Automatic Question Detection in Speech using Deep Neural Network

TASLIMA AKTER, Indiana University Bloomington, USA
HASIKA MAHTTA, Indiana University Bloomington, USA
KHANDOKAR MD. NAYEM, Indiana University Bloomington, USA

In this paper, we describe our efforts towards the automatic detection of questions in speech. We analyze the utility of various features for this task, Spectrogram and Mel-frequency cepstral coefficients (MFCC). We have used IEEE corpus and self-prepared corpus of human-voice recorded audio files and trained the data on Recurrent Neural Networks (RNNs) and Convolutional Recurrent Neural Networks (CRNNs) and compared their performances. Our system, provides state-of-the-art results on the clean corpus and in noisy environments as well.

ACM Reference Format:
Taslima Akter, Hasika Mahtta, and Khandokar Md. Nayem. 2018. Automatic Question Detection in Speech using Deep Neural Network. 1, 1 (May 2018), 12 pages. https://doi.org/

1 INTRODUCTION

Dialogue Act Recognition[3] is a challenging problem in dialogue interpretation which aims to attach semantic labels to utterance and characterize the speaker's intention. The most frequent DA types are [10] are statements and opinions, questions, back-channels. Our project focuses on detecting questions in the speech. Automatic Speech recognition is the versatile field of Computational Linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. The system analyses the person's specific voice and and use it to detect the person's speech with better accuracy. Questions in human dialogues is an important first step to automatically processing and understanding the natural speech. It can be viewed as a subtask of speech act or dialogue act tagging, which aims to label functions of utterances in conversations. The various types of questions are Yes-No, wh, Declarative, Rhetoric, echo, etc. It is useful for meeting indexing and summarization. Examples-
Yes-No : Have you looked at that?
Wh: What was the nature of the email?
Declarative: You are editing your slide?
Echo: He has undergone a surgery?
Rhetorical:Do you know that person?

Authors' addresses: Taslima Akter, Indiana University Bloomington, Bloomington, IN, USA; Hasika Mahtta, Indiana University Bloomington, Bloomington, IN, USA; Khandokar Md. Nayem, Indiana University Bloomington, Bloomington, IN, USA.

XXXX-XXXX/2018/5-ART $15.00
https://doi.org/

The main motivation behind this project is to develop a model that will consider intonation, stress and other speech attributes to classify the group questions. The current speech recognizer examples such as Google Home mini and Alexa can understand speech and respond accordingly. But these techniques are not smart enough to detect echo questions, rhetorical questions, etc. For an instance, questions like - "What is your name?" is easily recognizable by the current systems. But if the question is of echo type - "Your name is abc?", then in this case the current systems will classify it as a normal sentence. But our model will classify these types also as "Questions" only unlike the current systems.

1.1 PROBLEM

The main essence of the problem is that if we are given an audio recording, can we train a system to classify different types of Question (considering intonation and stress).

1.2 GOALS

The goal of this project is to build an audio classifier capable of recognizing questions and declarative sentences for male and female corpus. We aim to use a complex dataset which is composed of different types of questions such as yes-no, declarative, wh, tag, rhetorical, and echo. The reason behind increasing the complexity in the data set is the neural networks. As we aim to train our model using Recurrent Neural Network and Convolutional Neural Network, data should be complex. Neural Networks have evolved to address more and more difficulties of complex real world problems, varying from time evolving data to sophisticated data structures as graphs and functions. Deep Neural Networks tend to be applied on large scale problems that offer sufficient data to learn features automatically. They are able to give better accuracy if the nets are very deep and data is diverse.

