear words

3.4.1 Removing HTML tags

Often, unstructured text contains a lot of noise, especially if you use techniques like web or screen scraping. HTML tags are typically one of these components which don't add much value towards understanding and analysing text so they should be removed. We will use BeautifulSoup library for HTML tag clean up

3.4.2 Removing special characters

Special characters, as you know, are non-alphanumeric characters. These characters are most often found in comments, references, currency numbers etc. These characters add no value to text-understanding and induce noise into algorithms. Thankfully, regular-expressions (regex) can be used to get rid of these characters and numbers.

3.4.3 Removing Stop words

Stopwords are often added to sentences to make them grammatically correct, for example, words such as a, is, an, the, and etc. These stopwords carry minimal to no importance and are available plenty on open texts, articles, comments etc. These should be removed so machine learning algorithms can better focus on words which define the meaning/idea of the text. We are using list from nltk.corpus and this list can further be enhanced by adding or removing custom words based on the situation at hand.

3.4.4 Lemmatization

Though stemming and lemmatization both generate the root form of inflected/desired words, but lemmatization is an advanced form of stemming. Stemming might not result in actual word, whereas lemmatization does conversion properly with the use of vocabulary, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. Before using lemmatization, we should be aware that it is considerably slower than stemming, so performance should be kept in mind before choosing stemming or lemmatization.

3.4.5 Normalization

Normalization is the process of tansforming text into a single canonical form by process of adding input text to process of removed special characters function and this output is added to the process of removed stop words as an input and then that output is added to input of lemmatization. Finally lemmatization output corpus will be stored into a single form. That form will be used for future purposes of sentimental analysis and also used for removing the swear words.

3.4.6 Removing Swear words

Here we used profanity detection algorithm and discover the swear words in our message or review. This algorithm is based on around 340 words from noswearing.com, which does a basic string match to catch swear words and it also uses a SVM model trained on 200k human labeled samples of clean and profane text strings. Why it works like this? One simplified way you could think about why profanity-check works, during the training process, the model learns which words are ”bad” and how ”bad” they are because those words will appear more often in offensive messages. Thus it’s as if the training process is picking out the ”bad” words out of all possible words and using those to make future predictions. this better than just relying on arbitrary word blacklists chosen by humans!

4 Results

We show the text generation module training results of different story books because we took text from those story books as an input for dataset like

1. Alice’s Adventures in Wonderland
2. Harry Potter
3. Twilight

Fig 1. Training results of Alice’s adventure story book

Epoch	Loss	Accuracy
1/56	5.5106	0.0336
2/56	5.2189	0.0417
3/56	5.1652	0.0485
4/56	5.1135	0.0485
5/56	5.0819	0.0350
./56	....	....
./56	....	....
./56	....	....
./56	....	....
52/56	0.3144	0.9219
53/56	0.3654	0.9058
54/56	0.2983	0.9179
55/56	0.2260	0.9448
56/56	0.1906	0.9562

Fig 2. Training results of Harry Potter story book

Epoch	Loss	Accuracy
1/60	6.2306	0.0236
2/60	6.2189	0.0617
3/60	6.0252	0.0585
4/60	6.0135	0.0605
5/60	5.6819	0.0850
./60	....	....
./60	....	....
./60	....	....
./60	....	....
./60	....	....
./60	....	....
56/60	0.3344	0.9419
57/60	0.3734	0.9228
58/60	0.2763	0.9379
59/60	0.2140	0.9588
60/60	0.1406	0.9682

Fig 3. Training results of Twilight story book

Epoch	Loss	Accuracy
1/57	3.2306	0.0636
2/57	3.2189	0.0817
3/57	3.0252	0.0985
4/57	3.0135	0.1205
5/57	2.6819	0.1250
./57	....	....
./57	....	....
./57	....	....
./57	....	....
53/57	0.2324	0.9319
54/57	0.2724	0.9236
55/57	0.2561	0.9423
56/57	0.1540	0.9573
57/57	0.1396	0.9632

4.1 Django Implementation results

We implemented the above methodology and procedure in django and we have taken some pictures of our project web pages. Below are some pictures of our project for better understanding.

