Advances in Science, Technology and Engineering Systems Journal
Vol. 6, No. 2, 526-540 (2021)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTES Journal
ISSN: 2415-6698

A Model for the Application of Automatic Speech Recognition for Generating Lesson Summaries

Phillip Blunt[*], Bertram Haskins

Department of Engineering, The Built Environment and Technology, Nelson Mandela University, Port Elizabeth, 6031, South Africa

ARTICLE INFO

Article history:
Received: 23 October, 2020
Accepted: 20 February, 2021
Online: 17 March, 2021

Keywords:
Feature extraction
Machine learning
Continuous speech recognition
Keyword spotting
Transcription
Education
Noise management
Lesson summary

ABSTRACT

Automatic Speech Recognition (ASR) technology has the potential to improve the learning experience of students in the classroom. This article addresses some of the key theoretical areas identified in the pursuit of implementing a speech recognition system, capable of lesson summary generation in the educational setting. The article discusses: some of the applications of ASR technology in education; prominent feature extraction and speech enhancement techniques typically applied to digital speech; and established neural network-based machine learning models capable of keyword spotting or continuous speech recognition. Following the theoretical investigation, a model is proposed for the implementation of an automatic speech recognition system in a noisy educational environment to facilitate automated, speech-driven lesson summary generation. A prototype system was developed and improved based on this model, ultimately proving itself capable of generating a lesson summary intended to bolster students' secondary contact with lesson content. This topic-oriented lesson summary provides students with a lesson transcript, but also helps them to monitor educator-defined keyword terms, their prevalence and order as communicated in the lesson, and their associations with educator-defined sections of course content. The prototype was developed using the Python programming language with a modular approach so that its implemented Continuous Speech Recognition system and noise management technique could be chosen at run-time. The prototype contrasts the performance of CMUSphinx and Google Speech Recognition for ASR, both accessed via a cloud-based programming library, and compared the change in accuracy when applying noise injection, noise cancellation or noise reduction to the educator's speech. Proof of concept was established using the Google Speech Recognition System, which prevailed over CMUSphinx and enabled the prototype to achieve 100,00% accuracy in keyword identification and association on noise-free speech, contrasted with a 96,93% accuracy in keyword identification and association on noise-polluted speech using a noise-cancellation technique.

1 Introduction

Student lesson summaries are a valuable resource for allowing students to focus on the key points of a lesson, boosting secondary contact with lesson content. They allow students to realise which aspects of a lesson may be more important and streamline the study process for courses which employ both formative and summative assessments. Formulating a lesson summary from notes and lesson content alone can be challenging to students especially when covering larger segments of course content over a short period of time. Students are also not all equally capable of creating their own lesson summaries and often rely on available course material and notes made during their lessons as study material. These challenges are also further compounded by a lack of context whenever students attempt to summarise a lesson without integrating with the theoretical focus of the classroom teaching session. Toward providing a solution to these challenges of lesson summary generation; this work is an extension of the paper presented at The 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC) wherein a model was proposed for automated lesson summary generation in a noisy educational environment using keywords in the educator's speech as the prompting mechanism for summary of key points [1]. This extension reintroduces the proposed model, along with the theoretical background it is based in, reinforced by a proof of concept prototype system for lesson summary generation which is used to demonstrate the model's application.

[*]Corresponding Author: Phillip Blunt, Email: s213308762@mandela.ac.za

Automatic speech recognition (ASR) has been applied in assistive technologies for tasks such as closed captioning, voice search, command detection as well as keyword identification. It therefore stands to reason that these applications of the technology may be applied to the voice of the educator in the educational setting. This work posits that it is possible to generate a lesson summary, by transcribing the voice of the educator during a theory lecture and identifying keywords based on the concepts of the course material being taught. In support of this theory, the objective of this work is to establish an abstract model for the application of ASR technology in the educational setting for generating lesson summaries. Section 2 provides the theoretical base for the model, focusing in three broad areas of theory. First, a number of applications of ASR technologies in education are discussed, particularly as they pertain to disabled learners and distance-based education. Second, recent trends in statistical models used to perform ASR and the underlying machine learning techniques used to design and train these models are addressed. In addition, voice enhancement and noise management strategies are addressed to account for noise pollution in the educational environment known to negatively affect the accuracy of ASR technologies and resulting transcriptions. The model itself is presented in Section 3 and the proof-of-concept prototype is discussed in Section 4. The prototype is evaluated in Section 5, with a discussion of test results in Section 6 and the study's conclusion in Section 7.

2 Background

This section provides an overview of work related to the concepts required to establish and propose the model for lesson summary generation in this work. To that end, Section 2.1 discusses the utility of ASR as applied to an educational context. The topics of feature extraction and noise cancellation are discussed in Section 2.2.2, as these techniques may be used to improve the reliability of speech features in the educator's speech during lesson transcription. Having clearly detectable features is critical in training a machine learning model to recognise, decode and transcribe speech. Neural networks are a widely used technique to train such a machine learning model and Section 2.3 presents a discussion on these techniques, revealing the trends in the improvement of the technology in recent years. The cumulative knowledge addressed in these theory sections provide the theoretical basis for the model and its application.

2.1 Automatic Speech Recognition in Education

ASR is widely known through commercial applications such as Amazon Alexa, Apple Siri, Microsoft Cortana and the Google Assistant [2]. These products enhance the productivity of their users by providing a means of interaction with a variety of applications. The use of ASR is, however, not limited to applications of convenience; it has also found purchase in the domain of education. ASR is one of the key interfaces that humans use to engage with machines in information technology, research in the field has important significance and the interface itself has wide value in application [3, p.84].

A broad overview is provided by 4 of both the underlying literature and experiments conducted with regard to ASR in the field of education. The technology not only holds the promise of helping students surmount the challenges associated with reading, writing and spelling [4, p. 66], but also provides a facility for teaching staff to improve their pedagogical approach [4, p. 69]. A further application of this technology in the educational space is that it holds the promise of improving the interactions of deaf and second language speakers in the classroom [4, p. 66]. Furthermore, many of the studies included in the work by [4] have found utility as an emancipatory tool for those with physical and/or learning disabilities, as it allows them to write tests and complete projects by means of narration; tasks which would otherwise require the services of a human transcriber. With regard to deaf or hard of hearing students, such an ASR intervention could be invaluable if employed in a classroom environment to provide live captioning or to provide a lesson transcript after the fact. This may also alleviate their dependence on a sign language interpreter, transcriber or hearing aid. It has also been shown that it is possible to integrate an ASR system into the mathematics teaching process at a primary school level [5]. In the work conducted by 5, a voice-activated e-learning prototype was used and it was demonstrated that the use of such a tool could facilitate the learning experience at a primary school level, but also demonstrated feasibility up to a tertiary level; being especially helpful to students who have disabilities, learn on-line or are studying in a second language.

Beyond its utility in helping students with disabilities and in overcoming language barriers, ASR is also being applied to aid in other tasks related to the daily activities of students; including lesson reflection, group discussions and oral presentations [4, p. 66]. ASR can aid these activities by allowing students to have access to a lesson transcript, which facilitates a streamlined approach towards note-taking. It has been shown [6] that ASR technology has definite application in synchronous (real-time) cyber classrooms; an approach that has been pulled into the limelight with the advent of the COVID-19 pandemic [7]. Although such a live approach has been shown to be impeded by issues such as latency and bandwidth, resulting in students missing portions of the lecture [6, p. 367-368], the application of ASR could quite easily alleviate this with the lecturer providing a transcription of the lesson, generated by ASR from a full recording of their speech, after the lesson has been delivered.

ASR systems have also been applied to Elicited Oral Response (EOR) testing. EOR is employed to assess the speaking ability of an examinee by having them listen to a phrase and then restate the phrase to the examiner [8, p. 602]. The application of ASR allows such a test to be conducted automatically, with little to no need for a human examiner. The feasibility of such an ASR-based EOR testing process has been demonstrated [8]. It was found that it is a suitable tool for assessing the validity of content, and is able to do so reliably and in a practical fashion. It represents a means by which to support low-stakes decision making, especially when applied to second language learning. All of these technologies, however, have their downsides. The transcripts generated with an ASR-based tool tend to contain many punctuation errors or to eschew punctuation altogether. It also does not account for any recognition errors which may occur as a result of redundancies in repetitive speech [4, p. 68]. These shortcomings require manual human intervention or a third-party grammar analysis application to correct. Furthermore, a common criticism of using ASR-based technology in a noisy environment such as a classroom, is that the accuracy of the ASR process may be impacted. There are, however, means by which these noise artefacts could be overcome, minimised or filtered out, allowing the required high rate of accuracy to be maintained.

2.2 *Feature Extraction and Speech Enhancement Techniques for Robust Speech Recognition*

This section focuses on prominent feature extraction and speech enhancement techniques such as noise cancellation and noise reduction; with the noise cancelling techniques serving to augment or compliment feature extraction techniques that are not as noise robust.