2 RELATED WORK

One of the previous works [7] focused on the use of textual Internet conversations for detecting questions in spoken conversations. This work used the Meeting Recorder Dialog Act corpus(MRDA) data set and trained the model using logistic regression with LIBLINEAR package (Fan et al., 2008). This work explored the use of conversational web text to detect questions in conversational speech. These works did not perform well on declarative questions which can be improved by using prosodic features. Also relevant is the work of [2] who emphasized on the automatic detection of English questions in meetings.The corpus that was used was ICSI Meeting Recorder Dialog Act(MRDA). The features that were taken into consideration were Lexico-syntactic, Turn-related, FO statistics and Final FO slope. They have basically tried to analyze the utility of the above mentioned features for this task. And concluded that lexico-syntactic features enable the classifier to correctly identify the cues that signal a question. Word-n grams outperforms POS tags. The mismatch between the evaluation and parser training data has further to be reduced in future works and the quality of the parser output. The contributions of lexical and prosodic features have been explored in other works, e.g. [11] has presented the work on automatic question detection from speech signal and how developing automatic detection system could help in analyzing the probability of such a system to a new language. They have used Prosodic features like fundamental frequency (FO) and lexical features like unigram, bigram, etc specific to French or Vietnamese language to automatically detect questions in an audio input. The other aspect of this work was to conduct a cross-lingual(French/Vietnamese) evaluation concerning the use of prosodic features. They used decision trees for the classification and trained the model on the VietP corpus and Assimil corpus. There is this other work [1] that investigated on automatic detection of teacher questions automatically segmented human-transcripts of teacher audio recordings collected in

live classrooms. They used a dataset of audio recordings from 11 teachers across 37 class sessions. They were able to detect questions with a weighted F1 score of 0.66, suggesting the feasibility of question detection on automatically segmented audio from noisy classrooms. This other work [12] done on study on Chinese questions by building question detectors for Chinese and English conversational speech, and performing analytic studies and feature selection experiments. The most useful prosodic feature they recognized for their task was spectral balance, i.e., the distribution of energy over the frequency spectrum, of the final syllable. The corpus used was CALLHOME Mandarin Chinese corpus of telephone speech and its transcripts. The features they analyzed were textual, prosodic features and tones in question detection and all the experiments were run using the decision tree classifier. They concluded that for question detection, spectral balance is a more reliable feature than the overall intensity.

3 PROJECT DOCUMENTATION

We are running this experiment in order to fulfill the need of being able to identify all types of questions including echo and rhetorical questions also unlike the current systems which are not smart enough to actually detect the stress on each of the words in speech.
Our hypothesis is that given the sound recording as input, we will do feature extraction which convert the .wav format of the audio files into numerical features such as Spectrogram, MFCC , RMSE, and pitch that form the basis of testing our hypothesis and finally evaluating the model using Recurrent Neural Network (RNNs) and Convolutional Recurrent Neural Networks (CRNNs) to interpret the results which is the question class.

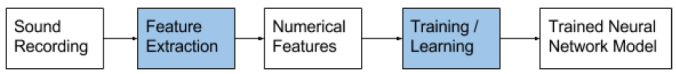


Fig. 1. **Automatic Question Detection classifier**

3.1 Models

3.1.1 ***Recurrent Neural Networks (RNNs):*** Recurrent Neural Networks (RNNs) is a class of artificial neural network where connections between the units from a directed graph are fed into a directed graph along a sequence [4]. It is thus a class of neural networks that exploit the sequential nature of the input. Such inputs could be text, speech, time series and anything where the occurrence of an element in the sequence is dependent on the elements that appeared before it. For example, the next word in the sentence the dog... is more likely to be barks than car, therefore, given such a sequence, an RNN is more likely to predict barks than car.
In RNN, if we assume a network containing RNN neurons, then each neuron will perform the same operation on every element of the sequence. RNNs are very flexible and have been used to solve problems such as speech recognition, language modeling, machine translation, sentiment analysis, image captioning, etc. They are adapted to solve problems by rearranging the way the cells are arranged in the graph. There are two variants of the RNN, **Long Short Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)**. Both LSTM and GRU are drop-in replacements for the SimpleRNN cell, so just replacing RNN cell with one of these variants can help obtain performance improvement in the network.

