**CSE1901 - Technical Answers to Real World Problems (TARP)**

**Project Report**

**Lecture Summarizer - Using Automatic Speech**

**Recognition Systems for transcription and Summary generation models for written summary of recorded lectures**

*By*

Mrigank Shukla 19BCE1200 Vikash Sunil 19BCE1376

Jeevesh Nandan 19BCE1387 Nidhish N. Suryawanshi 19BCE1711

B. Tech Computer Science and Engineering

*Submitted to*

**Dr. N. MAHESWARI**

**School of Computer Science and Engineering**



*April 2022*

**DECLARATION**

We hereby declare that the report titled “**Lecture Summarizer - Using Automatic Speech Recognition Systems for transcription and Summary generation models for written summary of recorded lectures”** submitted by us to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. N. Maheswari**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

**Mrigank Shukla 19BCE1200**

**Vikash Sunil 19BCE1376**

**Jeevesh Nandan 19BCE1387**

**Nidhish N. Suryawanshi 19BCE1711**

**CERTIFICATE**

Certified that this project report entitled “**Lecture Summarizer - Using Automatic Speech Recognition Systems for transcription and Summary generation models for written summary of recorded lectures”** is a bonafide work of **Mrigank Shukla (19BCE1200), Vikash Sunil (19BCE1376), Jeevesh Nandan (19BCE1387) and Nidhish N. Suryawanshi (19BCE1711)**and they carried out the Project work under my supervision and guidance for CSE1901 - Technical Answers to Real World Problems (TARP).

**Dr. N. Maheswari**

SCOPE, VIT Chennai

**ACKNOWLEDGEMENT**

We acknowledge the hard work of all the researchers who put so much efforts in developing such robust deep learning algorithms. We also have deep appreciation for the open source community, for their tireless efforts in maintaining repositories that help us cash constrained students. We also would like to thank Dr. N. Maheswari ma’am for her constant guidance and support. The important inputs and her insights kept our team on track. We are very thankful for her loving nature and friendly behavior that helped us discuss our problems freely with her, which made this project possible.

**Mrigank Shukla 19BCE1200**

**Vikash Sunil 19BCE1376**

**Jeevesh Nandan 19BCE1387**

**Nidhish N. Suryawanshi 19BCE1711**

**ABSTRACT**

The fast growth of the internet coverage combined with cheaper access to smartphones has lead to the development of massive open online courses. Many universities have also shifted to the use of online platforms for delivery of recorded and live lectures to the student. This paper aims to employ the cutting edge technology in the field of Machine Learning to assist in the transcription of these lectures, to generate a textual content, then use this content to generate a summary of the given lecture. This will help in generating a short summary of the relevant content discussed in the lecture. These summaries can assist students to select and narrow down their search for relevant topics, and universities can use these summaries for effective knowledge management systems, effective retrieval of the relevant lectures.

**CONTENTS**

Declaration ii

Certificate iii

Acknowledgement iv

Abstract v

1 Introduction 1

2 Literature Survey 2

3 Requirements Specification 9

3.1 Hardware Requirements 9

3.2 Software Requirements 9

4 System Design 10

5 Implementation of System 12

6 Results & Discussion 20

7 Conclusion and Future Work 22

8 References 22

Appendix

**1. Introduction**

In recent years, the use of internet streaming services for educational purposes has skyrocketed. In every discipline, leading educational institutions are in a perpetual rush to set up and use excellent digital lecture access platforms. Major companies such as Coursera, Udacity, and MIT's EDX platform have built a vast archive of online lectures that have been filmed and transcribed by human lecturers, making them freely available to students from all over the world. Other systems, such as Unacademy, Vedantoo, and others, rely on live lectures that are tailored to each class or batch. Colleges that have to focus on the change from offline lecture classes to online lectures using streaming and conferencing systems like Microsoft Teams, Google Meet, and Zoom are also included in these categories. Some of these tools provide the ability to transcribe lectures, but they are paid and the amount of functionality is determined by the subscription plans. Due to network issues, many students are unable to attend these lectures and must listen to the entire presentation in order to comprehend the material. Some prominent colleges, such as VIT and IITs, are focused on creating a catalogue of lectures for students to access; these lectures must include a transcript for greater comprehension and wider reach.These institutions cannot and should not rely solely on sites like YouTube, as evidenced in the case of (Russia vs Ukraine Crisis). They require a solution for automatic lecture transcriptions, as well as the tagging of lecture material for improved query retrieval systems. Our initiative aspires to take this procedure to the next level. The proposed system makes use of open source offline speech recognition methods, which will aid in the transcribing of lectures. Some libraries also support the usage of GPUs, which can result in significant time savings when it comes to transcribing. The transcripts created can be used to create closed captions for the videos.These textual transcripts can also be utilised to obtain a summary of the lecture's contents. The key phrases from the lecture may then be utilised to label the video with the appropriate themes. These tags can subsequently be used to categorise the lectures as part of a more efficient query retrieval procedure. For an in-house solution for the institution, the entire procedure may be automated. The institution is able to train machine learning models on the voices of its teachers, thus increasing the process and reliability of ASR.

**2 Literature Survey**

Speech is the most common form of the communication between human. Almost all the lectures given all around the world renowned universities depend on the effectiveness of the lecturer in delivering the lecture. We love the lectures or faculties who are able to communicate even the most difficult concepts in the simplest way which is easily digestible by students. Hence it is quite necessary to use the effectiveness of speech in the utilization of man-machine interaction. Most of the research work done in this field is being focused on two main fields that are Automatic Speech Recognition (ASR) which refers to the conversion of speech to its textual format [1]. The second type of the system that were being developed are the Text to Speech system that are used to convert the given text to its equivalent speech format, with correct pronunciation of each term with respect to its usage in the correct part of speech structure. We will be focusing on only the ASR part as this will be used to transcribe the speech in the lecture that are the input to the system. Though it has been asked to focus only on the recent most research papers it will be really helpful if I try to understand the development in this field in the linear manner without over explaining the earlier developments. The development of the most initial system began in the year 1952 where the system could understand the spoken digits when separated with sufficient pause [2]. There were further research done on this field but it was mostly constrained to small constrained speech samples with minimal vocabulary. Until the advent of machine learning and particular growth in the field of deep learning. Applications like Siri, Amazon Alexa and Google Assistant are very popular and useful in this field but this means that the data is uploaded in the severs based on U.S with no control to the user. Hence the need of open source development in this field is quite paramount and as a student of VIT, we are also thinking of a resource that can be trained and developed within our university with resources gathered and maintained by our institute.

The ASR application have some conman structure that is almost the same for all the implementations, a front-end to the system is an acoustic model that is used to extract useful features from the speech signal [3]. Using these features a feature vector is generated that has many features Cepstral Analysis, Kernal Based Feature Extraction, Dynamic Feature Extraction, Mel-Frequency Scale Analysis and many more. Then using these features and various approaches to machine learning algorithms, we develop an acoustic model that has the task of matching the phonetic units to the relevant words or alphabets then the generated words will be passed through a language model to get the grammatically sound words. Generally Deep Speech [4] based on the deep neural networks is one of the first and best known ASR system. It is also the fundamental basis for the Deep Speech implementation of Mozilla. The researchers from Baidu were able to further enhance their model to achieve a comparable performance with respect to human baseline on the LibriSpeech dataset [5]. The deep speech was made open source with contributions from all over the world. The model used Recurrent Neural Networks which are often used for Deep Neural network based ASR systems.