2.2.1 *Feature Extraction Techniques used in ASR systems*

Although ASR consists of many steps, one of the most important is undoubtedly that of feature extraction; as this step is used to highlight which components of an input speech signal will serve to support the *recognition* aspect of automated speech recognition. For speech-based audio data to be employed as meaningful data, whether for training purposes or in an active ASR system, it needs to be transformed into a less abstract representation which brings to fore the distinguishable components of speech in the input audio signal. The work done in 9, p. 3 distinguishes between two types of features; temporal features (e.g. short time energy and auto-correlation), which exist within the time domain and spectral features (e.g. fundamental frequency and spectral flux) which exist within the frequency domain of a speech signal. A spectrogram is a time-frequency representation of speech data. It is arguably the fundamental feature extraction method, performed by applying the Fast Fourier Transform (FFT) to the speech signal [10, p. 4525] to transform speech energy through time into frequency estimations through time. As there is a need to highlight the frequency intensity related to speech signals, many feature extraction techniques, related to speech processing, are spectral in nature. The words in most languages are built from smaller components of speech, known as phonemes and the various phonemes consist of distinct formants (fundamental frequencies), evident in their pronunciation. Formants are defined [11, p. 5176] as being created by the resonance of the vocal tract and recognisable as the spectral peaks on the frequency-time spectrum of speech. Therefore, techniques used for spectral feature extraction are highly applicable to ASR systems, allowing them to distinguish between the various formants at their distinct spectral peaks.

Another approach is to apply Linear Predictive Coding (LPC) to create observation vectors, based on the frame-based analysis of speech signals. These vectors may provide an estimation of the poles of the vocal transfer function [12, p .495]. During the process of performing LPC feature extraction, a signal is run through a pre-emphasis process to reduce the occurrence of audio pop at the beginning and end of each frame and to reduce signal discontinuity between frames. The first of these issues is addressed by the application of frame blocking and windowing. This is then followed by an auto-correlation analysis, applied window-wise, during which the LPC coefficients are derived as the observational vectors [12][p. 495]. To counter the lack of robustness of LPC with regard to noise, which may cause interference in the calculation of the coefficients, another spectral-based feature extraction technique, Relative Spectral Filtering (RASTA), may be applied to extract features from the spectogram. The intent of RASTA is to enhance the speech characteristics of the signal by means of the reduction of unwanted and additive noise [12, p. 495]. As part of this technique, a spectral analysis is performed, after which static non-linearities are compressed, a filter is applied based on linear band trajectory in the cepstral or log spectral domain, and then finally, the static non-linearities are decompressed, resulting in the set of RASTA features [12, p. 495]. As stated by [12, p. 496] RASTA-based feature extraction finds purchase where speech recognition needs to be performed in a noisy environment.

Another very widely used feature extraction technique is the Mel-Frequency Cepstral Coefficient (MFCC). This technique focuses on audio frequencies in the range 300Hz to 3400Hz, the critical frequency range of the human vocal tract interpretable by the human ear [12, p. 495; 11, p. 5176]. MFCC is associated with a very efficient method of calculation which follows a similar approach to LPC, in that it involves pre-emphasis of the speech signal, frame blocking and windowing [12, p. 495]. After the windowing process, the FFT is applied and the absolute values it returns are placed in a Mel-filter bank; the log of the filter bank values is calculated and the final MFCC feature vectors are created by applying the discrete cosine transform to each Mel-filter bank. Because it relies on auto-correlaton analysis, MFCC shares the trait with LPC that it is not noise robust [13, p. 358]; although there are many MFCC variants, each with their own improvements and compromises [14]. Other feature extraction techniques include Perceptual Linear Predictive Coefficients (PLP), which are often used in conjunction with RASTA for improved performance; Wavelet-based features; and Linear Predictive Cepstral Coefficients (LPCC), an addition to LPC [13–16]. The work performed in [3, p.83] contrasts LPCC and MFCC, demonstrating that LPCC generally results in lower accuracy but has a faster computation rate while MFCC is slower to compute, but often results in improved recognition accuracy.

2.2.2 *Speech Enhancement Techniques for improved ASR*

The effect of noise has always been a major consideration when implementing ASR system, as a noisy input signal may interfere with the feature extraction process. This may yield unreliable speech features, which in turn leads to a low level of accuracy for the underlying ASR model. There are various major sources of noise, namely background noise from a noisy environment, echoes resulting from recording in confined spaces, feedback resulting from two-way communication when a loudspeaker is too close to the recording device, the background hum caused by an amplifier, quantisation noise resulting from analog to digital conversion, noise resulting from the loss of signal quality when compression is applied, and finally, the distorted signal resulting from reverb, which can also be considered noise pollution [11, p. 5176; 17, p. 318]. Within these various noise sources, noise may be classified according to the following types, which can be mitigated via noise cancellation techniques, namely narrow-band noise, coloured noise, white noise, transient noise pulses and impulse noise [11, p. 5177]. The ultimate goal of a noise cancellation technique is to suppress or de-emphasise the noise components within an input speech signal [11, p. 5178]. Speech enhancement is the process of enriching the spectral characteristics of a speech signal to make it easier to recognise by machine. This process incorporates noise cancellation techniques, but may also transform the speech signal so that the features within it are more distinguished. 9, p. 2 discuss the principles behind speech enhancement, stating that the performance of such a system is measured based on the quality (detected by human ear) of the enhanced speech and the residual noise level that remains in the speech after enhancement. The main objective of a speech enhancement strategy is to remove any additive noise that exists within a speech signal as a result of the recording being performed in a noisy environment [9, p. 8]. The terms speech enhancement and noise cancellation have become largely synonymous, but 9, p. 9 make the point that feature selection is optimised by ”selecting the most uncorrelated features” which often determines the effectiveness of the overall speech enhancement strategy.

Noise cancellation techniques fall under two broad categories, namely linear filtering and adaptive filtering [18; 11; 17]. These noise cancellation techniques may occur in the frequency domain, the time domain or in both [11, p. 5179]. Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) are two types of linear filters [11, p.5178]. The purpose of linear filtering is to remove all frequencies which exist outside of the desired frequency domain by moving linearly along the time domain. In the frequency domain these filtering techniques fall under four categories, namely low-pass, band-pass, band-stop and high-pass filters [11, p.5178]. A combination of these filters may be used to remove all frequencies existing outside of the interpretable range of the human vocal tract. This may be done by applying a high-pass filter to remove frequencies below 300Hz and a low-pass filter to remove frequencies above 3400Hz. This approach, however, will not remove noise within the range of the human vocal tract and may also be problematic in noisy environments because the characteristics of the noise may vary (in intensity) over time and as a result it may not be possible to predict the position in the audio stream at which the noise will occur due to its non-stationary nature [18, p. 336; 11, p. 5178]. Linear filtering has utility in cases where the noise levels are more predictable, such as with amplifier or quantisation noise.

Another form of filtering, the adaptive filter, is based on the mathematical principle of cancellation. This process combines two signals to remove the noise from the original. In such an Adaptive Noise Canceller (ANC), the original signal contains the desired speech to serve as input to an ASR process, but also the noise, which may negatively impact the accuracy of recognition. The second signal serves as a representation of the noise and is adaptively filtered from the original speech signal. This second signal is then subtracted from the original signal [11, p. 5178]. This kind of approach accounts for the dynamic nature of a signal containing speech and other audio. This presents an approach where the parameters and band-pass type are adjusted automatically, depending on the signal, rather than relying on pre-set parameters and a specific band-pass type [11, p. 5180]. This approach is conducted by performing audio framing, after which a unitary transform of the time domain for every frame to the given transform domain is performed. This allows filtering to be applied to individual frames; after which the frames are returned to the time domain by applying the inverse unitary transform. As a final step, the frames are converted back into a congruent audio file, representing the noise to be subtracted from the original audio [11, p. 5179]. An alternative to focusing on the time domain is to approach the problem from the frequency domain. Examples of adaptive filters are the Weiner and Kalman filters and the Recursive Least Squares (RLS) algorithm [17, p. 318]. A Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) or Karhunen-Loève Transform may be used to perform frequency domain transforms [18, p. 336]. Of these approaches, the computational efficiency of DFT makes it the most popular [11, p. 5179]. By applying these techniques to an appropriately selected transform domain, better separation may be achieved between the speech and noise signals. This could result in improved filter estimation which may yield superior speech enhancement performance.

This section has discussed a number of feature extraction techniques and the importance of their appropriate selection, especially in noisy environments. As demonstrated in [19], a feature extraction technique can be chosen to work in conjunction with its underlying ASR model along with a speech enhancement strategy for improved ASR performance.

2.3 *Machine Learning Techniques for Automatic Speech Recognition*

Hidden Markov Models (HMM) were widely used in the early development of ASR systems. The HMM applies probability theory to track the likelihood of the phonetic state transitions within words based on spectral templates of phonetic units which are decoded from the speech signal to predict voiced utterances. This process was eventually augmented through the addition of a Gaussian Mixture to each state of an HMM to model the short-time phonetic units. This combined process is referred to as a Gaussian Mixture Model Hidden Markov Model (GMMHMM) [20]. The advent of faster computer processors has made it feasible to train Deep Neural Network (DNN) models for speech recognition. DNN-based models deal very well with the high dimensionality of speech data, using fewer parameters to optimise [21] and have been shown to outperform GMMHMMs [22] leading to the widespread adoption of DNN architectures for the purpose of performing machine learning in ASR. There have been many major improvements since the first implementations of DNNs for ASR. Two of these improved architectures, namely Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are discussed in the following subsections.

2.3.1 *Convolutional Neural Networks*

A CNN depends on two additional logical layers for performing speech recognition, namely convolutional layers, which act as localised filters, and max pooling layers, which normalise spectral variation[20]. The series of filters in a convolutional layer are applied in an over-lapping fashion across acoustic frames that overlap in time over the entire input space and are often referred to as feature detectors [23, p 1534; 20]. By modelling the associations between the frequency and time domains, and using the local filtering and shared weights of the convolutional layers, a CNN maintains the correlations between these domains and provides a superior result to the input mapping of DNNs [21]. Varying speaking styles also present a challenge to traditional DNNs as they are not inherently designed to model the translation invariance resulting from formant frequency shifts [21]. Convolutional layers combat this through the use of local filtering and the shared weights. The resulting translation invariance improves the robustness of the model on diverse speech signals by allowing speech features to be detected regardless of their location within local input spaces.