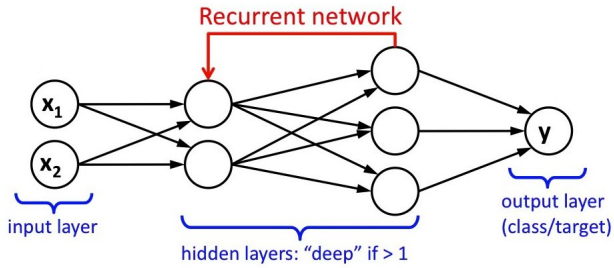


Fig. 2. **Recurrent Neural Network classifier**

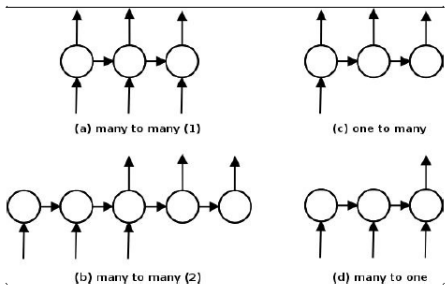


Fig. 3. **Recurrent Neural Network Topologies**

GRU is a variant of the LSTM and was introduced by K.Cho Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. It retains the LSTM's resistance to the vanishing gradient problem, but its internal structure is simpler, and therefore is faster to train, since fewer computations are needed to make updates to its hidden state. The gates for a GRU cell are illustrated in the following diagram:

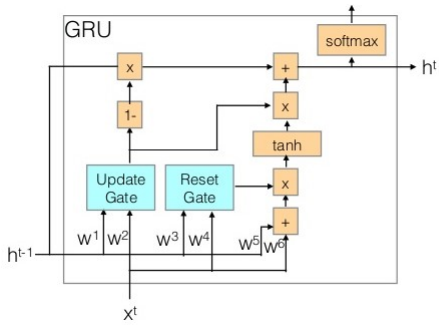


Fig. 4. **Gated Recurrent Unit (GRU)**

3.1.2 ***Convolutional Recurrent Neural Network:*** Traditional convolutional layers extract features from patches of data by applying a non-linearity on an affine function of the input. The CRNN [6] model feeds every window frame by frame into a recurrent layer and use the outputs and

the hidden states of the recurrent units in each frame for extracting features from the sequential windows. This will provide better features in comparison to the standard convolutional layers.

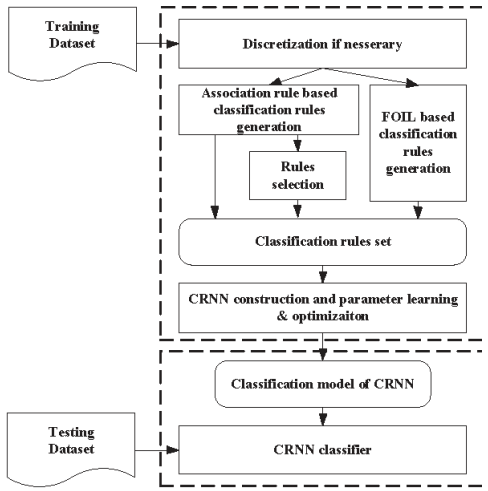


Fig. 5. **Convolutional Recurrent Neural Networks (CRNNs)**

3.2 Input and output

The Input of these algorithms is the audio files in the form of wav format and output is question class. The prerequisites are the recorded sound files.

3.3 Software/System Requirements

The software requirements includes **Anaconda** which is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Anaconda Distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS. The Other software on which we run the programs in Jupyter Notebook for python. **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. While Jupyter runs code in many programming languages, Python is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook itself. Runtime behavior: Recurrent Neural Networks consume a lot of time to run even on GPU which was causing Resource exhausted error.

4 DATA PROCESSING

In this section we describe about the data collection procedure and how we have processed them for our model.

4.1 Data Preparation

A speech corpus is generally comprised of speech audio files and text transcriptions. Large collections of speech corpora is available nowadays but speech corpus specifically related to our problem was hard to find. We needed labeled declarative statements and questions for our project. Some questions should have various levels of stress and intonation which is not freely available. So, we collected around 300 audio files containing labeled declarative sentences from IEEE corpus which covered 150 male and 150 female voices. Our professor also provided 12 sentences which have stress on one word for each sentence.