Facebook’s (now Meta) developed a model called Wav2Letter that utilized Convolutional Neural Networks (CNN) [6]. The models was able to get good results on the LibriSpeech Dataset with improvement done in the next model Wav2Letter++ as well [7] reaching a Word Error Rate (WER) of 5% in LibriSpeech as well as being faster. The team is also working on transformer based model that were able to outperform RNN based model or LSTM models reaching a WER of 2.26% [8]. Kladi speech recognition toolkit [9] adopted DNN’s as a part of their toolkit. They also created an interface for the toolkit to work with PyTorch [10]. The combined approach performed well giving an average WER of 6.2% on LibriSpeech dataset using light gated recurrent units (Li-GRU) which are less complex version of LSTMs.

The approach incorporated by the research paper I am referencing as one of our main papers for this project [12] combines the approach to speech detection and speech transcription. The researchers have utilized the idea of Voice Activity Detection for getting the parts of the audio which have speech in it, as the task of transcribing speech is a computationally strenuous task. It was very necessary for them to take the part of the audio files which had speech in it. They used an effective Speech Segmenter [13] that utilizes Convolution Neural network to identify the parts of the speech which had human speech in it. The used model is able to effectively segement the audio provided for the parts that contain the speech parts. The parts of the audio where speech is present is stored as a CSV file that contains information of the audio as a mathematical representation. This done to the full audio files. The generated CSV files are then passed to the Mozilla DeepSpeech engine for the process of ASR. The engine uses the language models that were trained on the massive open dataset of voices. The engine then transcribed the audio segments. Generated transcripts from each part is then combined to create a full transcription of the whole audio files. They also used ffmpeg to extract only the audio from the input video files. The audio files were also converted to WAV format using ffmpeg. Researchers were inclined to using offline open source implementation of ASR only as there is no risk of sending sensitive information especially in case of forensic application to commercial cloud server.

The bulk of modern ASR systems are focused on reducing Word Error Rate (WER), with little attempts to discover struc- tural information in spoken texts. The output of a traditional speech recognition system is usually single-case words with no punctuation marks. Such content is difficult to read and understand because of the missing information. Furthermore, missing data, notably punctuation, sentence boundaries, and capitalization, is crucial for parsing, information extraction, conversation act modelling, Named Entity Recognition (NER), summarization, and translation, among other types of auto- matic downstream processing. There are numerous punctua- tion marks to consider for spoken texts: comma, period, excla- mation mark, question mark, colon, semicolon, and quotation marks. The bulk of these markers, on the other hand, are infrequent and difficult to implant or analyse. For example, quotation marks and semicolons are commonly used in a hap- hazard and inconsistent manner. As a result, most studies con- centrate on the full stop and comma, which have higher corpus frequencies.In a few more restricted studies, the question mark is also taken into account. The problem of recovering punctuation is inextricably tied to the problem of recognising sentence boundaries, especially when anticipating punctuation such as full stops, question marks, and exclamation marks, which are all associated with sentence boundaries. These tasks serve as a basis for more complex Natural Language Processing (NLP) tasks, and their impact on future tasks has been investigated in a variety of studies on speech processing. [14]. The researchers [15] incorporated three models for punctuation prediction: a DNN model, a bidirectional RNN with an attention mechanism, and a bidirectional LSTM with a CRF layer. On the reference data, their experimental findings on the English IWSLT 2011 dataset received a 64.2 percent F-score, exceeding prior best-practice results. According to the authors, guessing a comma in English is more difficult than guessing a period or a question mark. Kim [16] offers a layer-wise multi-head attention architecture for deep recurrent neural networks. Their experimental outcomes have an overall F-score of 68.9% on the IWSLT 2011 dataset. Yi and Tao [17] described a process where punctuation marks are predicted via the self-attention technique. Word and speech embedding characteristics from the pre-trained Word2Vec and Speech2Vec are used to train the self-attention based model. The model can learn lexical and acoustic properties from any type of text data that does not have equivalent audio or speech data. The suggested strategy is successful, as evidenced by the experimental results on the English IWSLT2011 datasets. The self-attention based model, which was trained utilising word and voice embedding characteristics, beats the prior single model. It also outperforms the previous best ensemble model in terms of performance. Further research on this model was done by the same researcher [18] in which adversarial transfer learning was supported in this study as a strategy to improve punctuation prediction ability. To transfer bidirectional repre- sentations to punctuation prediction algorithms, a pre-trained BERT model was employed. Furthermore, for the punctuation prediction job with an auxiliary POS tagging work, adversarial multi-task learning was employed to obtain task invariant knowledge. In the experiments, IWSLT2011 datasets were used. Models with parameters transferred from a pre-trained BERT model beat models with random initialization by a significant margin, according to the findings. In addition, task invariant information boosted the accuracy of punctuation predicting models, according to the findings. Their most advanced model outperforms previous best-in-class models. The researchers use the deep bidirectional transformer (Bert) model to cope with the long-distance reliance. Because the PR at a given position will look up the full sentence, Bert’s self-attention mechanism enables it to learn and apply long- distance syntactic and semantic links. The tests demonstrated that this new method accurately anticipates question marks. The most difficult aspect of identifying interrogatives is the syntax (English) and semantic (Chinese) long-distance present in interrogative sentences. These major improvements are largely associated with greater recall, demonstrating that Bert can discover long-distance patterns that are not detectable by standard approaches and use these patterns to identify interrogatives. The researchers [14] presented a low-latency, RNN-based punctuation model in this method, with the pri- mary goal of improving closed captions. They used objective metrics of information retrieval to test the model on Hungarian and English datasets, and performance was compared to a MaxEnt sequence tagger baseline. They next expanded their investigation to Hungarian, employing subjective criteria to analyse punctuation. In comparison to some of the more complicated models, they chose a lightweight and fast RNN model that maintained performance. The proposed technique also aimed towards real-time, low-latency operation. They utilised this method to model sequences in strongly agglu- tinating Hungarian, which has a lot less limited word order than English because grammatical functions are less dependent on word order than on suffixes, making sequence modelling more challenging owing to the larger variation in the data. They also put this strategy to the test by conducting subjective testing to see if users gain from automated punctuation, which they found to be true for the primary audience of Hungarian speakers.