The convolutional layers of a CNN work in conjunction with the max-pooling layers. Max-pooling layers are used to reduce the dimensionality of the resulting convolutions by ensuring maximum filter activation at varying points. This is done to reduce the dimensionality of the convolutions [20]. The performance of the speech recognition task, using CNNs, is also improved by performing pooling in frequency and in time, yielding a robustness towards speaking rate [21]. Convolutional and max-pooling layers may also be applied in alternating pairs to further reduce dimensionality. This improves performance in fully connected hidden layers with fewer trainable parameters. This added robustness towards variations in speech styles, provided by the pairing of convolution and max-pooling layers, allow a CNN to learn the acoustic features for various classes, such as speaker, phoneme and gender [23, p. 1534]. This is another feature which adds to the superiority of CNNs over DNNs for analysing speech signals. Studies have also shown that it is possible to add multiple channels of features, like those from a cochleogram and a spectogram, as inputs to a CNN. This approach allows the CNN to learn from multiple sets of features simultaneously, providing improved performance over learning from a single channel [10].

2.3.2 *Recurrent Neural Networks*

RNNs are very useful for language modelling, but are also capable of performing predictions with regard to the likelihood of a feature by making an association based on previously identified spectral speech features. This makes them highly capable of predicting future words or phonetic units based on previously observed words or phonetic states. RNNs make use of Long Short Term Memory (LSTM) cells to keep track of any associations identified in the previous layer with the current layer. RNNs have been employed [24; 25] for creating robust speech recognition models and these approaches have been improved upon [26] by implementing a light gated architecture. It has also been shown [27] that it is feasible to apply convolution and max pooling as inputs of RNN layers to perform local filtering and pooling. The condensed features are then passed to RNN layers, which make use of LSTM cells or Gated Recurrent Units (GRU) to maintain the contextual associations between features through time. HMMs have also been combined with convolution and LSTM to tie phonetic state transitions for speech recognition [28].

3 Model Overview

This section reintroduces the proposed model, which has been updated since its inception, initially published in [1]. This model is theorised through background literature review and exploration; then reinforced by a proof of concept prototype implementation and the experimentation performed to evaluate it. Figure 1 illustrates the model using a process flow diagram which demonstrates the sequence of processes applied and data flow between them that are applied to generate a topic-oriented lesson summary.

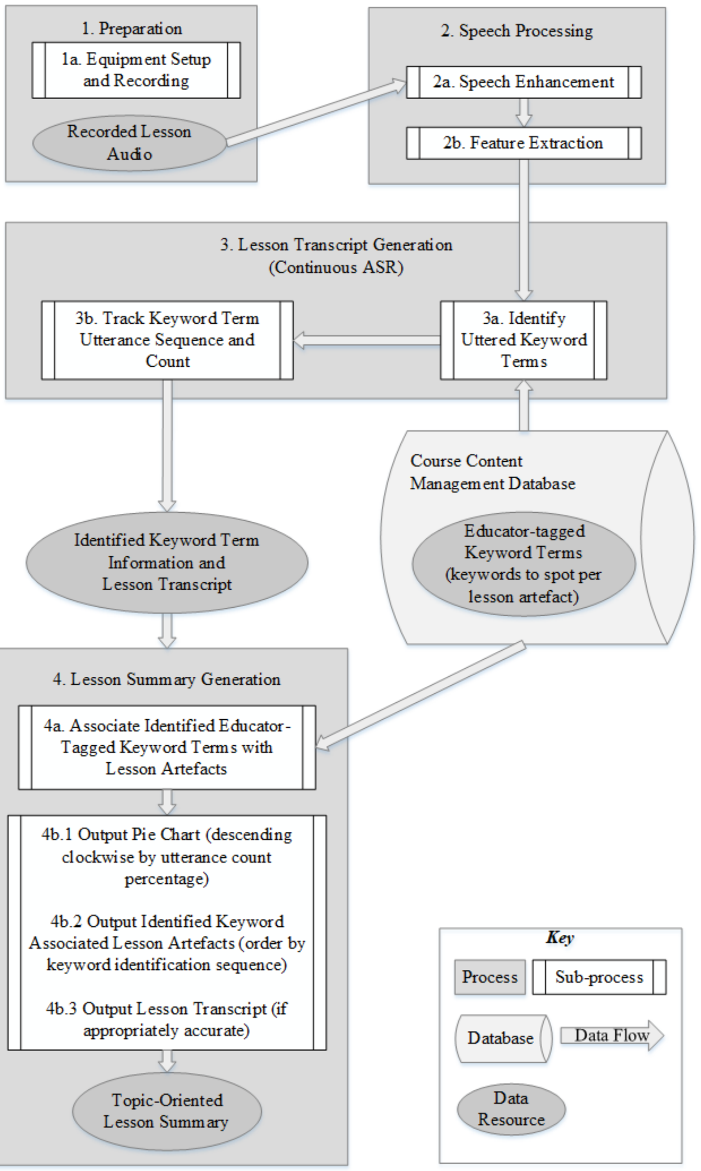
1. Preparation
1a. Equipment Setup and Recording
Recorded Lesson Audio
2. Speech Processing
2a. Speech Enhancement
2b. Feature Extraction
3. Lesson Transcript Generation (Continuous ASR)
3b. Track Keyword Term Utterance Sequence and Count
3a. Identify Uttered Keyword Terms
Course Content Management Database
Educator-tagged Keyword Terms (keywords to spot per lesson artefact)
Identified Keyword Term Information and Lesson Transcript
4. Lesson Summary Generation
4a. Associate Identified Educator-Tagged Keyword Terms with Lesson Artefacts
4b.1 Output Pie Chart (descending clockwise by utterance count percentage)
4b.2 Output Identified Keyword Associated Lesson Artefacts (order by keyword identification sequence)
4b.3 Output Lesson Transcript (if appropriately accurate)
Topic-Oriented Lesson Summary
Key
Process
Sub-process
Database
Data Flow
Data Resource

Figure 1: Proposed model for the application of an ASR system used to facilitate topic-oriented lesson summary generation in a noisy educational environment

The subsections that follow discuss the four main processes involved in implementing the proposed model to generate a topic-oriented lesson summary, driven by known keyword terms uttered by the educator during a recorded lesson.

3.1 *Preparation*

To capture the speech of the educator throughout the lesson, a microphone, connected to a recording device is required. There are a number of microphone options, widely available from commercial retailers. This section discusses two variations of microphone setup for use in conjunction with the proposed model. These are addressed with regard to ease and suitability of use and constraint on budget, as many of these hardware components are expensive and are not all appropriate for a lesson recording setup. There are a number of microphone products to choose from and their quality and capability of audio capture is what sets them apart. Seemingly, the most professional, and consequently the most expensive, are lapel microphones, often used in stage performances and business presentations. Lapel microphones provide a wireless, high fidelity (16000Hz) audio capture solution for the educator, which promotes freedom of movement. These microphones are typically fixed onto the speaker's collar and plugged into a transmitter usually attached at the waist of the speaker, which transmits the stream of their speech signal to a soundboard of sorts (which could be be integrated into a computer or some other audio mixer) for playback and storage. Lapel microphones are intended to capture the primary speaker's voice, with low gain, so as not to capture background noise or over-amplify the speech.

In the medium price range, handheld microphones, with wired and wireless variants, are a more affordable solution for recording the educator's speech during the lesson. Wireless options can still be expensive, but promote freedom of movement compared to their mounted or cabled counterparts. These microphones also transmit the stream of captured speech directly to an audio mixer or computer for storage and playback. Wireless recording solutions provide the most ease of use for the educator or a secondary speaker. The cheapest option for voice capture is the stereo headset, often used for voice communication while gaming or with online voice communication applications like Zoom and Skype. While stereo headsets can be bought cheaply, wireless variants are also more expensive and equally bulky, since the microphone cannot usually be detached from the headset itself and still function. Headsets are also only designed to capture the speech of the primary speaker and cabled variants are more suitable for use while seated, once again limiting the movement of the speaker unless a wireless headset can be used.

Each of these microphone options are also available with hardware-integrated noise cancellation. This additional feature can be costly, but can also remove the need for additional processing for noise management as the lesson is recorded. The choice of which recording devices to use should be made based on budget, fitness of purpose and requirement as well as convenience of use; ideally with a vision of high audio fidelity for a real-world classroom implementation, with the lowest permissible functional costs for testing purposes.

To facilitate voice processing and speech enhancement, discussed in the next section, a combination of microphones can be used. In this case, the primary microphone is used to record the speech of the educator or a student, should they have a question (to be transcribed), in which case they would need to speak into the educator's microphone, with a secondary microphone used to record environmental noise. A mounted wide-band microphone is the cheaper solution to consider for recording environmental noise during the lesson and if appropriately positioned, away from the educator, toward the back of the classroom, will capture noise pollution in the classroom without explicitly recording too much of the educator's speech. These microphones need to record in parallel so that their recordings can be easily aligned and used in combination with a chosen speech enhancement technique whereby the primary recording of the educator's speech is enhanced using the secondary recording of environmental classroom noise to 'cancel' noise captured in the educator's speech signal by the primary microphone.