During the data collection phase our main challenge was collecting question data as they are not available as audio files. So we manually recorded the questions for our project. We initially collected a text corpus for questions and extracted 300 questions from it which are a good combination of 'wh', 'yes-no', 'open-ended', 'echo', etc. After extracting the questions we used Google Speech to get the audio from each questions and recorded that with the audio recorder software named 'EasyAudioRecorder'. We used both male and female voices and multiple speakers to record the questions.

To make this model robust and workable for real-life environment, we synthetically add 10 different types of noises (Babble, Cafe, Car, Factory, Machine gun, Plane, Restaurant, SSN, Tank, White Noise) in 5 different mixture signal-noise ratio (SNR) levels, -3dB, 0dB, 3dB, 6dB, 9dB. So we make the data size, $(300 + 300 \times 10 \times 5) = 30,000$. We take 75 percentile as train dataset, 5 percentile as validation dataset, and rest 20 percentile as test dataset. We take a random portion of noise and mix it with the clean audio. For train and test, we take random noisy portion from first half of the noise file, whereas for test, we take for last half. All these audios are normalized and then feed into RNN or CNN layer.

4.2 Data Exploration

After preparing the data we started parsing the audio files into memory. We used python LibROSA module to load the audio files. Once loaded into memory, the audio file is stored as a 1-dimensional numpy array, representing a time series with a single floating point number that represents the change in the audio signal from one sample point to the next. LibROSA converts the samples to a single (mono) channel, and down-samples each to 22050 hz, which seems to be considered a good compromise quality for audio analysis. Hence a 1 second sample would be represented as 22050 floats, and a 4 second sample would be 4 times longer, with 88200 values. This is a considerable quantity of data for each sample, to make processing tractable.

To visualize the data a common technique for audio recordings is the waveform plot, which depicts the amplitude (relative loudness) of the sound at each successive time interval. Fig 6 shows the waveforms for 10 randomly chosen example of 2 classes (declarative and questions). In these plots, time is on the horizontal axis and amplitude on the vertical.

Matplotlib provides an alternative visualization method called spectrogram that calculates and plots the different intensities of the frequency spectrum. This creates (Fig 7) a different depiction of each sound:

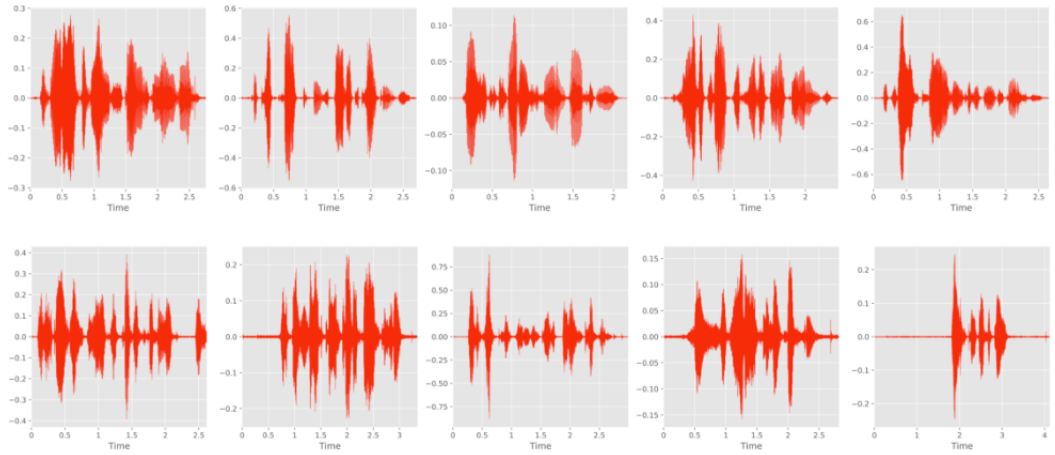


Fig. 6. **Wave plots for Declarative (Top) and Questions (Bottom) sentences**

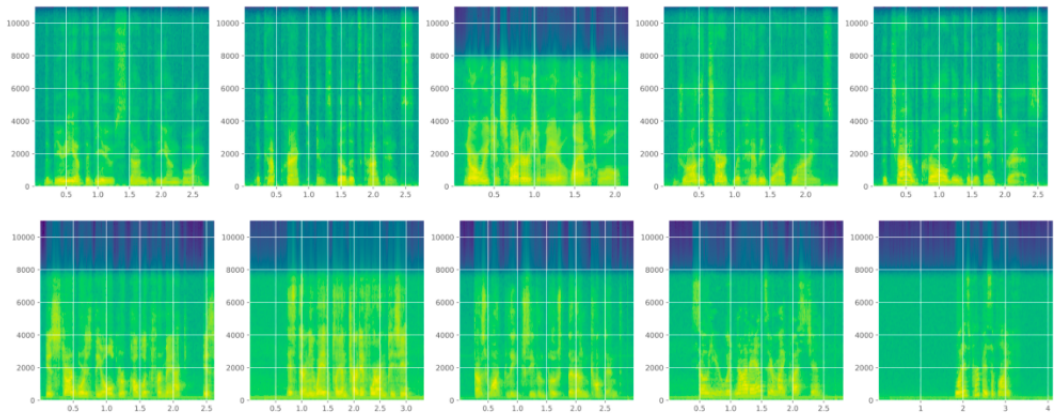


Fig. 7. **Spectrograms for Declarative (Top) and Questions (Bottom) sentences**

5 EVALUATION METHODS

In this section we will explain how our question classifier was implemented with two different neural network architectures. For the first step we will describe how features were extracted for input into the Recurrent Neural Network (RNNs) and Convolutional Recurrent Neural Networks (CRNNs), and how these networks are implemented and refined.

5.1 Feature Extraction

In the data exploration section we found each second of audio consists of 22050 data. The convention for neural networks is to have one input node per feature, so a 2 second sample would require 44100 input nodes, and then a similar order of magnitude of hidden nodes values which is too much data. To reduce the tens of thousands of data points in each file into a much smaller set of features of fixed number we use LibROSA feature extraction methods. This would allow the network to

compare features regardless of the duration of the audio sample. For our project we considered using 4 feature extraction methods:

- **Spectrogram:** Time-Frequency representation of the speech signal
- **Mel-frequency cepstral coefficients (MFCC):** The coefficient that collectively make up the short term power spectrum of a sound.

For our Recurrent Neural Network (RNN) model we used MFCC as feature. We used spectrogram as feature for the Convolutional Recurrent Neural Networks (CRNNs) model. Our two models are described in the following sections.

5.2 Recurrent Neural Networks (RNNs)

We considered 3 versions of Recurrent Neural Networks (RNNs) and compared their performances for this model. We used MFCC to create feature vector. We randomly divide this data in 75%, 5%, 20% percentile for respectively train, validation and test. First we applied MFCC on the raw sample data for each file and set number of mfcc to 40.

5.2.1 ***Single .wav single sample:*** In this model we considered the full *.wav* file as input in the RNN model. To do this we took the feature vector as 500×40 size of numpy matrix. When there are less than 500 values present in a matrix, we padded the vector with zeros. So for N training sample documents, we get $N \times 500 \times 40$ size matrix.
In our project, we used Keras Sequential Model. For the RNN part we used bidirectional GRU layer with 500 cells. We assumed that each vector will be at most 500 long. Since this classification problem is a many-to-one type sequence problem, we took output from the last cell of the 2^{nd} layer of LSTM.
To prevent over-fitting in the model we include a dropout policy between each LSTM layer. We used a value of 0.2, meaning there'd be a 20% chance that any neuron's activation output will be ignored and not propagated to its downstream connections. The idea is that this random throwing away of information helps prevent the network from learning simple spurious dependencies, and promotes the creation of complex co-adaptations between neurons of the hidden layers. By encouraging multiple neurons to involve in learning, they can learn on behalf of 'missing' neurons, resulting in the creation of multiple independent internal representations by different groupings of neurons across the network.
At the end of the network, we used a Dense layer with only 2 neurons because we are using one-hot vector encoding of declarative and question classes. Though it is a classification model we used *sigmoid* as activation function at this layer.
Another implementation decision was the choice of optimizer, where we used *Adam* optimizer with default learning rate and momentum. *categorical_crossentropy* was the lost function that we used in our network. We considered 20 number of epochs for our model as it took a long time to train the network but increasing the number of epoch may improve the performance we believe. We also considered batch size as 64. After testing the data we found that the accuracy is 97.00%.

