# **Project Name**

**Keyword Summarizer for Meetings**

# **Introduction**

The end semester project for the Machine Learning course was introduced with the aim of challenging students to create an innovative solution using machine learning models to solve a real-life or existing problem. The project required us to select a problem and come up with a unique solution using machine learning models. Our team selected the task of building a summarizer or keyword classifier to summarize long meetings, as it is a common problem faced by businesses and organizations. The aim of our project was to create an automated tool that could assist individuals in summarizing long meetings and highlighting key information. Through this project, we aimed to demonstrate our understanding of machine learning models and how they can be used to solve practical problems.

# **Problem Definition**

The problem identified for our machine learning project was the tedious and time-consuming task of taking notes and summarizing information during business and finance meetings. In both physical and online meetings, there is usually a designated notes taker and summarizer who is responsible for capturing key information discussed during the meeting. However, this task can be quite challenging and can often divert attention from the important discussions at hand. Additionally, notes takers may miss important points, leading to incomplete or inaccurate summaries. With automation being the need of the time, there is a growing demand for solutions that can help automate the process of notetaking and summarization, as this is required in almost every application. Our project aimed to address this problem by creating an automated tool that uses machine learning models to summarize long meetings and highlight key information discussed during the meeting, freeing up time and attention for participants to focus on the discussion itself.

# **Proposed Solution vs Used Solutions**

**Proposed Solution**

The proposed solution for our machine learning project involved building a keyword classifier that listens to audio from meetings, converts it into text statements, and identifies keywords that might be important in each sentence. Unlike traditional summarization tools that generate blocks of text, our solution focuses only on the important content. The proposed solution included the following key components:

* Keyword classifier that focuses on important content
* DNNs and CNNs for audio to speech conversion
* Hidden Markov Models for reference
* NLP techniques such as Maximum Entropy and Neural Networks for keyword classification
* Continuous real-time training through Reinforcement Learning for improved accuracy.

**Used Solution**

There were multiple attempts were tried to utilize the attention layer concept which allows the model to focus on the more important parts of the input text sequence and then utilizing LTSM model to summarize the text, including encoding and decoding model. The second attempt

involved the use of Random Forest classifier, while using the glove.6B.100d.txt pretrained model to convert text into numeric form, however these attempts were partially unsuccessful, so we went with a more modern approach of utilizing whisper API to convert audio to text and prompt engineering using GPT 3.5 turbo model.

# **Achieved Novelty**

The solution we proposed for our machine learning project achieved several novelties and competitive advantages that set it apart from other systems available in the market.

* Firstly, our solution was unique in its application, as there were very few systems available that were specifically designed for use in business and finance meetings.
* Secondly, our solution was designed to be integrated into any module or application, making it a versatile and valuable tool for a wide range of users.
* In terms of competitive advantage over other systems, our solution was **open source**, which meant that it was available for free to anyone who wanted to use it. This made it accessible to a wider range of users and removed any barriers to entry for small businesses or organizations.
* Also, our solution was completely **free** to use, which made it an attractive option for businesses looking to save costs on notetaking and summarization tools.

By providing a unique and innovative solution that could be integrated into any module or application, and by offering an open-source and free-to-use system, we believe that our solution had a strong competitive advantage over other systems available in the market.

# **Model Description/ Working**

For audio to text conversion, a pre-trained model API was used, namely whisper, which convert the given audio to text which can be directly passed into summarizer model as input.

After considerable research and multiple attempts to create the model for text summarization, some attempts were unsuccessful, but we wish to encompass our effort and knowledge to implement those solution. Here, we will discuss those attempts one by one:

The datasets used for models involved Corpus dataset which included conversations between people as X label and their summaries as Y label. Another dataset we used for model creation attempts was the Reviews model, which included product reviews as the input label and their summaries as output. It goes without saying that initial data was raw, and it involves significant text preprocessing before it can be used for model training.

**Attempt 1**

There were multiple attempts were tried to utilize the attention layer concept which allows the model to focus on the more important parts of the input text sequence and then utilizing LTSM model to summarize the text, including encoding and decoding model to convert text into numerical form for model training and text reconstruction.

The attention layer was sourced from a git link <https://github.com/thushv89/attention_keras/blob/master/src/layers/attention.py>.

The dataset used in this was the Reviews, which was first visualized including the histogram that visualized the word count between summaries and their text.

Then comes the preprocessing part, here, keras preprocessing was utilized, it involved removal of stop words(these are words that do not add anything to the actual meaning of the sentence in context of summarization) using nlpk library, tokenization and padding sequences to homogenize the data.

This includes encoder definition. It takes in a sequence of text (encoder\_inputs), applies an embedding layer to convert the input text into a dense vector representation (enc\_emb), and then applies three LSTM layers (encoder\_lstm1, encoder\_lstm2, and encoder\_lstm3) to encode the input text into a fixed-length vector (encoder\_outputs).

The decoder is defined similarly. It takes in a sequence of text (decoder\_inputs), applies an embedding layer to convert the input text into a dense vector representation (dec\_emb), and then applies an LSTM layer (decoder\_lstm) to decode the encoder output into a sequence of text.