3.2 Speech Processing

Once all the speech of the educator has been recorded for analysis after the lesson using the equipment setup and an appropriate recording method mentioned in the previous section, the digital speech must be processed to manage environmental noise. The primary requirement of this recording is that it must be of the waveform file format (file extension .wav); the raw, uncompressed audio format that ASR systems are typically developed and trained to be able to recognise speech from using an appropriate feature extraction technique. The quality of the captured audio must be considered in terms of its cost effectiveness. Improved audio quality results in larger waveform audio files as the range of captured frequencies as well as the bit depth and sampling rate increases. These factors contribute to the file size of the lesson audio, and should be considered carefully, especially if a cloud speech recognition service will be used to to transcribe lesson audio, as the file will need to be uploaded for recognition which will take more time if large audio files are uploaded. If a cloud based ASR solution is used to transcribe the recorded lesson, the audio may need to be segmented in overlapping windows of appropriate duration before the ASR system will transcribe the audio due to file size limitations put in place by the ASR service. Resulting segments of lesson transcript will also need to be aligned according to this overlap. In addition, should the audio need to be stored on a web server to be made available to students for review purposes, a large file size will also contribute to storage costs, data usage and buffer time.

Classrooms are notoriously noisy environments owing to the number of students, inevitable chatter, chairs shuffling, corridor activity and a myriad of other possible noise generating events. For this reason, speech enhancement, wherever possible should not be overlooked when implementing speech recognition technology in the educational environment. In the proposed model, speech enhancement is an optional process within the Speech Processing block, and will require the application of a linear or adaptive filter on the recorded lesson audio should this option be taken. This sub-process is optional because if the chosen ASR model uses a noise robust feature extraction technique, applies its own speech enhancement technique or if a noise cancelling microphone is used, then there may not be a need for additional processing for noise management. Over application of speech enhancement can also degrade the educator's speech to a point that it is no longer recognisable by ASR. Speech enhancement should always be considered for classroom applications of ASR technology, but it can be also be accommodated inherently by the feature extraction technique of the ASR system if this technique is noise robust (e.g. RASTA). As stated in the background section, LPC and MFCC are not noise robust feature extraction techniques and will from the application of speech enhancement.

Applying a Speech Enhancement technique will typically require a distinct waveform audio recording of the noise-polluted speech of the educator during the lesson, recorded by the primary microphone as well as a distinct parallel recording of the noise pollution itself occurring within the classroom, recorded by a secondary microphone. In the ideal recording setup, the noise audio can then be removed from the educator's speech signal almost entirely, resulting in the audible speech of the educator to be analysed by the chosen ASR system, improving its performance and ultimately the accuracy of generated lesson summaries. The chosen speech enhancement technique can be performed during the recording of the lesson, applied to the real-time speech audio stream, using the real-time noise pollution audio stream; or alternatively, the lesson can be recorded in its entirety along with environmental noise and then the speech enhancement technique can be applied to the resulting recording after the lesson once the speech audio and noise audio have been aligned for cancellation. Most importantly, if deemed a necessary intervention, a speech enhancement technique should always be applied prior to ASR.

3.3 *Lesson Transcript Generation*

The background section covered the capability of ASR systems to transcribe the speech of the educator. Historically, this has been the fundamental reason for implementing speech recognition technology in the educational setting. Lesson transcripts provide many benefits, notably the potential for content reflection and note taking as mentioned in [6, p. 369], as well as improved teaching methods and support of students with disabilities, as described in [4, pp. 65-66], there are clear motivations for lesson transcript generation. In the age of information and with the int eruptions in teaching caused by the COVID-19 pandemic, online learning and video conferencing are becoming more prevalent approaches to education. In these learning environments lesson transcripts are an additional resource for students. Cloud-based ASR systems provide an easily accessible ASR service and can allow researchers to access advanced ASR Models like Google Cloud Speech Recognition and CMU Sphinx. An ASR model can be incredibly challenging to develop from scratch even for an educational institution, due to the mathematical complexities involved in training a machine learning model and the tremendous amount of training data required to optimise it, especially for data as diverse as speech. Depending on the available resources and with recent trends making continuous progress on ASR performance, the task of lesson transcript generation itself might be better suited to a well established, cloud-based ASR model.

3.4 *Educator Tagged Keywords and Course Content*

The proposed model capitalises on the transcript-generation process by analysing the lesson audio to detect, sequence, count and associate known keyword terms with course content items, prompted by the ASR system's detection of a single utterance of each associated keyword. The count of the number of times each keyword term is uttered throughout the lesson audio can also be maintained to show topic prevalence, allowing students to gauge the importance of various keywords used throughout the lesson and prioritise the amount of time they should spend studying associated topics. The identified keyword terms along with their sequence and utterance counts should only be extracted/calculated based on the new segment of lesson transcript (appended to the overall transcript), after the overlap has been accounted for to avoid duplicate keyword terms being counted.

For the proposed model to be successfully applied, each section within the course content must be tagged with one or more keywords to facilitate the associations between each of the identified keywords in the lesson transcript, with their relevant sections in the course content. This can be achieved by creating meta-tags in a database-bound content management system, storing the relevant keywords for each section of the course. Since the course content is generated from these systems by querying the course content database, the keywords identified through lesson transcription can be queried against the meta-tags within the course content database. Many keywords can be tagged to account for different teaching styles and linguistic preferences. By adding these meta-tags wherever relevant, the identification of a keyword can trickle down through the entirety of the course content and highlight all the sections of the course where the keyword has been tagged. The course content director or the educator should delineate the relevant keywords in each of their associated sections within the course content. The educator must then ensure that they use some of the specific keywords when delivering the lesson so that the model can highlight these associations. The keywords that the educator plans to use are then provided as an input to the transcript generation process to specify which keywords to look for as the speech of the educator is transcribed. This entails sufficient lesson planning and familiarity with subject matter.

3.5 *Topic-oriented Lesson Summary Generation*

Through the generation of the lesson transcript, the full transcript resulting from segmented ASR performed on the lesson audio, as well as the recorded keyword details were collected. The keyword details include the sequence of its utterance and its utterance count. This information can be summarised and presented to students to help them reflect on what was covered during the lesson, as emphasised in [4, pp. 66-67]. As stated in [4, p. 67] the lesson transcript acts as the primary resource for clarification of what was directly communicated during in the lesson. The additional keyword details captured are supplementary, but allow for the generation of the lesson summary to be structured in accordance with keyword sequence. The identified keyword terms and their utterance counts serve to emphasise the specific topics discussed and their prevalence within the lesson transcript, further bolstering secondary contact with the subject matter.

To finalise the lesson summary and present it to students, one final process must be performed to associate the identified keywords uttered with their relevant sections in the course content. This is achieved by iterating through the list of identified keywords and for each keyword, testing whether it has been tied using meta-tags to a section of the course content, by the educator. If the keyword is assigned then the association is made via the database bound content management system and the section heading along with any additional information stored in the database (such as the page number) is added to the lesson summary. Once all of the keyword information, along with associated course content artefacts have been extracted, the information can be arranged and presented to students for review. The topic-oriented lesson summary could be structured as follows: first, the details for each keyword term can be stated or plotted on a pie chart (or bar graph) in accordance with their sequence and utterance count to show topic prevalence; second, the sections associated with each keyword can be listed, aligned with the sequence of their related keyword terms; and third, the lesson transcript itself can be added to the lesson summary. It is worth noting that the release of the lesson summary to students could be delayed to allow for the educator to edit the transcript (should it be added to the summary); removing redundancies and adding punctuation where needed to correct mistakes made by the ASR model during transcription.

4 Proof of Concept Prototype Overview

This section discusses the implementation of the prototype based on the proposed model. This prototype was developed in the Python programming language, over versions 3.6 and 3.7, using the Anaconda platform with the Spyder Python editor. Python is a flexible, object-oriented programming language, providing exceptional access to various programming libraries written in Python, C and C++ which are appropriate for handling and manipulating digital speech data. Python was the programming language of choice for the prototype, since it provided access to numerous programming libraries available within the topic area, made available free of charge by its longstanding data science community. The subsections that follow describe the functions of each of the major modules which facilitate the prototype's goal of generating a topical lesson summary. Figure 2 shows the class diagram of the proof of concept prototype developed alongside the model. For simplicity, method input parameters and return types have been omitted.

4.1 *Large Vocabulary Continuous Speech Recognition (LVCSR)*

The LVCSR Class is responsible for all ASR performed by the prototype. This is achieved through cloud speech recognition services made available by [29], an API which provides access to a number of cloud-based ASR services, some at a cost, and others free for application development, research and testing. The two ASR service providers utilised by the prototype were CMU Sphinx [30] and Google Cloud Speech Recognition [31]. Both of these providers have made their APIs available free of charge for use in the development of speech-driven systems. Regardless of provider, the access methods of this API take as input a single waveform audio file, which the prototype uploads to the given recogniser via the World Wide Web. The chosen recogniser then processes this data on the cloud server and returns the result of transcription of the uploaded audio. Google Cloud Speech Recognition [31] is based in a DNN strategy for ASR and reports to perform speech enhancement technique, while [30] is based in a HMMGMM strategy for ASR and does not perform noise cancellation. Other paid cloud speech recognition solutions made available by the API in [29] include IBM Speech to Text and Microsoft Bing Speech Recognition, among others. What makes this particular ASR service API so useful is that each recognition service is exposed in the same way, making the implementation cross-compatible, regardless of the recogniser chosen at run-time.