Dealing with large volumes of textual data necessitates the implementation of effective solutions. This is a problem that can be solved by using automatic text summarization systems.There is a substantial growth of textual material in digital form in today’s technology period, and it is constantly growing. Automatic summarising methods make it easier to deal with large amounts of text data in a timely manner. These methods aim to provide summaries that are thorough, succinct, fluent, and capable of retaining all of the important information in a topic. Search engine snippets created as a result of querying a document, and even some online news sites which produce condensed news in the format of headlines to aid surfing, are examples of text summarising applications [20]. As a result, working on the architecture of current au- tomatic summarization systems and innovating them to make them capable of fulfilling the demands of continually rising data, based on user preferences, becomes extremely important. Automatic summarization systems may be characterised in one of two ways: extractive or abstractive. The key portions of the content are extracted based on specific score parameters and then combined to form the summary when modelled using extractive techniques [21]. Since this text is paraphrased to provide a summary containing terms that differ from the original text, abstractive approaches are more difficult to work with. All summary approaches and models, whether extractive or abstractive, have the very same goal: to produce fluent, non- redundant, and coherent summaries. Both techniques may be used to create summaries from a single source document or a collection of source documents. In the subject of abstractive text summarization, some new models have recently been developed. It begins by discussing Facebook (Now Meta ) researchers’ [22]work, which proposes a model for phrase summarization. A neural network with attention is used in the model. In addition to the model, the researchers applied a generation method to aid the system in producing accurate summaries. They then upgraded their model using the con- ditional RNN framework in another paper [23]. They used a convolutional attention-based encoder for conditioning to assist the decoder focus on relevant input words at each phase of summary synthesis. The new approach outperformed prior work using the Gigaword corpus and fared exceptionally well on DUC 2004. Scaling the abstractive summarising system to create paragraph-level summaries, accurate alignment, and consistency in output creation are among the problems which they were facing earlier.

The researchers [23] used a microblogging service to con- struct a dataset called LCSTS (Large-scale Chinese Short Literature Summarization) for another work on abstractive summarization in Chinese language text. This collection is made up of a big corpus of Chinese short sentences with outlines. This corpus was then used to introduce an RNN- based summary generating model. The usage of hierarchical RNNs for summary creation and study into unusual word difficulties is one of the researchers’ proposals for future work. PEGASUS (Pre-training with Extracted Gap-sentences Abstractive Summarizing Sequence to Sequence Models)a new abstractive summarization technique introduced by Google researchers [24], outperformed earlier models with even superior results. The encoder-decoder design is based on a typical transformer-based encoder-decoder architecture . For numerous abstractive summarization tasks, the authors initially established a pre-training aim. This self-supervised aim was dubbed GSG (Gap Sentences Generation). Their approach was characterised as fundamental sentence selection. C4 (Colossal and Cleaned version of Common Crawl) and HugeNews datasets were utilised for pre-training. They utilised XSum, Gigaword, arXiv, CNN/DailyMail , PubMed, Newsroom , Bigpatent, WikiHow, Reddit TIFU, MultiNews, AESLC, and BillSum for downstream summarising. For automatically gen- erated meeting transcripts, Microsoft’s speech and dialogue research department [25] developed an abstractive text sum- marising system. Meeting summarization, they claim, differs from document summary. Because a meeting involves a large number of people, the form of meeting transcripts is ex- tremely varied due to differences in semantic styles, opinions, and participation roles. The suggested meeting summaris- ing architecture, dubbed Hierarchical Meeting Summarization Network, is built on deep learning methodologies (HMNet). To construct abstractive summaries of meeting transcripts, the HMNet model employs the encoder-decoder transformer architecture . The model was tested using three variants of the ROUGE [26] metric (Rouge-1, Rouge-2, and Rouge-SU4) on the AMI and ICSI meeting corpus. Because manual text summarization of the vast amount of textual content on the Internet or in other archives is impossible, ATS systems have emerged as the primary solution to address this urgent and pressing problem. Although there are several artificial text summarizers in the literature, their results are still far from those of human text summary. The relevance of a text changes with its kind, application domain, and user choice, making it challenging for the computer to grasp and determine the ”most significant” elements in the text. Hence as show above since the 1950s, there have been ongoing research attempts to overcome these challenges and find new solutions. The scientific community has primarily focused on the extractive text summarization approach over time, and has implemented the summarization methods of this approach for a variety of applications, including customer reviews, media publications, blog posts, email communications, scientific articles, legal documents, and biomedical documents. In practise, extractive ATS systems generate summaries that are vastly different from those produced by humans [27]. Abstractive and hybrid text summarization algorithms have been tested in the past. We are also trying to scratch the surface of this modern rapidly developing fields in the hope to get some good results, which can be useful. Researchers hope to automate the creation of summaries that are similar to those produced by humans.

**3** **Requirements Specification**

3.1 **Hardware Requirements**

The hardware requirements of the model are a bit intensive. The designed application needs a minimum Intel i5 8th Generation CPU or any other CPU of similar performance the model can perform better if you have a GPU of minimum capacity of 6GB.

3.2 **Software Requirements**

The software requirements of the model are represented in the code. For the first run of the program please be connected to a good internet connection preferably of minimum speed of 20 MbPs not some hostel WiFi. The code has been designed such that it will automatically store the given model in your computer’s memory. After the initial run, internet connectivity is not required at all.

**4 System Design**

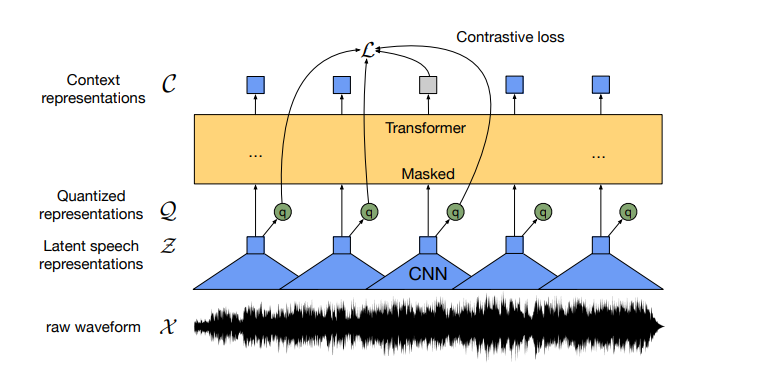


Fig.1 Wav2Vec2.0 Deep Learning Model

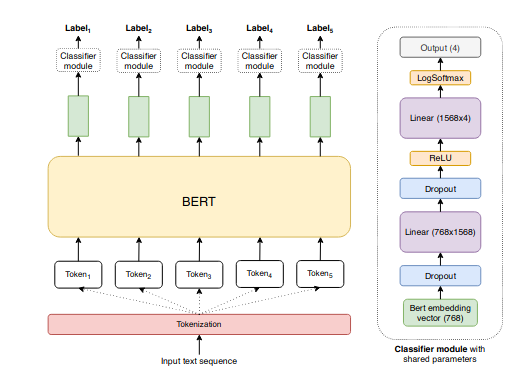
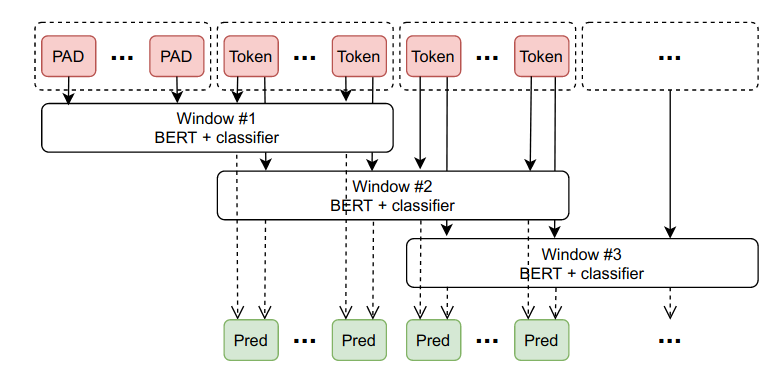


Fig.2 Complete Model for Punctuation Restoration. [11]

Fig.3 BERT based punctuation model. [11]

