Chatbots and Voice Assistant mean for Customer Service

2018102151
Abdullaev Akhidjon

Background Study

Customer service is an essential element of successful businesses especially in the banking, insurance, healthcare, and retail industries. Today, artificial intelligence software and applications like multilingual chatbots and voice assistants are revamping customer service allowing businesses and brands to better serve their customers.

According to a recent report by the Capgemini Research Institute, about 70% of people will gradually use voice assistants and customer service chatbots instead of visiting a store or local bank location.

What is driving this change is the continued improvement in these AI-based applications as well as growing numbers of brands and customers who are embracing this technology.

The Quality of Chatbots and Voice Assistants

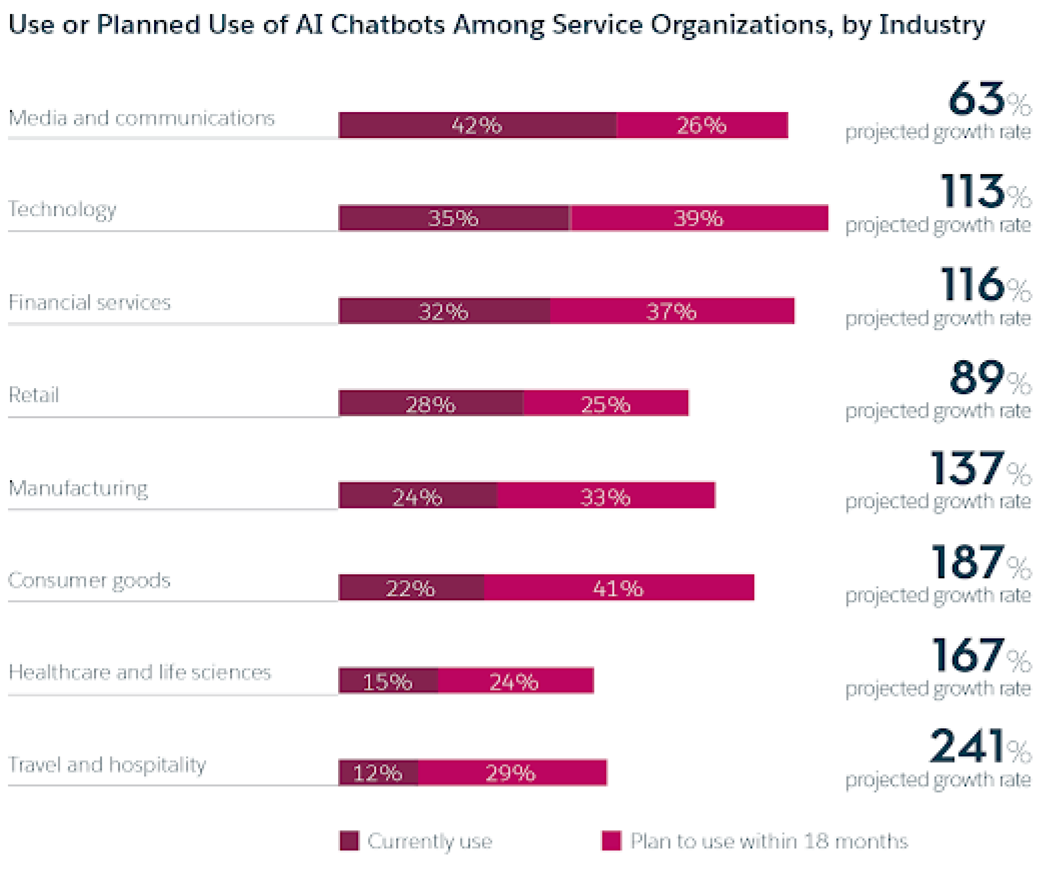
The quality of chatbots and voice assistants has come a long way. These improvements have not gone unnoticed as more and more consumers are willing to engage with chatbots and voice assistants.

In 2019, 72% of people who use voice assistants like Apple's Siri or Amazon's Alexa, for example, were satisfied with their interactions, compared to just 61% back in 2017. Furthermore, customer satisfaction among user of voice assistants via smart speakers has also increased from 46% in 2017 to 64% in 2019.

Willingness to Use Chatbots & Voice Assistants

Businesses across a variety of industries are positioning themselves to take advantage of voice-based assistants and chatbots. The use of chatbots is expected to grow by 116% in the financial services industry, 241% in the hospitality and tourism industry, and 167% in the healthcare industry.