The implemented prototype was adapted to generate a lesson transcript by uploading multiple shorter segments of lesson audio to accommodate cloud-based ASR using Google Cloud Speech Recognition and CMU Sphinx. To achieve this, lesson transcripts were generated by sequencing the overlapping segments of lesson audio from start to finish; then performing ASR on overlapping segments of recorded lesson audio with the transcript accumulating as new audio segments were analysed. The overlap was applied to account for sudden cuts in the audio mid sentence owing to the segmentation and also to ensure continuity between the results of transcribed segments of lesson audio, resulting in a more accurate lesson transcript once the transcript was aligned and textual overlap had been accounted for between segments. Through this process of segmentation analysis, the sequence and a counter of each keyword term uttered by the educator were maintained.

4.2 *The Wave Handler Module*

The Wave Handler module facilitates all the necessary access, storage and segmentation of Waveform Audio Specification files associated with lesson audio analysis. This is achieved using two methods. The first returns the duration of a WAV file specified by file path. The duration of the specified file is calculated by dividing the number of frames in the file by the frame rate of the file, retrieved from its file header. The result of the division is returned as a floating point number representing the duration, measured in seconds, of the specified file. This method primarily serves to determine the boundaries of a WAV file when accessing or segmenting WAV data, but is also used by the control loop to iterate the analysis window over the entirety of the recorded lesson audio.

The second method is used to segment WAV audio and also takes as input a specified WAV file path as well as a start time and an end time. The start time and end time parameters are used to window the speech audio for segmentation and, ultimately, for analysis by the chosen ASR system. This method also accepts three optional parameters: a possible WAV file path to a noise sample WAV file (defaulted to empty), a noise reduction Boolean (defaulted to false) and a noise cancellation Boolean (defaulted to false) indicating whether the segmented audio should have pseudo-random noise injection, noise reduction and/or noise cancellation applied respectively, using the specified noise sample. This allows for any WAV file to be read and segmented according to appropriate time intervals, as desired. When a noise sample is specified for injection and possibly subsequent noise reduction and/or cancellation, a random interval from within the roughly ten hour noise audio file is chosen to be used for injection each time the speech audio is segmented for analysis.

4.3 *Database Design for Lesson Summary Generation*

In order to establish associations between known keyword terms and sections of course content, a structure was required to retain these associations. The structure of choice to meet this requirement was a MySQL database. This database has three tables: first, a keywords table, which stores a primary key identity and lower-case

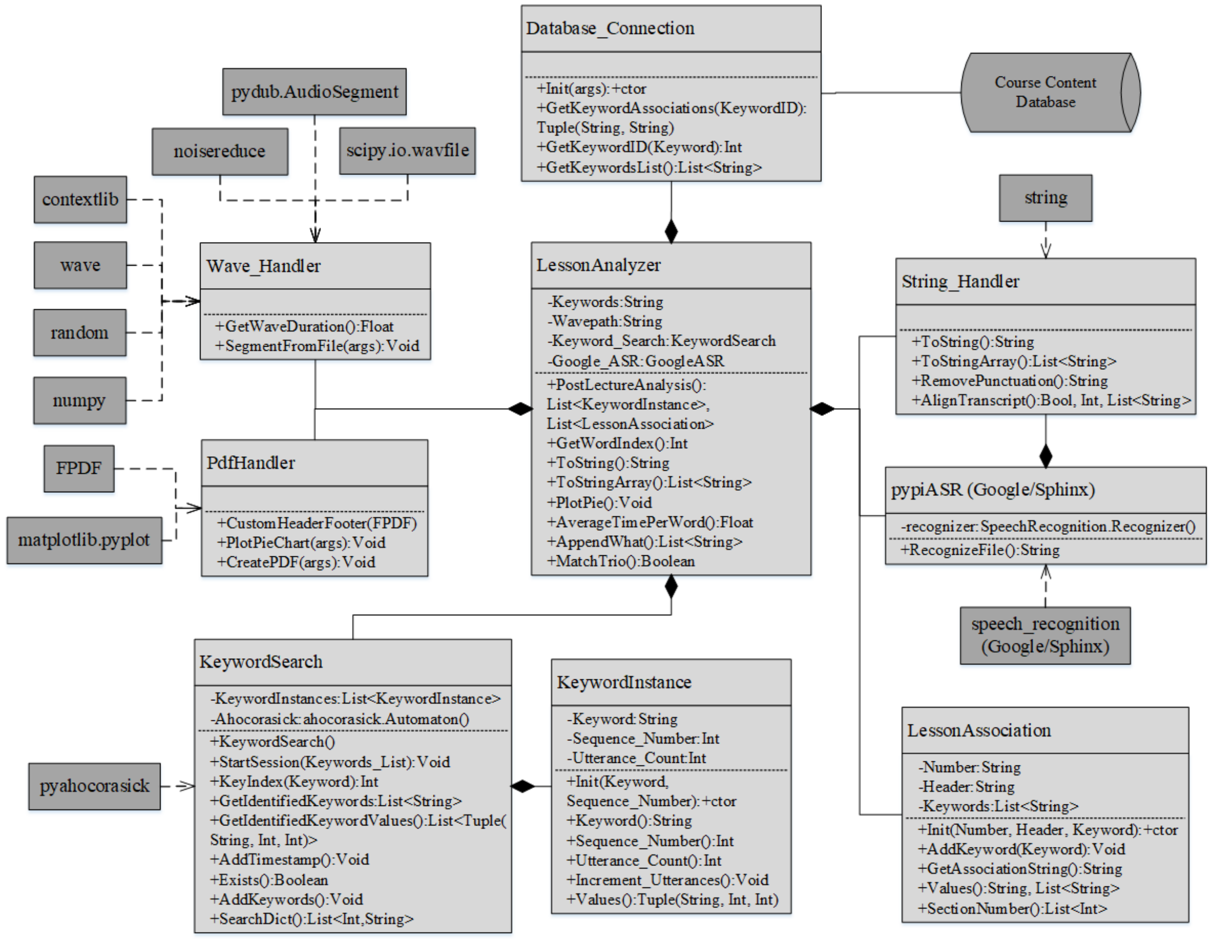
Database_Connection
+Init(args):+ctor
+GetKeywordAssociations(KeywordID): Tuple(String, String)
+GetKeywordID(Keyword):Int
+GetKeywordsList():List<String>
Course Content Database
pydub.AudioSegment
noisereduce
scipy.io.wavfile
contextlib
wave
random
numpy
Wave_Handler
+GetWaveDuration():Float
+SegmentFromFile(args):Void
LessonAnalyzer
-Keywords:String
-Wavepath:String
-Keyword_Search:KeywordSearch
-Google_ASR:GoogleASR
+PostLectureAnalysis(): List<KeywordInstance>, List<LessonAssociation>
+GetWordIndex():Int
+ToString():String
+ToStringArray():List<String>
+PlotPie():Void
+AverageTimePerWord():Float
+AppendWhat():List<String>
+MatchTrio():Boolean
string
String_Handler
+ToString():String
+ToStringArray():List<String>
+RemovePunctuation():String
+AlignTranscript():Bool, Int, List<String>
FPDF
matplotlib.pyplot
PdfHandler
+CustomHeaderFooter(FPDF)
+PlotPieChart(args):Void
+CreatePDF(args):Void
pypiASR (Google/Sphinx)
-recognizer:SpeechRecognition.Recognizer()
+RecognizeFile():String
speech_recognition (Google/Sphinx)
KeywordSearch
-KeywordInstances:List<KeywordInstance>
-Ahocorasick:ahocorasick.Automaton()
+KeywordSearch()
+StartSession(Keywords_List):Void
+KeyIndex(Keyword):Int
+GetIdentifiedKeywords:List<String>
+GetIdentifiedKeywordValues():List<Tuple(String, Int, Int)>
+AddTimestamp():Void
+Exists():Boolean
+AddKeywords():Void
+SearchDict():List<Int,String>
pyahocorasick
KeywordInstance
-Keyword:String
-Sequence_Number:Int
-Utterance_Count:Int
+Init(Keyword, Sequence_Number):+ctor
+Keyword():String
+Sequence_Number():Int
+Utterance_Count():Int
+Increment_Utterances():Void
+Values():Tuple(String, Int, Int)
LessonAssociation
-Number:String
-Header:String
-Keywords:List<String>
+Init(Number, Header, Keyword):+ctor
+AddKeyword(Keyword):Void
+GetAssociationString():String
+Values():String, List<String>
+SectionNumber():List<Int>

Figure 2: Class diagram of the proof of concept prototype system based on the proposed model

text for each of the known keyword terms the prototype is concerned with; second, a content table, which stores a primary key identity for each course content item, the header text of the course content item, a potential section number as well as a description; and third, a keyword-content relational table that ties each keyword term to one or more course content items. These ties are stored as distinct combinations of keyword term and content item primary key identity pairs which can be queried to yield the course content items associated with any known keyword term that the ASR system has identified.

In the keywords table, spelling is essential and distinctions should be made between standardised (American and British English) spelling. While the associations exist between the identity values of keywords and course content items, the association can only be triggered if the chosen speech recogniser identifies the text of the keyword term while generating the transcript. If the keyword term is misspelled compared to its recognised counterpart, it will not be identified and the association cannot be made. This is also why all string comparisons are performed in lower case since uppercase characters have different values to their lowercase counterparts. The prototype is concerned with seventeen keyword terms in total (four of which are not associated with any course content items), lists fourteen content items (four of which are not associated with a keyword term), and maintains fifteen keyword-content associations, with some associations existing between one keyword and many course content items.