**5 Implementation of System and Results Discussion**

**Automatic Speech Recognition**

The most common way for humans to interact with each other is through speech. Almost all of the lectures given at world-class universities are dependent on the lecturer’s ability to deliver the course. We respect lecturers or professors who can explain even the most complicated subjects in a way that students can understand. As a result, employing the efficacy of speech in the use of man-machine interaction is crucial. The majority of research in this area is focused on two main areas: Automatic Voice Recognition (ASR), which refers to the conversion of speech to its written format [1], and Natural Language Processing (NLP).The Text to Voice system, which translates a given text into its corresponding speech format, with exact pronunciation of each word in connection to its employment in the right region of the speech structure, is the second type of system under development. We’ll focus on the ASR component since it will be used to transcribe the lecture’s speech, which will be the system’s input. Large amounts of labeled training data improve neural networks. Labeled data, on the other hand, is far more difficult to come by in many situations than unlabeled data: To achieve acceptable performance, existing speech recognition algorithms require thousands of hours of transcribed speech, which is not accessible for the great majority [2] of the languages spoken globally. Hence we have decided to go for a Neural Network model that is based on self-supervised learning instead of purely supervised learning this makes the same model easily reproducible in languages such as Hindi and Tamil or any other Indian Origin Language. This approach is similar to what we see in kids there is no labeled data provided to kids for learning purposes, they tend to learn based on associating each sound to different meaning. They can then reproduce the sound even though they cannot write it in linguistic form.

The model we are using is Wav2Vec2 as mentioned in the previous project report. It is useful for our project as it is based on a different type of learning which only requires a minimal amount of annoted data but large amount of unlabeled data called self-supervised learning. It is a machine learning approach that allows you to learn broad representations of data from unlabeled data samples and then fine-tune the model with annotated data. This has shown to be quite successful in natural language processing and is currently a hot topic in computer vision research. A technique for self-supervised learning of representations from raw audio data has been developed by Facebook researchers. Their method, which is comparable to masked language modelling, uses a multi-layer convolutional neural network to encode spoken sounds and then masks spans of the latent speech representations. The latent representations are supplied into a Transformer network, which creates contextualised representations, and the model is trained using a contrastive task that distinguishes the actual latent from the distractor.

For the purpose of our project we have used pre-trained models as there is no way we can compete with already trained models by tech giants such as Facebook with the given resources. So for the purpose of our project we have used tested two models the first model is the Facebook Wav2Vec2 960 hours of voice trained data another model is an effort by a startup for Indian Dialect English called Vakyansh trained on 700 hours of voice data.

For the purpose of our project we have used pre-trained models as there is no way we can compete with already trained models by tech giants such as Facebook with the given resources. So for the purpose of our project we have used tested two models the first model is the Facebook Wav2Vec2 960 hours of voice trained data another model is an effort by a startup for Indian Dialect English called Vakyansh trained on 700 hours of voice data. They use masked language modelling in BERT to pre- train the model by masking a particular fraction of time steps in the latent feature encoder space. The goal of the training is to find the best quantized latent audio representation for each masked time step in a set of distractors, then fine-tune the final model using the labelled data. They substitute a trained feature vector that is shared across all masked time steps with a subset of the feature encoder outputs, or time steps, before feeding them to the context network; quantization module inputs are not masked. To mask the latent speech representations output by the encoder [3], we randomly sample without replacement a certain proportion p of all time steps to be starting indices for latent features. Specifically, we randomly sample without replacement p time steps for latent features, and set the starting index of these time steps as the time indices of the corresponding latent features, i.e., we set the starting index of a time step as the index of the latent feature it belongs to. The result of this training and building of models is that given the speech dataset it creates and understand hidden patterns in speech. Then this model can be trained with a labeled dataset which is very small. This results in the model understanding the phoneme level structure of the language. On top of that we are using Language Models these models help in giving a correct representation to the words given by the Wav2Vec2.0 model so that the most suited word is selected out of the given options.

The time taken for transcription is a little large, hence we have curated a specific set of videos from NPTEL videos as they closely represent the video lecture of any online lecture platforms. The results of ASR on these videos are explained in TABLE 1 and TABLE 2. Now the problem with the ASR transcripts is the presence of many spelling errors the best way will be to give the use the choice to remove such errors by manually going through the transcript like in Word or any other document editing format. But the problem with such approach is that it will take a huge amount of time to go through length transcripts in an effort to find spelling errors. Hence we have planned to automate this task of spelling correction. We are using a new algorithm developed by Wolf Garbe [4], the basic idea of the algorithm is to find the minimum edit distance of the closest word to the misspelled word by performing deletion alone. This speeds up the process instead of the regular edit distance which includes deletion, insertion, replacement. The algorithm takes a dictionary that is the Googles Indexed term dictionary then for each word calculates all possible words with the specified edit distance by deletion of the letters. Then the word which matches with the misspelled word,is selected the correct word associated with edit distance word is then selected. In this manned we select the best word for each of the misspelled word, then correct them automatically. The results on deeper analysis show that these ASR tran- scriptions work very well, when the lectures are in a free flow non-technical format. Whereas the model tends to perform worse on the videos where the content of the video is super technical involving very domain-specific terms or the lecturer explaining various mathematical or graphical concepts. Also the second model on Indian English though taking almost half the time than Facebook’s model, the WER for it is large in many instances as compared to Facebook’s. Hence we finall decided to use the Facebook’s 960H model trading accuracy for time.

**Punctuation Of Transcription**

The bulk of modern ASR systems are focused on reducing Word Error Rate (WER), with little attempts to discover structural information in spoken texts. The output of a traditional speech recognition system is usually single- case words with no punctuation marks. Such content is difficult to read and understand because of the missing information. Furthermore, missing data, notably punctuation, sentence boundaries, and capitalization, is crucial for parsing, information extraction, conversation act modelling, Named Entity Recognition (NER), summarization, and translation, among other types of automatic downstream processing. There are numerous punctuation marks to consider for spoken texts: comma, period, exclamation mark, question mark, colon, semicolon, and quotation marks. The bulk of these markers, on the other hand, are infrequent and difficult to implant or analyse. For example, quotation marks and semicolons are commonly used in a haphazard and inconsistent manner. As a result, most studies concentrate on the full stop and comma, which have higher corpus frequencies.In a few more restricted studies, the question mark is also taken into account. The problem of recovering punctuation is inextricably tied to the problem of recognizing sentence boundaries, especially when anticipating punctuation such as full stops, question marks, and exclamation marks, which are all associated with sentence boundaries. These tasks serve as a basis for more complex Natural Language Processing (NLP) tasks, and their impact on future tasks has been investigated in a variety of studies on speech processing. [5]. The researchers [6] incorporated three models for punctuation prediction: a DNN model, a bidirectional RNN with an attention mechanism, and a bidirectional LSTM with a CRF layer. On the reference data, their experimental findings on the English IWSLT 2011 dataset received a 64.2 percent F-score, exceeding prior best-practice results. According to the authors, guessing a comma in English is more difficult than guessing a period or a question mark. Kim [7] offers a layer-wise multi-head attention architecture for deep recurrent neural networks. Their experimental outcomes have an overall F-score of 68.9% on the IWSLT 2011 dataset. Yi and Tao [8] described a process where punctuation marks are predicted via the self-attention technique. Word and speech embedding characteristics from the pre-trained Word2Vec and Speech2Vec are used to train the self-attention based model. The model can learn lexical and acoustic properties from any type of text data that does not have equivalent audio or speech data. The suggested strategy is successful, as evidenced by the experimental results on the English IWSLT2011 datasets. The self-attention based model, which was trained utilising word and voice embedding characteristics, beats the prior single model. It also outperforms the previous best ensemble model in terms of performance. Further research on this model was done by the same researcher [9] in which adversarial transfer learning was supported in this study as a strategy to improve punctuation prediction ability.