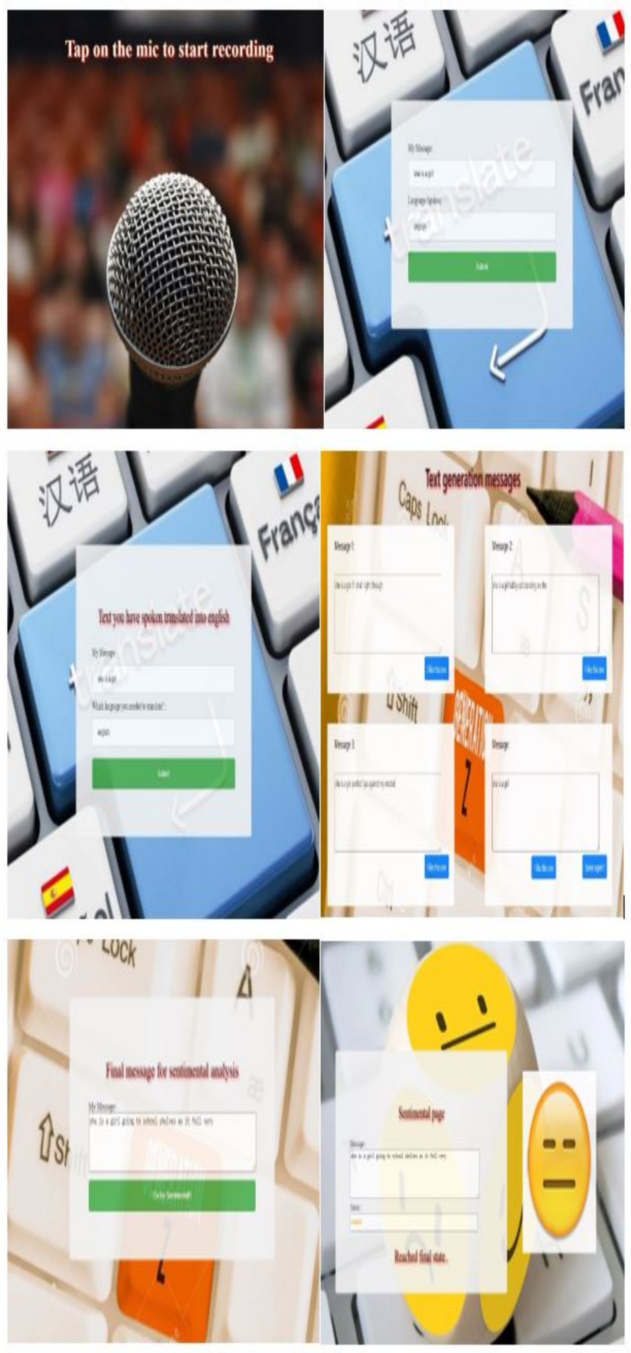
Tap on the mic to start recording
Text you have spoken translated into english
Text generation messages
Final message for sentimental analysis
Sentimental page
Reached final state.

Fig. 3. **(a)** Here in this page, where speaker speaks **(b)** After speaker speaks it gets converted to text and in which language the speaker has spoken will be displayed **(c)** It asks user to which language to be translated **(d)** After asking language it will generate the three texts in those languages **(e)** If user likes any message/text then that will be displayed **(f)** Finally the liked message/text will have sentiment analysis and detect swear words

5 Conclusions

The conclusions that we can draw from this project are:

1. We can speak in any language we want.
2. And the text can be translated to any language of our choice.
3. This model can complete an incomplete sentence we have spoken.
4. After the sentence is completed , we can get to choose one from three complete sentences.
5. We can tell whether the spoken text is positive or negative or neutral.
6. This model will censor the extremely bad words.

Finally this project will really help physically handicapped and lazy people. If this project is **implemented in Android apps** then it would be the best way to reach to people so that they can write messages and correct their messages without typing the keyboard more and without having some foul text in that message/review.

6 References

[1] Ayushi Trivedi,Navya Pant, Pinal Shah,Simran Sonik and Supriya Agrawal, *"Speech to text and text to speech recognition systems-Areview",*. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 20, Issue 2, Ver. I (Mar.- Apr. 2018), PP 36-43

[2] Su Myat Mon, Hla Myo Tun, *"Speech-To-Text Conversion (STT) System Using Hidden Markov Model (HMM)",*. INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH VOLUME 4, ISSUE 06, JUNE 2015, ISSN 2277-8616

[3] Prachi Khilari1, Prof. Bhope V. P, *"Implementation of Speech to Text Conversion",*. International Journal of Innovative Research in Science, Engineering and Technology (An ISO 3297: 2007 Certified Organization) Vol. 4, Issue 7, July 2015, ISSN(Online) : 2319-8753, ISSN (Print) : 2347-6710

[4] Alex Graves, *"Generating Sequences With Recurrent Neural Networks",*. arXiv:1308.0850v5 [cs.NE] 5 Jun 2014

[5] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau, *"Sentiment Analysis of Twitter Data",*. Department of Computer Science Columbia University New York, NY 10027 USA

[6] Saif, Hassan; He, Yulan and Alani, Harith (2012). *"Semantic sentiment analysis of twitter",*. In: The 11th International Semantic Web Conference (ISWC 2012), 11-15 Nov 2012, Boston, MA, USA.

[7] Mikhail Bautin, Lohit Vijayarenu, Steven Skiena, *"International Sentiment Analysis for News and Blogs",*. Stony Brook University, Stony Brook, NY 11794-4400

[8] David Mcdonald, Kathleen McKeown, *"Text Generation",*. published article on June 2002.

[9] Sivasurya Santhanam, *"CONTEXT BASED TEXT-*

GENERATION USING LSTM NETWORKS",. arXiv:2005.00048v1 [cs.CL] 30 Apr 2020

[10] Tim Ashwell, Jesse R. Elam, *"How accurately can the Google Web Speech API recognize and transcribe Japanese L2 English learners' oral production?",*. issn 1832-4215 Vol. 13, No.1 Pages 59–76, ©2017 jalt call sig

[11] Bogdan IANCU, *"Evaluating Google Speech-to-Text API's Performance for Romanian e-Learning Resources",*. Informatica Economică vol. 23, no. 1/2019, The Bucharest University of Economic Studies, Romania

[12] Gamal Bohouta, Veton Z Këpuska, *"Real Time Speech Translation (Google TV and Chat)",*. published on January 2018