Today, consumers have a fairly positive view of this AI-based technology and, as such, are willing to use chatbots and voice-based assistants. According to a survey by Gartner, nearly 80% of consumers see chatbots as providing a higher level of customer service care.



"State of Service," Salesforce Research, March 2019.

Surveyed consumers also revealed that their willingness to use chatbots is based on the ability to interact with chatbots 24/7 (48%), not having to wait to speak to a customer service representative on the telephone (46%), and having their questions answered faster (37%).

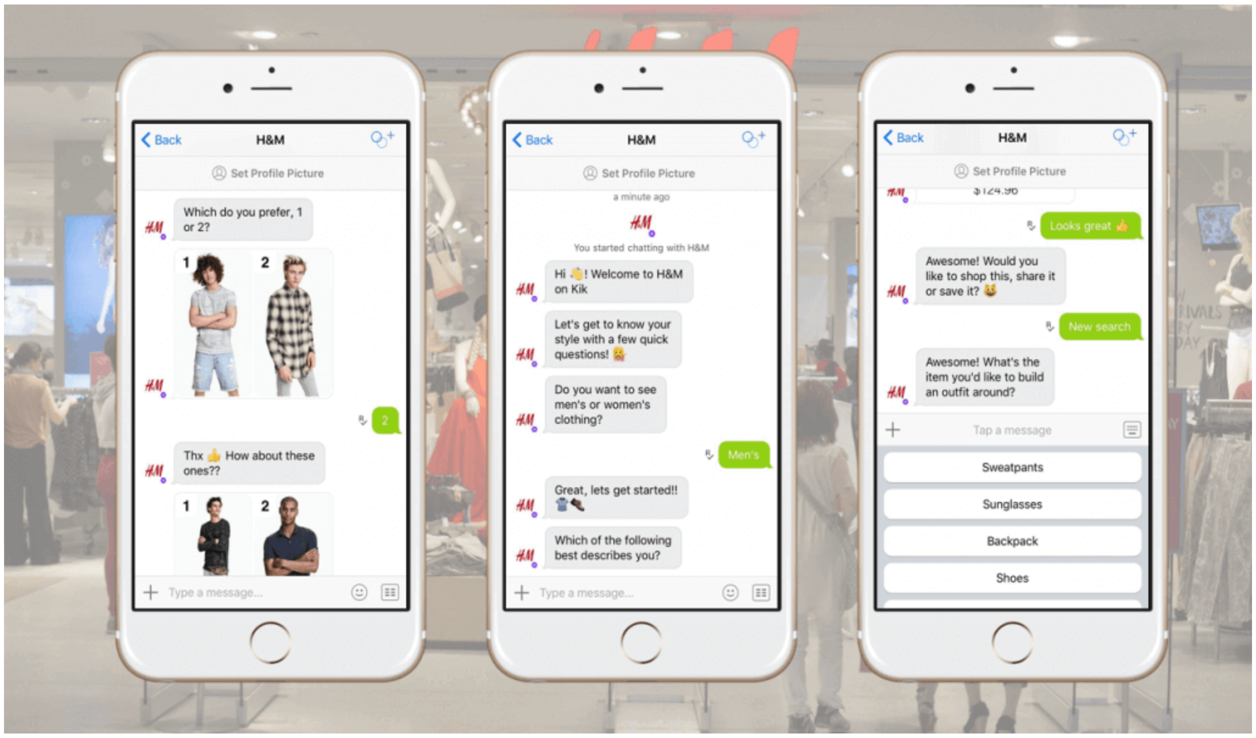
The Impact of Customer Service Chatbots & Voice Assistants

Customer service chatbots and voice assistants allow businesses to serve customers around the clock. The ability to provide consumers with important information quickly or handle straight forward requests in a shorter amount of time are also major benefits.

Customer service chatbots and voice assistants also improve the overall efficiency of customer service teams while also reducing costs. In the banking industry, for example, 90% of customer interactions can be automated using chatbots.

When chatbots and voice assistants can manage a number of routine tasks, brands can use their time more wisely directing their workers to more complex or value-added tasks. In the end, this will enable banks to run more efficiently and reduce operating costs in the process.