We finally settled for a BERT based uncased pre-trained model for the task of punctuation restoration. The idea is similar to that of sequence labeling problems. The target classes in punctuation restoration are NONE, FULL STOP, COMMA, and QUESTION. Most of the researchers seldom use other punctuation as there frequency of occurrence is quite low, and training model to identify them negatively impacts the accuracy of more semantically relevant punctuation. The model was trained using a dataset which is relevant to our scope of study, that is TED Talks dataset IWSLT [10]. The dataset is first converted to lowercase and then it is transformed to the structure shown in TABLE III.The model is based on BERT. The model applies a tokenization which it has learned after being trained over thousands of documents. The model then formulates a continuous representative of all of the tokens. The model then slides over the input data of tokenized representation of the words, giving multiple predictions for each token and then aggregates the probabilities of each of the output class and then predicts the class with the highest prediction.

**Text Summarization**

Dealing with large volumes of textual data necessitates the implementation of effective solutions. This is a problem that can be solved by using automatic text summarization systems.There is a substantial growth of textual material in digital form in today's technology period, and it is constantly growing. Automatic summarizing methods make it easier to deal with large amounts of text data in a timely manner. These methods aim to provide summaries that are thorough, succinct, fluent, and capable of retaining all of the important information in a topic. Search engine snippets created as a result of querying a document, and even some online news sites which produce condensed news in the format of headlines to aid surfing, are examples of text summarizing applications [13]. As a result, working on the architecture of current automatic summarization systems and innovating them to make them capable of fulfilling the demands of continually rising data, based on user preferences, becomes extremely important. Automatic summarization systems may be characterized in one of two ways: extractive or abstractive. The key portions of the content are extracted based on specific score parameters and then combined to form the summary when modelled using extractive techniques [12]. We have used extractive text summarization and then applied abstractive text summarization over it.

In this text summarization what we do it that we find most relevant sentences from the text corpus and use those sentences for summary of document. The process by which we do this is by finding most relevant words among text corpus and then get score of all those sentences which the greatest number of relevant words among sentences of document and use those sentences to build summary of document.

*TF-IDF:* It stands for Term Frequency – Inverse Document frequency. This generally used in web mining or web related work to compute score for each word over various document or web pages. This method is widely used in text mining and information retrieval. We have used this method on our single text document which get divided into number of sentences and treated as different document of text corpus. After that we calculate term frequency of each word after some cleaning means removal of stop words and replacing all other character, then we calculated IDF for list of all words over different sentences.

*Cosine similarity*

Cosine similarity measures the similarity between two vectors of an inner product space these vectors in case of Natural Language processing tends to be count vectorization of documents. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction this generally means that if those count vector of document or document are similar if cosine similarity value is 1 then it means that both documents are the same. We have used this similarity method to check similarity among various sentences of our text corpus. This is also one of the widely used method and simple of checking similarity among documents or in our case sentences.\\

*Page Rank Algorithm*

The PageRank algorithm give us a probability distribution used to represent the likelihood that a person randomly selecting different links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called “iterations”, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

We have used this algorithm to get weight of sentences in text corpus with help cosine similarity matrix. It is based on the concept that words which occur more frequently are significant. Hence, the sentences containing highly frequent words are important. Based on this, the algorithm assigns scores to each sentence in the text. To check what kind of text summary we getting for document we have used rouge score method in this we get to find what amount of our extractive text summary near to the reference summary. Rouge method returns three metrics for summary ‘rouge-1’, ‘rouge-2’ and ‘rouge-l’ which means unigram, bigram and longest common sequences of summary to get ‘f1 score’, ‘recall score’ and ‘precision score’ for all three metrics. In our case, value of f1, recall and precision score of summary for different rouge metric is around same value between 0.4 - 0.7 this show that even though summary will not be absolutely perfect but whenever reference summary will have words from document chances are high for getting same kind of result.

**6. Results Tables**

12§latex§\begin{table*}[h]
	\begin{center}
		\renewcommand{\arraystretch}{2}
		\caption{Results of Automatic Speech Recognition for Curated Videos Using Facebook 960 Hrs \\}
		\begin{tabular}{ | c | c | c | c | c |  }
			\hline
			\textbf{ Video Type } & \textbf{Length} & \textbf{WER} & \textbf{Time Taken} \\
			\hline
			NPTEL English Literature & 55 Mins & 20.45 & 15 Mins \\
			\hline
			NPTEL English Literature & 50 Mins & 24.15 & 14 Mins \\
			\hline
			NPTEL Operating System & 55 Mins & 47.10  & 15 Mins  \\
			\hline
			NPTEL Language Processing & 58 Mins & 42.65 & 15 Mins \\ 
			\hline
			NPTEL Aritificial Intelligence & 52 Mins & 44.65 & 15 Mins  \\
			\hline
			NPTEL Industrial Engineering & 58 Mins & 37.65 & 15 Mins \\
			\hline
		\end{tabular}
	\end{center}
\end{table*}

§png§600§FALSE§

![12§inline§\begin{table*}[h]
	\begin{center}
		\renewcommand{\arraystretch}{2}
		\caption{Results of Automatic Speech Recognition for Curated Videos Using Facebook 960 Hrs \\}
		\begin{tabular}{ | c | c | c | c | c |  }
			\hline
			\textbf{ Video Type } & \textbf{Length} & \textbf{WER} & \textbf{Time Taken} \\
			\hline
			NPTEL English Literature & 55 Mins & 20.45 & 15 Mins \\
			\hline
			NPTEL English Literature & 50 Mins & 24.15 & 14 Mins \\
			\hline
			NPTEL Operating System & 55 Mins & 47.10  & 15 Mins  \\
			\hline
			NPTEL Language Processing & 58 Mins & 42.65 & 15 Mins \\ 
			\hline
			NPTEL Aritificial Intelligence & 52 Mins & 44.65 & 15 Mins  \\
			\hline
			NPTEL Industrial Engineering & 58 Mins & 37.65 & 15 Mins \\
			\hline
		\end{tabular}
	\end{center}
\end{table*}