5.2.2 ***Dividing each .wav into word chunk:*** In this model we divided each *.wav* files into several chunks based on each words. The idea of doing this is to divide streams of audio files up by quiet or silence within the stream. We used the *pydub* module from *AudioSegment*. The use of this library would be in making it possible to detect or split audio by word breaks in spoken language. We used *split_on_silence* method to cut an audio file into chunks based on silence in between the words. We set the parameters $min\_silence\_len = 100$ and $silence\_thresh$ to 5 less than the average dBFS of the audio file so that it can better cut the audio files on the silence. We then export all the chunks for each audio file into .wav format.

We then again load all the chunks for each full audio file and applied MFCC on them. We took the mean of MFCC at this time and set the number of MFCC to 40. So we got a 40×1 matrix for each chunk. We concatenated the vectors for each chunk horizontally for all the chunks of each audio files and found a vector of size $40 \times M$, where M is the number of chunks for each audio files. For training the dataset we converted the feature vector as 25×40 size of numpy matrix. When there are less than 25 values present in a matrix, we padded the vector with zeros. So for N training sample documents, we get $N \times 25 \times 40$ size matrix.
Here we again used Keras Sequential Model. For the RNN part we used bidirectional GRU layer with 25 cells. We assumed that each vector will be at most 25 long. Since this classification problem is a many-to-one type sequence problem, we took output from the last cell of the 2^{nd} layer LSTM. The dropout policy and decision on optimizer is same as the first model of RNN (Single .wav single sample). We also used the same activation function and loss function. The number of epoch and batch size is also same. In this model we got the accuracy of 98%.

5.3 Convolutional Recurrent Neural Network (CRNN)

In RNN model, we pass the mfcc feature vector to rnn cells and model eventually learn the important structure in the input features. Another way is to use a CNN layer to learn the structure in the feature vector and use that for RNN layer [9]. This CRNN model are faster converging since CNN layer already extracted the important relation and structure hidden in feature, also CNN layer can be parallelized.

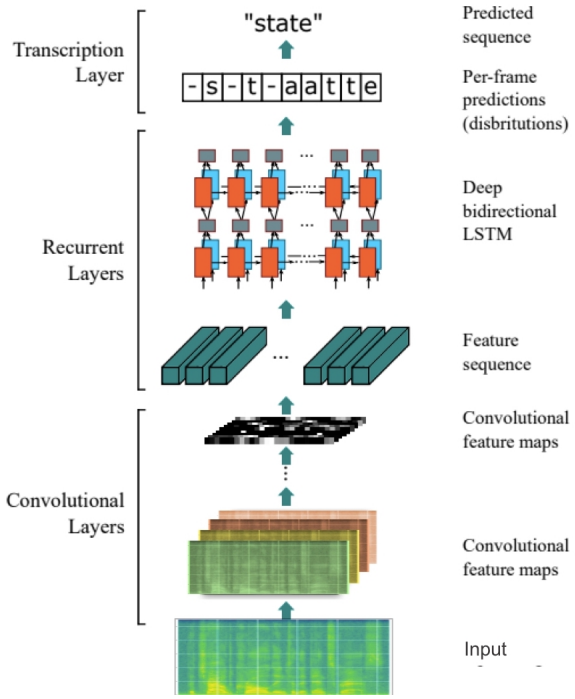


Fig. 8. The CRNN network architecture consists of three parts: 1) convolutional layers extract feature sequence from the input image; 2) recurrent layers predict a label distribution for each frame; 3) transcription layer translates the per-frame predictions into the final label sequence.