4.4 The String Handler Module

The String Handler module deals with all text data transferred by prototype operation between processes. This module has five functions, and all string comparisons implemented by these functions are performed in lower case. The first two methods are used to split a sentence (on space characters) into an array of words, and the second recombines this array back into a sentence in the word order of the array indices. These functions are used to analyse and align word sequences during prototype operation. The third function is defined to remove punctuation and accepts any continuous string as input; then returns the equivalent string with commas, apostrophes, full stops, etc removed. This method is also used to facilitate string comparisons and to ensure that grammatical character differences do not prevent word sequence matches.

The last two methods of the String Handler module are used to align transcript segments. The fourth method performs a sequence match using two string arrays, created by the first method, as input. This method iterates through the words in the first array and determines whether the sequence of words in the first array matches the sequence of words in the second input array. If a match is found, this function returns a Boolean true; and if not, it returns Boolean false. The fifth method in the String Handler module is used to align transcript segments as the audio is windowed, analysed and transcribed by the speech recogniser. This method applies the previous sequence match method to identify the index where two given string arrays align, and then returns the aligned text from the identified index onward. As input, this method takes two string arrays; the current transcript (or an empty array if this does not yet exist) and the most recently transcribed text (converted to an array) as well as an integer to define how many words in sequence are required to match, and a second integer to define how deep through the array the method should search. The last two parameters are dependant on the duration of the overlap of the speech analysis window and the maximum number of words in the longest keyword term. This alignment method also persists with its search once a match is identified to ensure that the last possible matching index is used to return the overlapping text, and not simply the first identified match. This is to prevent transcribed utterances that are repeated from being matched prematurely, which would result in repeating segments of the transcript. This alignment is performed with each iteration of the prototype over the lesson audio, prior to any keyword term identification and possible association.

4.5 The Aho-Corasick Keyword Search

To identify keyword terms uttered by the educator, an Aho-Corasick keyword search was implemented. This search algorithm, made available by [32] was chosen because it is fast and reliable, and would not result in significant delays when searching for keyword terms between overlapping windows. Technically, any string-search algorithm would suffice here, but the Aho-Corasick search was deemed most appropriate to meet the requirements of the prototype. This search algorithm is implemented as an automaton whereby each keyword term that the prototype is concerned with is added to the watch list of the automaton at run-time. Any known keyword term can be identified in any potential search string. The Aho-Corasick automaton searches for keyword terms in parallel, meaning that the prototype does not need to iterate though each known keyword term and search for it; instead, the automaton has its bound keyword terms set at run-time and will search for any of these terms simultaneously. This behaviour facilitated the identification and accumulation of keyword term utterances stored by a customised Keyword Instance class.

4.6 The Lesson Analyser Program

The Lesson Analyser class combines the logic of each of the modules discussed in the previous subsections to form the execution algorithm of the prototype for lesson summary generation. The prototype is configured at run-time and requires the path to the recorded lesson audio; a directory into which to output the lesson summary; a duration for the analysis window (defaulted to ninety seconds); a duration for the overlap between analysis windows (defaulted to fifteen seconds); a string to specify the speech recogniser to use; an overlap comparison length used by the transcript alignment method (defaulted to twenty five words); and an overlap comparison match count (defaulted to four words). In addition, the algorithm accepts a possible noise audio path to use for noise injection, a noise reduction Boolean and a noise cancellation Boolean to specify which noise management technique(s) to apply.

The prototype's algorithm iterates over the lesson audio in ninety second windows of data with fifteen seconds of overlap in the data between them. If a path to a noise file has been specified, the prototype samples a random ninety second window of data from the noise audio and injects it over the current segment of lesson speech audio. The windowed audio is then written to a temporary directory, where it is uploaded via the World Wide Web to the specified speech recogniser. Once the upload completes, the lesson audio is transcribed and the text result is returned to the prototype. The returned transcript text is then analysed for overlap with the existing transcript text and the new text is appended to the transcript. The newly appended section of the transcript is then analysed for known keyword terms and, upon identification, a database lookup executes to identify the lesson content item(s) that the educator has associated with the keyword term. In addition, the utterance counts for each known keyword are maintained or updated. Identified keyword terms and their associations are then stored in memory and the loop iterates over the next 75 seconds (analysis window minus the overlap window) of lesson audio. This process continues until all the lesson audio has been processed, transcribed and analysed for keyword terms and associations. Once the process completes, the information acquired during analysis is handed off to be ordered and written to a PDF document to be presented to students. This PDF lesson summary document has three sections: first, a pie chart demonstrating the keyword terms uttered during the lesson, the sequence of their utterance and the number of times each was uttered; second, the list of associated course content items for students to review; and third, the lesson transcript. The lesson summary generated by the prototype is then provided to students, either via email or in print to support them in their secondary contact with the lesson material and to point them toward the course content that the lesson was based on.

5 Evaluation of the Prototype

To test the robustness of the prototype for ASR, keyword term identification and subsequent association to lesson content, a series of evaluation metrics were considered and a series of test case scenarios were established to provide quantitative evidence of the prototype's performance. The subsection that follows provides contextual definitions of the chosen recorded (base-line) performance metrics, and the chosen comparative (derived) performance metrics used for statistical evaluation of the prototype. After this, the next subsection defines each of the five established evaluation test scenarios. These scenarios are established to contrast the potential for acceptable lesson summary generation of Google versus CMU Sphinx cloud-based ASR systems against the combined effects of noise injection, noise reduction and noise cancellation.

5.1 Chosen Prototype Performance Metrics

Since the prototype system is driven by the speech of the educator, the lesson summary generation process is dependent on the ASR system for keyword term identification. Consequently, the accumulation of keyword-term utterances and the association of keyword-terms to lesson content is also dependant on the implemented ASR system. If the prototype fails to identify a keyword-term which is tied (via database relationship) to a lesson association, the association will not be made. In this sense, the proposed model is dependent on the speech recogniser. However, at a functional level, this dependence is binary in terms of the prototype's ability to associate any correctly uttered keyword term with its database-bound relationship to course content, defined by the educator. Established binary predictive machine learning models use base-line metrics known as True Positives, True Negatives, False Positives and False Negatives to measure binary predictive performance.

The four base-line performance metrics are defined in [33] and [34] where the outcomes of the predictive model result in a confusion matrix that describes its positive and negative behaviours. While the prototype system and reflective proposed model presented herein isolate the associations between keyword terms and course content as a database relationship, the ASR component of the prototype system either identifies a known keyword term - resulting in association, or it does not. An example is provided [35] where a binary classifier for diabetes detection is laid out. Contrasted with this example, one can think of each keyword identification as an independent classification resulting in the detection of known lesson content association. The descriptions of base-line performance metrics in the context of the prototype system presented in this work are described in Table 1.

Table 1: List of recorded performance metrics considered to evaluate the prototype

Recorded Metric	Known Keyword Term Recognition	Educator Defined Association
True Positives	Prediction is +ve	Keyword Term has +ve Association(s)
False Negatives	Prediction is -ve	Keyword Term has +ve Association(s)
False Positives	Prediction is +ve	Keyword Term has -ve Association(s)
True Negatives	Prediction is +ve	Keyword Term has -ve Association(s)

Table 1 defines the measurements recorded by the prototype during lesson analysis. A true positive occurs when the prototype identifies a known keyword term uttered by the educator and associates this keyword term to the correct course content item(s). Conversely, false negative occurs when the prototype identifies a known keyword term, but cannot associate the keyword because there is no required database constraint. A false positive occurs when the prototype identifies an unknown, spoken keyword term and associates it with unintended course content item(s). On a functional level, the prototype should not measure any false positives given that the relational database constrains the associations between keyword terms and lesson content. A true negative occurs when the prototype identifies a known, spoken keyword term, but there is no association between the keyword term and course content item(s). Due to the database constraints, true negatives can be eliminated by ensuring that only keyword terms with existing associations can be identified by the prototype.

The number of utterances of the keyword term gives the ASR component multiple attempts at keyword-term identification; however, if the particular term (e.g. the word "Euclidian") is not in the ASR system's lexicon, then the likelihood of detection falls to zero and the association cannot be made until further training of the ASR component. This constraint or dependency of the prototype on its ASR component is a trade-off between the number of times a keyword term is uttered and the reliability of the speech recogniser in identifying the keyword term.

Using the baseline performance metrics described allows for the measurement of derived performance metrics that can be calculated to provide a better indication of the prototype system's performance. Table 2 defines the formulas according to [33] and [34], which are used to calculate each of these comparative performance metrics using the values captured by the baseline performance metrics.

Table 2: List of comparative performance metrics considered to evaluate the prototype

Performance Metric	Measurement Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$ (1)
Precision	$\frac{TP}{TP+FP}$ (2)
Recall (Sensitivity)	$\frac{TP}{TP+FN}$ (3)
Specificity	$\frac{TN}{TN+FP}$ (4)
F1-Score	$2\times\frac{Recall\times Precision}{Recall+Precision}$ (5)

As previously addressed, the prototype's design constraints and dependencies, in theory, prevent it from capturing false positives and can prevent it from capturing true negatives with optimal configuration of the prototype's database component. Although these are both measured for, when their values equal zero, they become trivial to some of the comparative metrics that use them in associated formulae. In addition to this factor, the third-party nature of using a cloud-based ASR component abstracts the prototype (as the user) from the base-line metrics measured by the ASR model itself when decoding (transcribing) the educator's speech. Thus, we cannot measure the ASR model's performance directly using these comparative metrics.