\begin{table*}[h]
		\begin{center}
			\renewcommand{\arraystretch}{2}
			\caption{Results of Automatic Speech Recognition for Curated Videos Using Vakyansh Indian English 700 Hrs \\}
			\begin{tabular}{ | c | c | c | c | c |  }
				\hline
				\textbf{ Video Type } & \textbf{Length} & \textbf{WER} & \textbf{Time Taken} \\
				\hline
				NPTEL English Literature & 55 Mins & 30.45 & 6 Mins \\
				\hline
				NPTEL English Literature & 50 Mins & 34.15 & 6 Mins \\
				\hline
				NPTEL Operating System & 55 Mins & 44.10  & 6 Mins  \\
				\hline
				NPTEL Language Processing & 58 Mins & 43.65 & 6 Mins \\ 
				\hline
				NPTEL Aritificial Intelligence & 52 Mins & 49.65 & 6 Mins  \\
				\hline
				NPTEL Industrial Engineering & 58 Mins & 41.65 & 6 Mins \\
				\hline
			\end{tabular}
		\end{center}
	\end{table*}§png§600§FALSE§](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABAQMAAAAl21bKAAAAA1BMVEX///+nxBvIAAAACXBIWXMAAA7EAAAOxAGVKw4bAAAACklEQVQIHWNgAAAAAgABz8g15QAAAABJRU5ErkJggg==)12§latex§\begin{table*}[h]
	\begin{center}
		\renewcommand{\arraystretch}{2}
		\caption{Results of Automatic Speech Recognition for Curated Videos Using Facebook 960 Hrs \\}
		\begin{tabular}{ | c | c | c | c | c |  }
			\hline
			\textbf{ Video Type } & \textbf{Length} & \textbf{WER} & \textbf{Time Taken} \\
			\hline
			NPTEL English Literature & 55 Mins & 20.45 & 15 Mins \\
			\hline
			NPTEL English Literature & 50 Mins & 24.15 & 14 Mins \\
			\hline
			NPTEL Operating System & 55 Mins & 47.10  & 15 Mins  \\
			\hline
			NPTEL Language Processing & 58 Mins & 42.65 & 15 Mins \\ 
			\hline
			NPTEL Aritificial Intelligence & 52 Mins & 44.65 & 15 Mins  \\
			\hline
			NPTEL Industrial Engineering & 58 Mins & 37.65 & 15 Mins \\
			\hline
		\end{tabular}
	\end{center}
\end{table*}

	\begin{table*}[h]
		\begin{center}
			\renewcommand{\arraystretch}{2}
			\caption{Results of Automatic Speech Recognition for Curated Videos Using Vakyansh Indian English 700 Hrs \\}
			\begin{tabular}{ | c | c | c | c | c |  }
				\hline
				\textbf{ Video Type } & \textbf{Length} & \textbf{WER} & \textbf{Time Taken} \\
				\hline
				NPTEL English Literature & 55 Mins & 30.45 & 6 Mins \\
				\hline
				NPTEL English Literature & 50 Mins & 34.15 & 6 Mins \\
				\hline
				NPTEL Operating System & 55 Mins & 44.10  & 6 Mins  \\
				\hline
				NPTEL Language Processing & 58 Mins & 43.65 & 6 Mins \\ 
				\hline
				NPTEL Aritificial Intelligence & 52 Mins & 49.65 & 6 Mins  \\
				\hline
				NPTEL Industrial Engineering & 58 Mins & 41.65 & 6 Mins \\
				\hline
			\end{tabular}
		\end{center}
	\end{table*}§png§600§FALSE§

12§latex§\begin{table*}[h]
	\renewcommand{\arraystretch}{2}
	TABLE III \\ 
   Punctuation Correction Strategy \\
	\begin{center}
	\begin{tabular}{|l|l|}
		\hline
		\textbf{Original Sentence} &   Dr G Viswanathan, founder and chancellor, praised students.\\ \hline
		\textbf{Preprocessed}      &  dr g viswanathan, founder and chancellor, praised students.\\ \hline
		\textbf{Tokenized}         & dr g viswanathan founder and chancellor praised students \\ \hline
		\textbf{Output}            & EMP EMP COMMA EMP EMP COMMA EMP FULLSTOP  \\ \hline
	\end{tabular}
\end{center}
\end{table*}§png§600§FALSE§

12§latex§\begin{table*}[h]
\begin{center}
	\renewcommand{\arraystretch}{2}
	TABLE IV \\ Performance of different methods of Punctuation Restoration  \\
\begin{tabular}{ | c | c | c | c | c |  }
	\hline
    \textbf{ Paper } & \textbf{Method} & \textbf{Precision} & \textbf{Recall} & \textbf{F1} \\
	\hline
	Yi and Tao [8] & Encoder-decoder with self-attention & 76.7 & 69.6 & 72.9 \\
	\hline
	Tundik and Szaszak [5] & Character + Words Based BiLSTM & - & - & 62.9 \\
	\hline
	Yi et al [9] & Adversarial learning & 80.9  & 75.0 & 77.8 \\
	\hline
	\textbf{Nagy et al} [11] & BERT Based Punctuation & - & - & 79.8
   \\
   \hline
  \end{tabular}
\end{center}
\end{table*}§png§600§FALSE§

12§latex§\begin{table*}[h]
 	\begin{center}
	TABLE V \\ Precision, Recall and F1 score for each category of punctuation using different models. \\ \\ 
	\makegapedcells
	\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
		\hline
		\multirow{2}{*}[-5pt]{\textbf{Paper}} 
		& \multicolumn{3}{c|}{\textbf{Period}}  & \multicolumn{3}{c|}{\textbf{Comma}}  & \multicolumn{3}{c|}{\textbf{Question Mark} }\\
		\cline{2-10}
		& \textbf{\textit{P}} & \textbf{\textit{R}}  & \textbf{\textit{F1}} 
		    & \textbf{\textit{P}} & \textbf{\textit{R}}  & \textbf{\textit{F1}} & \textbf{\textit{P}} & \textbf{\textit{R}}  & \textbf{\textit{F1}} \\ \hline
		    
		 Yi and Tao [8] & \textit{82.5} & \textit{77.4}& \textit{79.9} 
		 & \textit{67.4} & \textit{61.1} & \textit{64.1} & \textit{80.1} & \textit{70.2}  & \textit{74.8}  
		 \\ 	\hline
		 Tundik and Szaszak [5]  & \textit{68.4} & \textit{77.4} & \textit{72.6} & \textit{62.4} & \textit{53.7} & \textit{57.7}  &
		 		\textit{64.1}   & \textit{54.3} & \textit{58.8} \\
		 			\hline
		 	Yi et al [9]   & \textit{87.3} & \textit{81.1} & \textit{84.1} & \textit{76.2} & \textit{71.2} & \textit{73.6}  &
		 \textit{79.1}   & \textit{72.7} & \textit{75.8} \\
		 \hline
		 \textbf{Nagy et al} [11] & \textit{84.4} & \textit{87.3} & \textit{85.8} & \textit{89.0} & \textit{93.1} & \textit{91.0}  &
		 \textit{73.5}   & \textit{66.7} & \textit{69.9} \\
		 \hline
	\end{tabular}
\end{center}
\end{table*}§png§600§FALSE§

**7. Conclusion and Future Work**

We got some good results in reference to the Word Error Rate in transcriptions. It can be further improved using good language models. The models can be further trained on high end GPU’s to create models specific to our college VIT.

**8**. **REFERENCES**

[1] M. Anusuya and S. K. Katti, “Speech recognition by machine, a review,” arXiv preprint arXiv:1001.2267, 2010.

[2] M. P. Lewis, G. F. Simons, and C. D. Fennig, “Ethnologue: languages of the world, dallas, texas: Sil international,” Online version: http://www. ethnologue. com, vol. 12, no. 12, p. 2010, 2009.

[3] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchin- sky, and R. Collobert, “Wav2letter++: A fast open-source speech recognition system,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 6460–6464.