In retail, AI-based chatbots can also improve the marketing and sales channels by providing consumers with personalized interactions. Voice assistants, in particular, will play a key role in the retail industry. According to Juniper, voice commerce will continue to grow, eventually reaching more than $80 billion per year by 2023.



Which do you prefer, 1 or 2?

2

Thx 👍 How about these ones??

a minute ago

You started chatting with H&M

Hi 👋! Welcome to H&M on Kik

Let's get to know your style with a few quick questions! 😺

Do you want to see men's or women's clothing?

Men's

Great, lets get started!! 👕👞

Which of the following best describes you?

$124.96

Looks great 👍

Awesome! Would you like to shop this, share it or save it? 😺

New search

Awesome! What's the item you'd like to build an outfit around?

Across all industries, AI-based chatbots are expected to reduce customer service costs by 30%.
Multilingual chatbots and voice assistants also provide businesses with the resource they need to serve diverse markets or customer bases where more than one language is spoken. Airline companies in particular stand to benefit from multilingual chatbots or voice assistants since they often serve international markets.
Despite the benefits of this technology, it would be irresponsible not to note that many studies and surveys do highlight some of the limitations of chatbot and voice-based assistants.

The power of Chatbots!

To summarize here are some of the awesome benefits of deploying chatbots as part of your digital strategy.

- **Availability**
  Chatbots can be operational and ready to engage with their define target group 24 hours a day, 365 days a year.
- **Chatbots are excellent tools to process a large volume of requests:**
  If a particular company receives many requests, it is not necessary to increase the template or team capacity for receiving queries. A well built chatbots allows the brand to face all the questions in a simultaneous manner.
- **It allows companies to better know and understand their users:**
  Companies can get real insight into what they're costumers prefer, and therefore provide improved offers and answers to their users. There's lots to be learned from most searched and used words in regards to what interests the users most.
- **Low maintenance costs:**
  Chatbots allow companies to save money and are easy to configure to meet different needs (depending on the application used). Once they have been built the costs of keeping them updated are relatively low.
- **Costumer Service Improvement through Analytics and Data collection:**
  Chatbots can record data, trends and metrics to subsequently monitor interactions and adjust their processes and responses accordingly.
- **Platform Agnostic:**
  As mentioned before, a great strength of chatbots is that they allow for companies to meet their costumers in whichever platform they're native to. Chatbots allow to be built in different platforms and are easy to adapt and integrate to whatever business as they also allow for specific installations.
- **Time saving**
  A very common and great example of how Chatbots can help companies save time is in relation to how they can help companies by answering their clients FAQ. The efficiency of chatbots derives from the ease of interaction they create for employees, customers or other users. For example, Amigo an event bot created for the Komfo Summit in Copenhagen was used as an event guide.

Goal/Problem & Requirements

Goal

Creating Chatbot with STT and TTS. It should Recognize speech and convert it to text and pass it to a chatbot model and convert output to speech.

Approach and Project Architecture

Seq2Seq Model

The brains of my chatbot is a sequence-to-sequence (seq2seq) model. The goal of a seq2seq model is to take a variable-length sequence as an input, and return a variable-length sequence as an output using a fixed-sized model.
Sutskever et al. discovered that by using two separate recurrent neural nets together, we can accomplish this task. One RNN acts as an **encoder**, which encodes a variable length input sequence to a fixed-length context vector. In theory, this context vector (the final hidden layer of the RNN) will contain semantic information about the query sentence that is input to the bot. The second RNN is a **decoder**, which takes an input word and the context vector, and returns a guess for the next word in the sequence and a hidden state to use in the next iteration.