An attention layer (attn\_layer) is then added to the decoder. The attention layer computes the relevance of each encoder output to the current decoder input and produces a weighted sum of the encoder outputs. This helps the model focus on the most important parts of the input when generating the output. The decoder output and the attention output are concatenated (decoder\_concat\_input) and passed through a dense layer (decoder\_dense) to generate the final output.

However, this implementation however, was unsuccessful. And included unfeasible training times, for instance 3.7 hours for a single Epoch. As can be viewed in the figure attached:

|  |
| --- |
|  |

**Attempt 2**

Attempt involved the same early including dataset processing dropping unnecessary columns and its visualization, steps except the preprocessing was done using custom defined function, the code for the function can be seen:

|  |
| --- |
| import nltk  from nltk.corpus import stopwords  from nltk.tokenize import word\_tokenize  from nltk.stem import PorterStemmer  import string  nltk.download('punkt')  nltk.download('stopwords')  def preprocess\_text(text, max\_tokens=None):      # Convert to lowercase      text = text.lower()      # Remove punctuation      text = text.translate(str.maketrans('', '', string.punctuation))      # Tokenize the text      tokens = word\_tokenize(text)      # Remove stop words      stop\_words = set(stopwords.words('english'))      filtered\_tokens = [token for token in tokens if token not in stop\_words]      # Limit number of tokens      if max\_tokens is not None:          filtered\_tokens = filtered\_tokens[:max\_tokens]      # Stemming      porter\_stemmer = PorterStemmer()      stemmed\_tokens = [porter\_stemmer.stem(token) for token in filtered\_tokens]      # Convert back to text      preprocessed\_text = ' '.join(stemmed\_tokens)      # while (len(preprocessed\_text) < 94):      #   preprocessed\_text += "i"      return preprocessed\_text |

The code is self-explanatory using the comments, while the concept of stop words was explained earlier.

This attempt further involved training and testing a model created using Random Forest model. However, the output was not as satisfactory. It be seen here:

|  |
| --- |
|  |

**Attempt 2.5**

It is an extension of the previous model based on partial success of last attempt, we utilized a different dataset for this which involved sightly different preprocess conditions since last data was in the form of .json forms and the new dataset was a csv with different fields and datatypes. However, this attempt resulted in a worse failure than before.

**Final Attempt**

This attempt involved the use of GPT 3.5 prompt engineering for creation of the summarizer model. It was pretty straight forward procedure, where you connect to the GPT API using the api\_key then specify the relevant function gpt\_completion(), this takes prompt instructions and text to summarize and generate an almost perfect summary.

Output is demonstrated as:

|  |
| --- |
| Audio Conversion: |
|  |
| Text Summarization: |
|  |

**Dataset Links**

1. **Corpus:** Attached in the zip file.
2. **Reviews dataset:** https://drive.google.com/file/d/1PxNS-tk\_\_sGojoOomtGpVWa2TREg72\_o/view?usp=sharing
3. **glove.6B.100d.txt:** <https://drive.google.com/file/d/1z7lTTRAMpChrQbsguIREJ8Wlz1D82VLO/view?usp=sharing>

# **Problems Faced**

Following are some major problems that were faced by us:

1. For usable summarizer models, it was a prerequisite to use datasets that were adequately large, sadly we did have enough processing power to feasibly train models on such large datasets, which led to less than satisfactory models.
2. For the creation of summarizer model, advanced concepts not yet covered such as NLP processing, preprocessing techniques, LSTM, encoder-decoder model, and attention layer was used which required further research and time.
3. There were time constraints for the project also caused increased pressure to create a usable model which satisfies the functional requirements proposed.

# **Future Work**

For future work, we considered the following:

1. The use of APIs always involves some module dependencies that need to be resolved for model execution. We wish to make the model auto-resolve such dependencies.
2. The creation of GUI to add independence to the model as a commercialized application free for use.
3. To introduce scalability of the model to be used commercially and as a integratable model in large scale applications.

# **Conclusion**

We were successful in creating a program that takes audio input and produces an concise extractive summary, mentioning all the required keywords. Our wish to produce a model from scratch was unfulfilled, however, in the process of doing so we able to do considerable research. Therefore, in academic respect it was a good learning experience. Moreover, in the final attempt the process of API introduction allowed us to take a look at the modern technologies that be utilized to achieve variety of tasks efficiently.

# **References**

1. <https://intellipaat.com/blog/what-is-lstm/?US#:~:text=LSTM%20Explained,-Now%2C%20let's%20understand&text=First%2C%20you%20must%20be%20wondering,especially%20in%20sequence%20prediction%20problems>.
2. <https://vitalflux.com/encoder-decoder-architecture-neural-network/#:~:text=Encoder%20%E2%80%93%20decoder%20architecture%20is%20a,sequence%2Dto%2Dsequence%20modeling>.
3. <https://www.nltk.org/>
4. <https://huggingface.co/tasks/summarization>
5. <https://metatext.io/datasets-list/summarization-task>
6. <https://www.ibm.com/topics/recurrent-neural-networks#:~:text=A%20recurrent%20neural%20network%20(RNN,data%20or%20time%20series%20data>.
7. <https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/>
8. <https://openai.com/blog/introducing-chatgpt-and-whisper-apis>