[4] W. Garbe, “Symspell: 1 million times faster spelling correction and fuzzy search through symmetric delete spelling correction algorithm,” 2022. [Online]. Available: https://github.com/wolfgarbe/SymSpell

[5] M. Á. Tündik, B. Tarján, and G. Szaszák, “A low latency sequential model and its user-focused evaluation for automatic punctuation of asr closed captions,” Computer Speech & Language, vol. 63, p. 101076, 2020.

[6] J. Yi, J. Tao, Z. Wen, Y. Li et al., “Distilling knowledge from an ensemble of models for punctuation prediction.” in Interspeech, 2017, pp. 2779–2783.

[7] S. Kim, “Deep recurrent neural networks with layer-wise multi-head attentions for punctuation restoration,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 7280–7284.

[8] J. Yi and J. Tao, “Self-attention based model for punctuation prediction using word and speech embeddings,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 7270–7274.

[9] J. Yi, J. Tao, Y. Bai, Z. Tian, and C. Fan, “Adversarial transfer learning for punctuation restoration,” arXiv preprint arXiv:2004.00248, 2020.

[10] M. Federico, M. Cettolo, L. Bentivogli, P. Michael, and S. Sebasian, “Overview of the iwslt 2012 evaluation campaign,” in IWSLT- International Workshop onSpoken Language Translation, 2012, pp. 12–33.

[11] A. Nagy, B. Bial, and J. Ács, “Automatic punctuation restoration with bert models,” arXiv preprint arXiv:2101.07343, 2021.

[12] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, “Text summarization techniques: a brief survey,” arXiv preprint arXiv:1707.02268, 2017.

[13] A. A. Syed, F. L. Gaol, and T. Matsuo, “A survey of the state-of-the-art models in neural abstractive text summarization,” IEEE Access, vol. 9, pp. 13 248–13 265, 2021.

[14] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” arXiv preprint arXiv:1509.00685, 2015.

[15] S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, 2016, pp. 93–98.

[16] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization,” in International Conference on Machine Learning. PMLR, 2020, pp. 11 328–11 339.

[17] C. Zhu, R. Xu, M. Zeng, and X. Huang, “End-to-end abstractive summarization for meetings,” arXiv preprint arXiv:2004.02016, 2020.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002,

pp. 311–318.

[19] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, “Automatic text summarization: A comprehensive survey,” Expert Systems with Applications, vol. 165, p. 113679, 2021.

[20] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, “Text summarization techniques: a brief survey,” arXiv preprint arXiv:1707.02268, 2017.

[21] A. A. Syed, F. L. Gaol, and T. Matsuo, “A survey of the state-of-the-art models in neural abstractive text summarization,” IEEE Access, vol. 9, pp. 13 248–13 265, 2021.

[22] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” arXiv preprint arXiv:1509.00685, 2015.

[23] S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, 2016, pp. 93– 98.

[24] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization,” in International Conference on Machine Learning. PMLR, 2020, pp. 11 328–11 339.

[25] C. Zhu, R. Xu, M. Zeng, and X. Huang, “End-to-end abstractive summarization for meetings,” arXiv preprint arXiv:2004.02016, 2020.

[26] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[27] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, “Automatic text summarization: A comprehensive survey,” Expert Systems with Applications, vol. 165, p. 113679, 2021.

**Appendix**

(MOST IMPORTANT CODES ARE ONLY PASTED)

tfidf\_vect = TfidfVectorizer()

mat1 = tfidf\_vect.fit\_transform(data)

tfidf = pd.DataFrame(data=mat1.toarray(), index=list(doc.keys()), columns=tfidf\_vect.get\_feature\_names\_out())

tfidf

cos = cosine\_similarity(tfidf)

doc\_cos = pd.DataFrame(data=cos, index=list(doc.keys()), columns=list(doc.keys()))

doc\_cos.values[[np.arange(doc\_cos.shape[0])]\*2] = 0

doc\_cos

similarity\_matrix = doc\_cos.to\_numpy()

summarize\_text = []

sentence\_similarity\_graph = nx.from\_numpy\_array(similarity\_matrix)

scores = nx.pagerank(sentence\_similarity\_graph)

ranked\_sentence = sorted(((scores[i],s) for i,s in sentences.items()), reverse=True)

#print("Indexes of top ranked\_sentence order are ", ranked\_sentence)

for i in range(2):

summarize\_text.append(" ".join(ranked\_sentence[i][1]))

print("Summarize Text: \n", ". ".join(summarize\_text))

# SAMPLE CODE FOR LANGUAGE MODELS

with open(vocab\_file, encoding="utf-8") as vocab\_handle:

self.encoder = json.load(vocab\_handle)

self.decoder = {v: k for k, v in self.encoder.items()}

self.errors = errors # how to handle errors in decoding

self.byte\_encoder = bytes\_to\_unicode()

self.byte\_decoder = {v: k for k, v in self.byte\_encoder.items()}

with open(merges\_file, encoding="utf-8") as merges\_handle:

bpe\_merges = merges\_handle.read().split("\n")[1:-1]

bpe\_merges = [tuple(merge.split()) for merge in bpe\_merges]

self.bpe\_ranks = dict(zip(bpe\_merges, range(len(bpe\_merges))))

self.cache = {}

self.add\_prefix\_space = add\_prefix\_space

def train(self):

printer\_counter = 0

for epoch\_num in range(self.\_config.trainer.num\_epochs):

# Train loop

self.model.train()

pbar = tqdm(self.train\_loader)

for data in pbar:

self.optimizer.zero\_grad()

input\_bert\_text, class\_punctuation\_target = data

prediction\_punct, binary\_prediction\_punct = self.model(input\_bert\_text.to(self.device))

mask = ((class\_punctuation\_target == 0) & (np.random.rand(\*class\_punctuation\_target.shape) < .1)) | (class\_punctuation\_target > 0)

mask = mask.to(self.device)

not\_a\_word\_mask = (class\_punctuation\_target == -1).to(self.device)

word\_mask = ~not\_a\_word\_mask

class\_punctuation\_target[not\_a\_word\_mask] = 0

losses = self.criterion(prediction\_punct.reshape(-1, self.\_config.model.num\_classes),

class\_punctuation\_target.to(self.device).reshape(-1))

mask = word\_mask \* mask

# losses = mask.view(-1).to(self.device) \* losses

# loss = losses.sum() / mask.sum()

loss = losses.mean()

loss.backward()

import regex as re

import time

import datetime

from pathlib import Path

import subprocess

import os

import shutil

import soundfile as sf

# import wav2vec2\_transcriptions as w2v2

import w2v2\_transcriptions as w2v2\_asr

from timeit import default\_timer as timer

import spell\_correction\_using\_symspell as spcheck

# import punctuation\_restoration as punct\_rest

path\_base = "/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/AUDIO\_FILES/" #Original speech/audio files folder

audio\_report = "/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/EXTRACTED\_TRANSCRIPTIONS/" #This is the folder where your report will be stored

path\_converted\_audio = "/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/converted\_files/" #This is the temporary folder for converted audio files

resampled\_folder = "/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/resampled\_files/" #This is the folder for the resampled audio files

Path(audio\_report).mkdir(parents = True, exist\_ok = True) #This creates the reports folder