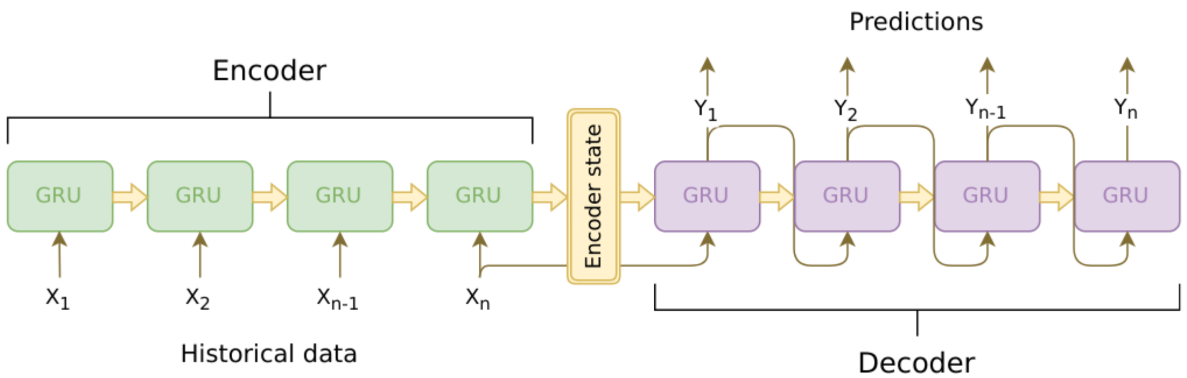


Image source: https://jeddy92.github.io/JEddy92.github.io/ts_seq2seq_intro/

Encoder

The encoder RNN iterates through the input sentence one token (e.g. word) at a time, at each time step outputting an "output" vector and a "hidden state" vector. The hidden state vector is then passed to the next time step, while the output vector is recorded. The encoder transforms the context it saw at each point in the sequence into a set of points in a high-dimensional space, which the decoder will use to generate a meaningful output for the given task.

At the heart of our encoder is a multi-layered Gated Recurrent Unit, invented by Cho et al. in 2014. We will use a bidirectional variant of the GRU, meaning that there are essentially two independent RNNs: one that is fed the input sequence in normal sequential order, and one that is fed the input sequence in reverse order. The outputs of each network are summed at each time step. Using a bidirectional GRU will give us the advantage of encoding both past and future contexts.

Bidirectional RNN:

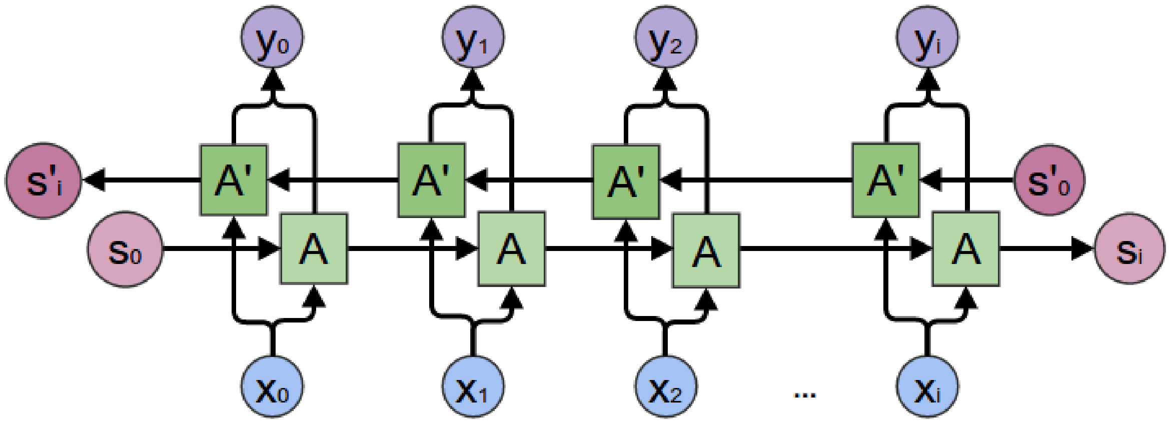


Image source: https://colah.github.io/posts/2015-09-NN-Types-FP/

Note that an `embedding` layer is used to encode our word indices in an arbitrarily sized feature space. For our models, this layer will map each word to a feature space of size *hidden_size*. When trained, these values should encode semantic similarity between similar meaning words.

Finally, if passing a padded batch of sequences to an RNN module, we must pack and unpack padding around the RNN pass using `nn.utils.rnn.pack_padded_sequence` and `nn.utils.rnn.pad_packed_sequence` respectively.

Computation Graph:

1. Convert word indexes to embeddings.
2. Pack padded batch of sequences for RNN module.
3. Forward pass through GRU.
4. Unpack padding.
5. Sum bidirectional GRU outputs.
6. Return output and final hidden state.