Table 3: List of considered comparative performance metrics to use in evaluating the prototype system

Performance Metric	Metric Formula
Average Accuracy (AA)	$$\frac{\sum_{n=1}^{n} Accuracy_n}{N} \quad (6)$$ where n is the test case number and N is the total number of tests conducted for the test case
Average True Accuracy (ATA)	$$\frac{\sum_{n=1}^{n} TAccuracy_n}{N} \quad (7)$$ where FP = 0, TN = 0, n is the test case number and N is the total number of tests conducted for the test case
Global Average Accuracy	$$\frac{\sum_{n=1}^{n} AA_n}{N} \quad (8)$$ where n is the test case number and N is the total number of tests conducted for the test case
Global Average True Accuracy	$$\frac{\sum_{n=1}^{n} ATA_n}{N} \quad (9)$$ where FP = 0, TN = 0, n is the test case number and N is the total number of tests conducted for the test case

The database constraints on the prototype system coupled with the lack of base-line recognition metrics of the underlying ASR model, mean that some of the comparative metrics from Table 2 do not truly reflect the intention of the metric in the context of the prototype system. Because the count of false positives measured by the prototype will always be zero, precision and specificity were deemed inappropriate metrics for prototype evaluation from the outset. This consequently eliminated F1-Score as a potential measure for contrasting the performance of cloud-based Google and Sphinx recognition services. If we cater for FP = 0 and TN = 0, our measure for sensitivity results in the same value for Accuracy. By this deduction, accuracy was deemed the most appropriate metric to measure prototype performance. Table 3 provides formulae for the types of accuracy, measured for the evaluation of the prototype.

The accuracy metrics in Table 3 will provide a measure of accuracy for the prototype over the series of tests conducted across test cases. To cater for the potential of true negative values being measured as zero, the Average Accuracy (AA), which includes the number of true negatives measured, is contrasted with the Average True Accuracy (ATA) where the number of true negatives measured is assumed to be zero. These accuracy metrics are reported for each test case and the Global Average Accuracy and Global Average True Accuracy is reported across all the test cases to provide a final measure of prototype performance using the given speech recogniser.

5.2 *Specification of Test Samples*

To test the ASR prototype system, two audio readings of the article 'Speech Recognition for Learning' [36] were recorded using a stereo headset. The participants, one male and one female, were encouraged to enunciate their speech as well as possible and to read at a comfortable pace in their natural voices. In addition to these two audio recordings, a total of twelve hours of captured classroom background noise was acquired to use as sample audio for noise injection. These three audio test samples were all recorded at 16000Hz with a 256kbps bitrate as signed 16-bit PCM encoded single channel (mono) waveform audio (.wav).

5.3 *Prototype Test Cases*

A series of test cases were established to test the prototype system's performance using the test samples and performance metrics discussed. These test cases simulate alternative classroom equipment setups and environmental noise constraints on the prototype. Table 4 indicates whether the input data had noise injection (NI), noise reduction (NR) and/or noise cancellation (NC) applied for each of the test case scenarios, as well as the recogniser used to identify keyword terms.

Test Case T is established to provide an indicator of the functional performance of the prototype system assuming that the underlying ASR system is completely accurate. Rather than performing speech recognition on the speech audio, this test case instead has the algorithm operate on the raw text of the document which was read aloud. This allows for the algorithm to be tested and modified until it was proven to be working ideally on a functional level, with its dependency on ASR accuracy removed. Test case A, on the other hand, is used to measure the prototype's performance working with the ASR system to provide a measure of the prototype system's accuracy with its ASR dependency, but without any environmental noise. Test case B introduces this environmental noise by isolating a random sample within the specified classroom noise sample audio and overlaying this randomly chosen sample on the speech audio prior to ASR analysis. This pseudo-random noise injection is applied at each overlapping window of lesson audio analysis. Test case B is designed to simulate spontaneous noise that may occur in the educational environment and be recorded by the educator's microphone as they deliver their lesson. Test cases A and B simulate a single microphone setup to capture the educator's voice.

Table 4: Test case scenario specifications used in evaluating the prototypes' performance with different audio pre-processing techniques

Test Case	NI	NR	NC	ASR System
T	FALSE	FALSE	FALSE	Aho-Corasick Search
A	FALSE	FALSE	FALSE	Google, Sphinx
B	TRUE	FALSE	FALSE	
C	TRUE	TRUE	FALSE	
D	TRUE	FALSE	TRUE	
E	TRUE	TRUE	TRUE	

Table 5: Sphinx Average Accuracy and Average True Accuracy per Test Case

Test Case	Average Accuracy	Average True Accuracy
A	0,6333	0,7000
B	0,3907	0,4730
C	0,1660	0,2350
D	0,5300	0,6170
E	0,4600	0,5490
Global Average	0,4360	0,5148

Test cases C, D and E are concerned with noise management interventions and are intended to simulate a dual-microphone equipment setup where one microphone records the educator's voice and the other records environmental classroom noise. These recordings are then aligned, and the recorded environmental noise is used as input to reduce or cancel environmental noise recorded by the educator's microphone. Note that the prototype would be drastically less effective if the microphone recording the environmental noise were also to record the educator's voice, as the noisy sample would include the speech, which would then be cancelled or reduced. Thus, this prototype is not appropriate for a situation where the educator's voice is being amplified by a loud speaker. The educator's microphone in this case is intended only to record their speech and inherently, any environmental noise generated in the classroom which is also captured.

5.4 Prototype Test Results

The prototype was tested one hundred times for each of the test cases defined in Table 4 using Google Cloud Speech Recognition and then using CMUSphinx ASR. Test case T was performed without a recogniser and instead used the Aho-Corasick search to identify keyword terms. The prototype was debugged and tested in a cyclical manner until the results of test case T showed an accuracy of 100% and it had been demonstrated that the logic used to measure each base-line performance metric was accurate. Ultimately, the prototype was able to demonstrate 100% accuracy on test case T and any doubt of inaccuracy of base-line metric measurement was resolved. This prompted the next phase of testing of the prototype using a speech recogniser to transcribe the educator's speech and then to generate a lesson summary. Table 5 shows the performance of the prototype when generating lesson summaries using CMUSphinx cloud-based ASR.

Table 6 shows the performance of the prototype when generating lesson summaries using Google Cloud Speech Speech Recognition.

Table 6: Google Average Accuracy and Average True Accuracy per Test Case

Test Case	Average Accuracy	Average True Accuracy
A	1,0000	1,0000
B	0,8927	0,9390
C	0,4387	0,5360
D	0,9693	0,9900
E	0,8860	0,9400
Global Average	0,8373	0,8810

6 Discussion of Prototype Performance

The discussion of test results will centre around the reported Average True Accuracy across test cases for each recogniser. The results of the prototype testing reported in Table 5 and Table 6 for test case A demonstrate a difference in accuracy of 30% for lesson summary generation between Google Cloud Speech Recognition (CSR) and CMUSphinx ASR. This can be attributed to the design difference of these machine learning models. It has been shown that DNN-based approaches to ASR outperform traditional HMM-based models as they provide more modelling complexity. When noise is injected into the educator's speech in test case B, the performance of both models declines, but Google CSR is reportedly noise robust so the drop in accuracy is far less significant than that of CMUSphinx, where Google had a 6.01% reduction in accuracy, contrasted with a drop in accuracy of 22.7% for CMUSphinx. Both recognisers saw a significant reduction in accuracy with noise injection and then with noise reduction applied in test case C. From a qualitative perspective, the noise reduction algorithm had the effect of dampening the speech and was likely de-emphasising spectral formants, leaving the speech sounding hollow and making it difficult to distinguish utterances by ear. By contrast, when noise cancellation was applied instead in test case D, both speech recognisers saw an improvement in performance; with Google improving in accuracy by 5.1% compared with a greater increase in accuracy for CMUSphinx of 14.4%. This demonstrates that noise cancellation was the ideal noise management technique applied by the prototype, most notably with Google CSR which lost only 1% accuracy of lesson summary generation when comparing performance without noise injection as opposed to noise injection with noise cancellation applied. The results of test case E are the most surprising where noise reduction and noise injection was applied, a balance in accuracy between noise reduction and noise cancellation was achieved, seemingly diminishing the negative effects of the noise reduction technique, while still removing background noise and retaining the integrity of the speech signal. Nevertheless, the prototype showed satisfactory performance using either recogniser under no noise injection, or while only applying noise cancellation to the noisy speech data. Overall, using Google Cloud Speech Recognition in combination with noise cancellation demonstrated the best performance of the prototype for speech-driven lesson summary generation in a noisy environment.

7 Conclusion and Future Work

This paper has discussed some of the key topic areas involved in incorporating ASR systems for use in educational settings. A model was proposed for the application of an ASR system in a noisy educational environment to automatically generate a lesson summary, driven by the speech of the educator. A prototype system was then developed based on the proposed model and improved and adapted alongside it. The prototype goes beyond the baseline utility of transcribing the speech of the educator with additional analysis on the transcribed text used to identify and associate keyword terms to course content artefacts, summarising this information by monitoring the number of times each keyword is mentioned, providing a reference point for keyword terms and directing students to the underlying course content from which they originate via database bound associations. These relatively simple additions, along with the lesson transcript can allow for the educator to speak at length about the given topics, cross-referencing to related course content artefacts and potentially helping to guide the flow of their lesson, with the peace of mind that relevant sections are made known to the students. Additionally, secondary contact with the lesson transcript after it has been taught helps students with making notes and the additional reference points could help in prioritising certain topics and reaffirming/reinforcing what was communicated during the lesson.