Path(path\_converted\_audio).mkdir(parents = True, exist\_ok = True) #This creates the folder for converted audio files

Path(resampled\_folder).mkdir(parents = True, exist\_ok = True) #This creates the folder for resampled audio files

print("STARTING THE PROCESS")

start = timer()

report\_path = w2v2\_asr.speech\_to\_textfunc(path\_base,path\_converted\_audio,resampled\_folder,audio\_report)

end = timer()

time\_asr = end-start

spell\_chkd\_asr = []

# start = timer()

# for i in report\_path:

# print(i)

# spell\_chkd\_asr.append(spcheck.spell\_correct(i,audio\_report))

# end = timer()

# time\_spell\_checking = end-start

# start = timer()

# for i in spell\_chkd\_asr:

# punct\_rest.punctuate(i,audio\_report)

# end = timer()

# time\_punctuation = end-start

# print("TIME FOR ASR",time\_asr)

# print("TIME FOR SPELL CHECK",time\_spell\_checking)

# print("TIME FOR PUNCTUATION",time\_punctuation)

# print("TOTAL TIME TAKEN",(time\_asr+time\_spell\_checking+time\_punctuation))

from transformers import pipeline

# summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from\_pretrained("facebook/bart-large-cnn")

model = AutoModelForSeq2SeqLM.from\_pretrained("facebook/bart-large-cnn")

# ARTICLE = """ New York (CNN)When Liana Barrientos was 23 years old, she got married in Westchester County, New York.

# A year later, she got married again in Westchester County, but to a different man and without divorcing her first husband.

# Only 18 days after that marriage, she got hitched yet again. Then, Barrientos declared "I do" five more times, sometimes only within two weeks of each other.

# In 2010, she married once more, this time in the Bronx. In an application for a marriage license, she stated it was her "first and only" marriage.

# Barrientos, now 39, is facing two criminal counts of "offering a false instrument for filing in the first degree," referring to her false statements on the

# 2010 marriage license application, according to court documents.

# Prosecutors said the marriages were part of an immigration scam.

# On Friday, she pleaded not guilty at State Supreme Court in the Bronx, according to her attorney, Christopher Wright, who declined to comment further.

# After leaving court, Barrientos was arrested and charged with theft of service and criminal trespass for allegedly sneaking into the New York subway through an emergency exit, said Detective

# Annette Markowski, a police spokeswoman. In total, Barrientos has been married 10 times, with nine of her marriages occurring between 1999 and 2002.

# All occurred either in Westchester County, Long Island, New Jersey or the Bronx. She is believed to still be married to four men, and at one time, she was married to eight men at once, prosecutors say.

# Prosecutors said the immigration scam involved some of her husbands, who filed for permanent residence status shortly after the marriages.

# Any divorces happened only after such filings were approved. It was unclear whether any of the men will be prosecuted.

# The case was referred to the Bronx District Attorney\'s Office by Immigration and Customs Enforcement and the Department of Homeland Security\'s

# Investigation Division. Seven of the men are from so-called "red-flagged" countries, including Egypt, Turkey, Georgia, Pakistan and Mali.

# Her eighth husband, Rashid Rajput, was deported in 2006 to his native Pakistan after an investigation by the Joint Terrorism Task Force.

# If convicted, Barrientos faces up to four years in prison. Her next court appearance is scheduled for May 18.

# """

word\_string = ""

hypo = open("/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/EXTRACTED\_TRANSCRIPTIONS/TheAgeofChaucer1649579702spell\_corrected\_and\_punctuated.txt")

for row in hypo:

word\_string += (row+"\n")

length = len(word\_string)

print("HELLO WORLD")

print(length,(length//3)\*2)

import nltk

# nltk.download('punkt')

sentences = nltk.tokenize.sent\_tokenize(word\_string)

length = 0

chunk = ""

chunks = []

count = -1

print("STARTING SUMMARIZATION")

for sentence in sentences:

count += 1

combined\_length = len(tokenizer.tokenize(sentence)) + length # add the no. of sentence tokens to the length counter

if combined\_length <= tokenizer.max\_len\_single\_sentence: # if it doesn't exceed

chunk += sentence + " " # add the sentence to the chunk

length = combined\_length # update the length counter

# if it is the last sentence

if count == len(sentences) - 1:

chunks.append(chunk.strip())

else:

chunks.append(chunk.strip())

# reset

length = 0

chunk = ""

chunk += sentence + " "

length = len(tokenizer.tokenize(sentence))

len(chunks)

inputs = [tokenizer(chunk, return\_tensors="pt") for chunk in chunks]

for input in inputs:

output = model.generate(\*\*input)

print(tokenizer.decode(\*output, skip\_special\_tokens=True))

# print(summarizer(word\_string, max\_length=(length//3)\*2, min\_length=(length//3)+20, do\_sample=False))

import pkg\_resources

from symspellpy import SymSpell, Verbosity

audio\_report1 = "/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/EXTRACTED\_TRANSCRIPTIONS/" #This is the folder where your report will be stored

import os

from pathlib import Path

sym\_spell = SymSpell(max\_dictionary\_edit\_distance=4, prefix\_length=7)

dictionary\_path = pkg\_resources.resource\_filename(

"symspellpy", "frequency\_dictionary\_en\_82\_765.txt"

)

sym\_spell.load\_dictionary(dictionary\_path, term\_index=0, count\_index=1)

hypo = '/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/EXTRACTED\_TRANSCRIPTIONS/test\_trans\_facebook\_1.txt'

# gd\_truth = open('/home/mrigank/Public/WINTER\_SEMESTER\_2021\_22/CSE1901\_TECHNICAL\_ANSWERS\_FOR\_REAL\_WORD\_PROBLEMS/CSE1901\_PROJECT/YT\_TRANSCRIPTIONS/TheAgeofChaucer.txt','r')

# source ~/python\_virtualenv/lec\_sum/bin/activate

def spell\_correct(path\_asr,audio\_report=""):

hypothesis\_string = ""

if audio\_report == "":

audio\_report = audio\_report1

hypo = open(path\_asr)

for row in hypo:

li\_word = row.split()

corrected\_row\_string = ""

for i in li\_word:

input\_term = i

suggestions = sym\_spell.lookup(input\_term, Verbosity.CLOSEST, max\_edit\_distance=2, include\_unknown=True, ignore\_token=r"\w+\'\w+"

)

word\_sel = ""

for sug in suggestions:

word\_sel = sug.term

break

corrected\_row\_string += word\_sel + " "

hypothesis\_string += corrected\_row\_string + "\n"

# print(path\_asr.rfind('/'),len(path\_asr))

file\_name = path\_asr[path\_asr.rfind('/')+1:path\_asr.rfind('.')]

file\_name = file\_name+"spell\_corrected.txt"

file1 = open(audio\_report+file\_name, 'w+')

file1.write(hypothesis\_string)

file1.close()

return (audio\_report+file\_name)

# spell\_correct(hypo)

# input\_term = "chaucer's"

# suggestions = sym\_spell.lookup(

# input\_term, Verbosity.CLOSEST, max\_edit\_distance=2, include\_unknown=True, ignore\_token=r"\w+\'\w+"

# )

# sug = list(suggestions)

# print(sug[0])

# # display suggestion term, edit distance, and term frequency

# for suggestion in suggestions:

# print(suggestion[0])