Inputs:

- input_seq: batch of input sentences; shape=*(max_length, batch_size)*
- input_lengths: list of sentence lengths corresponding to each sentence in the batch; shape=*(batch_size)*
- hidden: hidden state; shape=*(n_layers x num_directions, batch_size, hidden_size)*

Outputs:

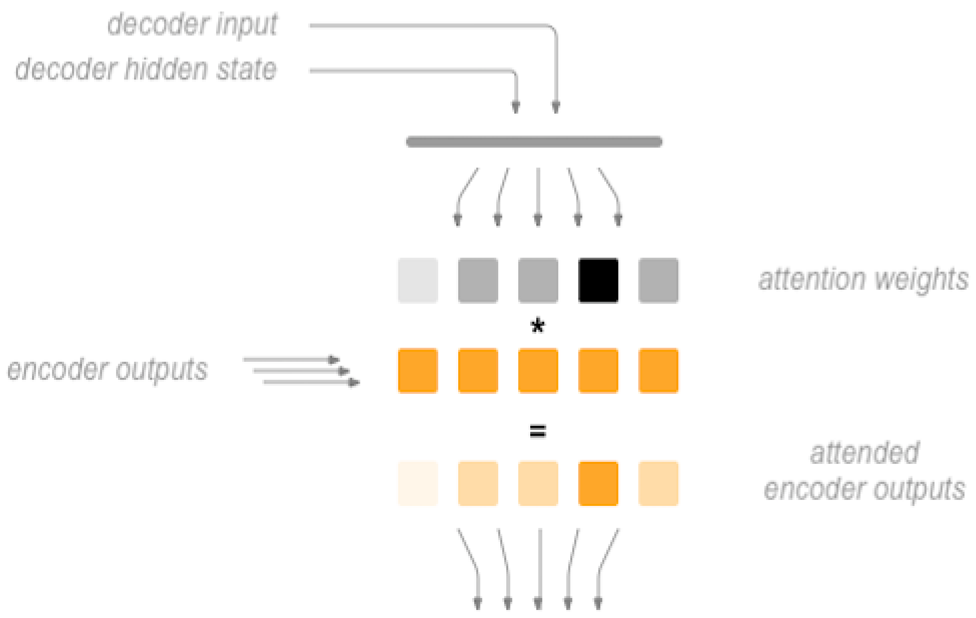
- outputs: output features from the last hidden layer of the GRU (sum of bidirectional outputs); shape=*(max_length, batch_size, hidden_size)*
- hidden: updated hidden state from GRU; shape=*(n_layers x num_directions, batch_size, hidden_size)*

Decoder

The decoder RNN generates the response sentence in a token-by-token fashion. It uses the encoder's context vectors, and internal hidden states to generate the next word in the sequence. It continues generating words until it outputs an *EOS_token*, representing the end of the sentence. A common problem with a vanilla seq2seq decoder is that if we rely solely on the context vector to encode the entire input sequence's meaning, it is likely that we will have information loss. This is especially the case when dealing with long input sequences, greatly limiting the capability of our decoder.

To combat this, Bahdanau et al. created an "attention mechanism" that allows the decoder to pay attention to certain parts of the input sequence, rather than using the entire fixed context at every step.