To further the proposed model, future work involves the development, testing and classroom implementation of a more advanced system based on the proposed model and the proof of concept prototype presented and argued herein. The classroom prototype would need to be evaluated using both qualitative and quantitative research methods. For qualitative evaluation, a survey could be designed to gauge both student and educator perspectives on the system's utility, potential for improvement and the overall sentiment of the system implemented in the classroom. Quantitative evaluation could be incorporated into the survey whereby aspects of the prototype's utility could be rated on a Likert scale or Linear Numeric scale to indicate positive or negative sentiment towards particular aspects of the system and the lesson summaries produced. The prototype's database components could also be expanded to cater for multiple lessons across multiple courses and the performance evaluation conducted in this work could then be applied at various levels to measure prototype performance on a lesson and course level and also performance contrasted between courses with varying subject matter. The model could also be improved by automating the definition of keyword terms and their association with course content items so that the educator would no longer need to. This could be done using unsupervised machine learning whereby a machine learning model would have access to course content and automatically extract and assign keyword terms to course content artefacts.

References

[1] P. Blunt, B. Haskins, "A Model for Incorporating an Automatics Speech Recognition System in a Noisy Educational Environment," in 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 1–7, IEEE, 2019, doi:10.1109/IMITEC45504.2019.9015907.

[2] G. López, L. Quesada, L. A. Guerrero, "Alexa vs. Siri vs. Cortana vs. Google Assistant: a comparison of speech-based natural user interfaces," in International Conference on Applied Human Factors and Ergonomics, 241–250, Springer, 2017.

[3] J. Meng, J. Zhang, H. Zhao, "An Overview of Speech Recognition Technology," 199–202, IEEE, 2012, doi:10.1109/ICCIS.2012.202.

[4] R. Shadiev, W.-Y. Hwang, N.-S. Chen, Y.-M. Huang, "Review of Speech-to-Text Recognition Technology for Enhancing Learning," Journal of Educational Technology & Society, **17**(4), 65–84, 2014.

[5] A. R. Ahmad, S. M. Halawani, S. Boucetta, "Using Speech Recognition in Learning Primary School Mathematics via Explain, Instruct and Facilitate Techniques," Journal of Software Engineering and Applications, **07**(4), 233–255, 2014, doi:10.4236/jsea.2014.74025.

[6] W.-Y. Hwang, R. Shadiev, C.-T. Kuo, N.-S. Chen, "Effects of Speech-to-Text Recognition Application on Learning Performance in Synchronous Cyber Classrooms," Journal of Educational Technology & Society, **15**(1), 367–380, 2012.

[7] J. Crawford, K. Butler-Henderson, J. Rudolph, B. Malkawi, M. Glowatz, R. Burton, P. Magni, S. Lam, "COVID-19: 20 countries' higher education intra-period digital pedagogy responses," Journal of Applied Learning & Teaching, **3**(1), 1–20, 2020.

[8] T. L. Cox, R. S. Davies, "Using Automatic Speech Recognition Technology with Elicited Oral Response Testing," CALICO Journal, **29**(4), 601–618, 2012, doi:10.11139/cj.29.4.601-618.

[9] N. Das, S. Chakraborty, J. Chaki, N. Padya, N. Dey, "Fundamentals, present and future perspectives of speech enhancement," International Journal of Speech Technology, 1–19, 2020, doi:10.1007/s10772-020-09674-2.

[10] A. Tjandra, S. Sakti, G. Neubig, T. Toda, M. Adriani, S. Nakamura, "Combination of two-dimensional cochleogram and spectrogram features for deep learning-based ASR," in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4525–4529, IEEE, 2015, doi: 10.1109/ICASSP.2015.7178827.

[11] S. Lakshmikanth, K. R. Natraj, K. R. Rekha, "Noise Cancellation in Speech Signal Processing-A Review," International Journal of Advanced Research in Computer and Communication Engineering, **3**(1), 5175–5186, 2014.

[12] K. Gupta, D. Gupta, "An analysis on LPC, RASTA and MFCC techniques in Automatic Speech recognition system," in 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), 493–497, IEEE, 2016, doi:10.1109/CONFLUENCE.2016.7508170.

[13] U. Shrawankar, V. Thakare, "Feature Extraction for a Speech Recognition System in Noisy Environment: A Study," in 2010 Second International Conference on Computer Engineering and Applications, 358–361, IEEE, 2010, doi:10.1109/ICCEA.2010.76.

[14] A. S. Mukhedkar, J. S. R. Alex, "Robust feature extraction methods for speech recognition in noisy environments," in 2014 First International Conference on Networks Soft Computing (ICNSC2014), 295–299, 2014, doi: 10.1109/CNSC.2014.6906692.

[15] U. Shrawankar, V. Thakare, "Techniques for Feature Extraction in Speech Recognition System: A Comparative Study," International Journal Of Computer Applications In Engineering, Technology and Sciences (IJCAETS), **2**(2), 412–418, 2013.

[16] U. Sharma, S. Maheshkar, A. N. Mishra, "Study of robust feature extraction techniques for speech recognition system," in 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 654–658, 2015, doi:10.1109/ABLAZE.2015.7154944.

[17] M. Sathya, D. S. P. Victor, "Noise Reduction Techniques and Algorithms For Speech Signal Processing," International Journal of Scientific & Engineering Research, **06**(12), 317–322, 2015.

[18] U. Shrawankar, V. Thakare, "Noise Estimation and Noise Removal Techniques for Speech Recognition in Adverse Environment," in Z. S. S. V. A. A. D. Leake, editor, 6th IFIP TC 12 International Conference on Intelligent Information Processing (IIP), volume AICT-340 of Intelligent Information Processing V, 336–342, Springer, 2010, doi:10.1007/978-3-642-16327-2\ 40.

[19] J. Yang, W. Zhenli, "Noise robust speech recognition by combining speech enhancement in the wavelet domain and Lin-log RASTA," 415–418, IEEE, 2009, doi:10.1109/CCCM.2009.5267457.

[20] O. Abdel-Hamid, A. Mohamed, H. Jiang, G. Penn, "Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition," in 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4277–4280, IEEE, 2012, doi:10.1109/ICASSP.2012.6288864.

[21] T. N. Sainath, C. Parada, "Convolutional Neural Networks for Small-Footprint Keyword Spotting," in Proc. Interspeech 2015, 1478–1482, Google, Inc. New York, U.S.A., 2015.

[22] J. Huang, J. Li, Y. Gong, "An analysis of convolutional neural networks for speech recognition," in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4989–4993, IEEE, 2015, doi:10.1109/ICASSP.2015.7178920.

[23] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu, "Convolutional Neural Networks for Speech Recognition," IEEE/ACM Transactions on Audio, Speech, and Language Processing, **22**(10), 1533–1545, 2014, doi:10.1109/TASLP.2014.2339736.

[24] O. Vinyals, S. V. Ravuri, D. Povey, "Revisiting Recurrent Neural Networks for robust ASR," in 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4085–4088, IEEE, 2012, doi:10.1109/ICASSP.2012.6288816.

[25] C. Weng, D. Yu, S. Watanabe, B. F. Juang, "Recurrent deep neural networks for robust speech recognition," in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5532–5536, IEEE, 2014, doi:10.1109/ICASSP.2014.6854661.

[26] M. Ravanelli, P. Brakel, M. Omologo, Y. Bengio, "Light Gated Recurrent Units for Speech Recognition," IEEE Transactions on Emerging Topics in Computational Intelligence, **2**(2), 92–102, 2018, doi:10.1109/TETCI.2017.2762739.

[27] S. . Arık, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, A. Coates, "Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting," in Proc. Interspeech 2017, 1606–1610, ISCA, 2017, doi:10.21437/Interspeech.2017-1737.

[28] H. Dridi, K. Ouni, "Applying long short-term memory concept to hybrid "CD-NN-HMM" model for keywords spotting in continuous speech," in 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), 413–418, IEEE, 2018, doi:10.1109/ATSIP.2018.8364510.

[29] A. Zhang, "SpeechRecognition 3.8.1," Retrieved December 5, 2017, from https://pypi.org/project/SpeechRecognition/, 2017.

[30] C. M. University, "CMUSphinx Open Source Speech Recognition," Retrieved October 23, 2019, from https://cmusphinx.github.io/wiki/, 2019.

[31] Google, "Google Cloud Speech-to-Text API," Retrieved January 17, 2020, from https://cloud.google.com/speech-to-text/docs/apis, 2020.

[32] W. Muła, "pyahocorasick 1.4.0," Retrieved January 14, 2019, from http://www.ldonline.org/article/38655/, 2019.

[33] A. Mitrani, "Evaluating Categorical Models," Retrieved November 28, 2019, from https://towardsdatascience.com/evaluating-categorical-models-e667e17987fd, 2019.

[34] A. Mitrani, "Evaluating Categorical Models II: Sensitivity and Specificity," Retrieved December 6, 2019, from https://towardsdatascience.com/evaluating-categorical-models-ii-sensitivity-and-specificity-e181e573cff8, 2019.

[35] S. Ghoneim, "Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on?" Retrieved April 2, 2019, from https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124, 2019.

[36] N. C. for Technology Innovation, "Speech Recognition for Learning," Retrieved June 12, 2017, from http://www.ldonline.org/article/38655/, 2010.