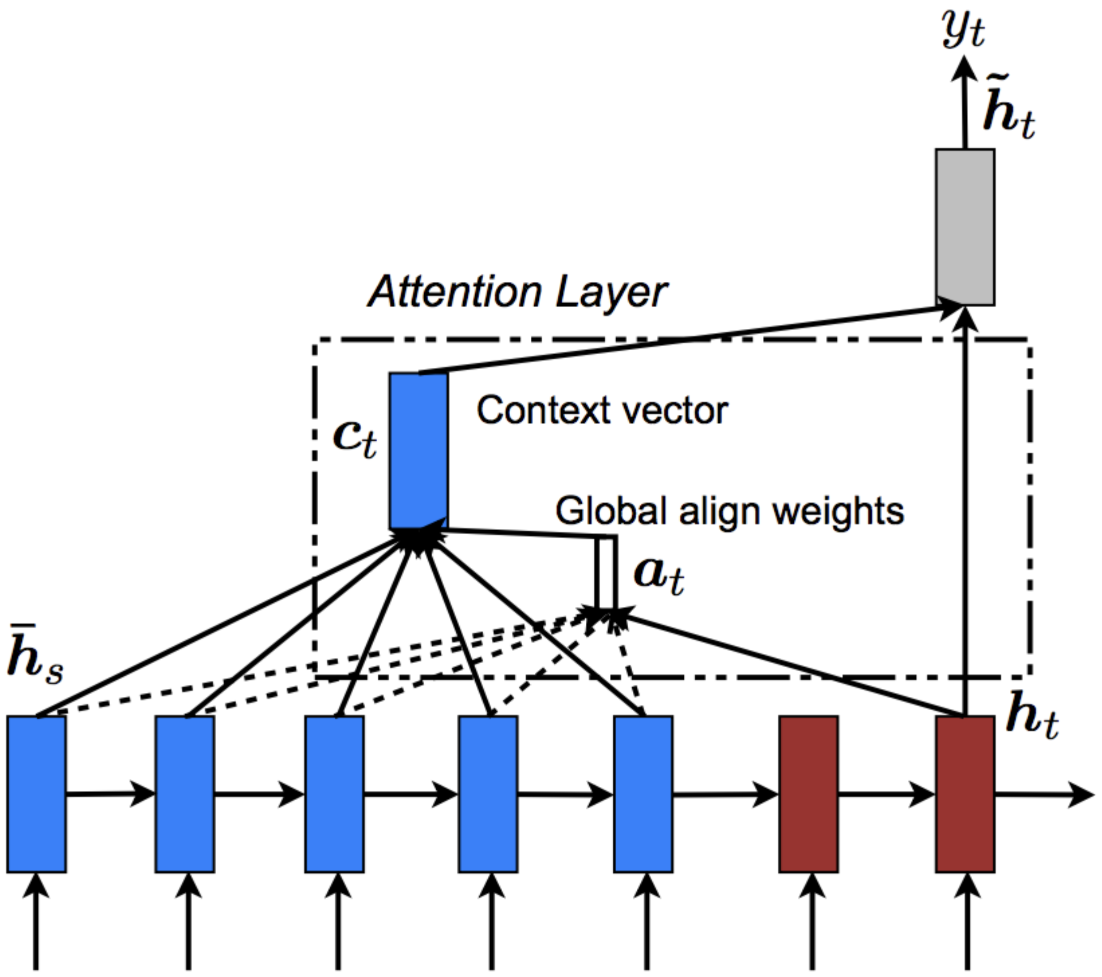
At a high level, attention is calculated using the decoder's current hidden state and the encoder's outputs. The output attention weights have the same shape as the input sequence, allowing us to multiply them by the encoder outputs, giving us a weighted sum which indicates the parts of encoder output to pay attention to. Sean Robertson's figure describes this very well:



Luong et al. improved upon Bahdanau et al.'s groundwork by creating "Global attention". The key difference is that with "Global attention", we consider all of the encoder's hidden states, as opposed to Bahdanau et al.'s "Local attention", which only considers the encoder's hidden state from the current time step. Another difference is that with "Global attention", we calculate attention weights, or energies, using the hidden state of the decoder from the current time step only. Bahdanau et al.'s attention calculation requires knowledge of the decoder's state from the previous time step. Also, Luong et al. provides various methods to calculate the attention energies between the encoder output and decoder output which are called "score functions":

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Overall, the Global attention mechanism can be summarized by the following figure. Note that we will implement the “Attention Layer” as a separate `nn.Module` called `Attn`. The output of this module is a softmax normalized weights tensor of shape *(batch_size, 1, max_length)*.



Now that we have defined our attention submodule, we can implement the actual decoder model. For the decoder, we will manually feed our batch one time step at a time. This means that our embedded word tensor and GRU output will both have shape *(1, batch_size, hidden_size)*.

Computation Graph:

1. Get embedding of current input word.
2. Forward through unidirectional GRU.
3. Calculate attention weights from the current GRU output from (2).

4. Multiply attention weights to encoder outputs to get new “weighted sum” context vector.
5. Concatenate weighted context vector and GRU output using Luong eq. 5.
6. Predict next word using Luong eq. 6 (without softmax).
7. Return output and final hidden state.

Inputs:

- `input_step`: one time step (one word) of input sequence batch; shape=*(1, batch_size)*
- `last_hidden`: final hidden layer of GRU; shape=*(n_layers x num_directions, batch_size, hidden_size)*
- `encoder_outputs`: encoder model’s output; shape=*(max_length, batch_size, hidden_size)*

Outputs:

- `output`: softmax normalized tensor giving probabilities of each word being the correct next word in the decoded sequence; shape=*(batch_size, voc.num_words)*
- `hidden`: final hidden state of GRU; shape=*(n_layers x num_directions, batch_size, hidden_size)*

SpeechRecognition

Recognizing speech requires audio input, and SpeechRecognition makes retrieving this input really easy. Instead of having to build scripts for accessing microphones and processing audio files from scratch, SpeechRecognition will have you up and running in just a few minutes.

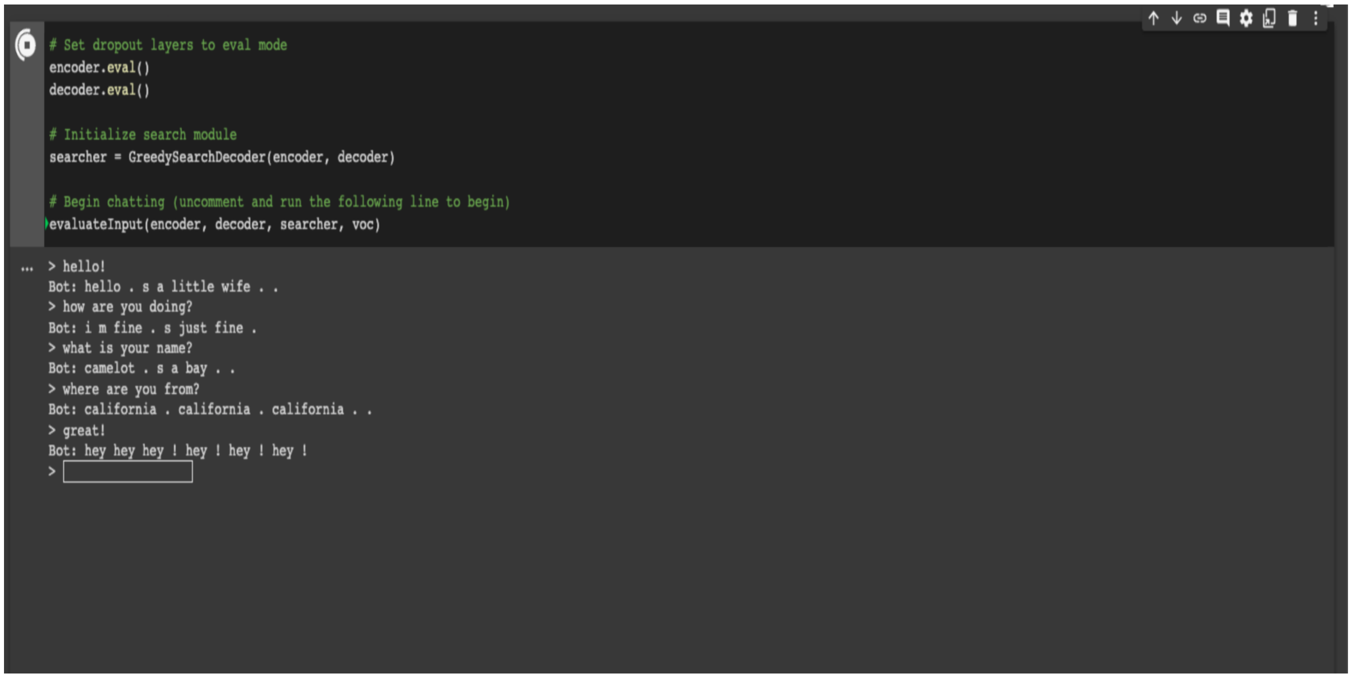
The SpeechRecognition library acts as a wrapper for several popular speech APIs and is thus extremely flexible. One of these—the Google Web Speech API—supports a default API key that is hard-coded into the SpeechRecognition library.

gTTS

gTTS (*Google Text-to-Speech*), a Python library and CLI tool to interface with Google Translate’s text-to-speech API. Writes spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation, or stdout. It features flexible pre-processing and tokenizing.

Implementation

After training model. This is the result



```
# Set dropout layers to eval mode
encoder.eval()
decoder.eval()

# Initialize search module
searcher = GreedySearchDecoder(encoder, decoder)

# Begin chatting (uncomment and run the following line to begin)
evaluateInput(encoder, decoder, searcher, voc)
```

```
> hello!
Bot: hello . s a little wife . .
> how are you doing?
Bot: i m fine . s just fine .
> what is your name?
Bot: camelot . s a bay . .
> where are you from?
Bot: california . california . california . .
> great!
Bot: hey hey hey ! hey ! hey ! hey !
>
```