CNN layers are now quite common in extracting feature from 2D structure (e.g. image) [8]. For audio, we experimented with both mfcc feature vectors and spectrogram of the audio. We make a 2D feature vector for an audio where each row is a feature value for all time-stamp and each column is all feature in a specific time-stamp. To make feature vector of same shape, we add padding. Structure of the convolution layer is shown in figure 9. We use batch-normalization layer in each convolution block. For first convolutional layer, we use 32 kernal, but for rest we use 64. All convolutional layer are with 0.1 dropout. We use Bi-directional GRU as RNN layer. LSTM cells can also be used, but GRU are known for simpler and faster convergence. With CRNN, we experiment using both mfcc feature vector and spectrogram feature Result are discussed in 6.

Layer Type	Configurations
Dense	2 neurons, activation: sigmoid
Dropout	Param: 0.3
Bidirectional GRU	# hidden units: 100
Reshape	(None,32)
Dropout	Param: 0.1
MaxPooling	Window: 2x2, s:2
BatchNormalization	axis = 1
Convolution	#maps:64, k:3x3, s:1, p:1
Dropout	Param: 0.1
MaxPooling	Window: 2x2, s:2
BatchNormalization	axis = 1
Convolution	#maps:64, k:3x3, s:1, p:1
Dropout	Param: 0.1
MaxPooling	Window: 2x2, s:2
BatchNormalization	axis = 1
Convolution	#maps:64, k:3x3, s:1, p:1
Dropout	Param: 0.1
MaxPooling	Window: 2x2, s:2
BatchNormalization	axis = 1
Convolution	#maps:32, k:3x3, s:1, p:1
BatchNormalization	axis = 2
Input	(mfcc, spectogram)

Fig. 9. CRNN Network configuration summary. First row is the top layer. 'k', 's' and 'p' stand for kernel size, stride and padding size respectively.

5.4 Evaluation

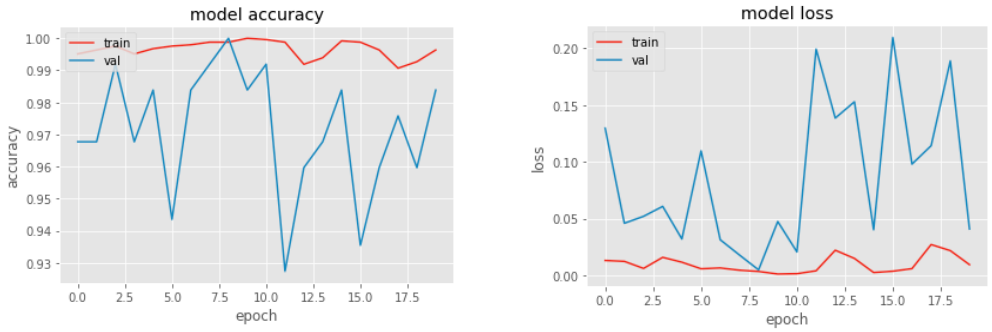
As performance criteria, we use cross-entropy loss function and Adam method as optimizer. Other loss functions like, L2 norm, KLD do not show better performance than cross-entropy. Moreover ours is a 2-class categorical problem, so categorical cross-entropy is good metric for evaluation.
Human evaluation is the best evaluation metric since they are the ultimate focus people. But in many cases, human annotation varies for a same sentences because of the context of the sentences or perception. We can not have true human evaluation because it will take long time and huge effort (most likely not free of cost).

6 RESULTS

We evaluate our models in metric of accuracy and how quickly model converges. In figure 10, we show our experiments' performance by accuracy. We can see that for our model simple RNN model

Model	Train Accuracy (%)	Accuracy (%)
RNN (mfcc, single wav)	99.51	96.95
RNN (mfcc, word chunck wav)	99.45	98.58
CRNN (mfcc)	98.39	96.96
CRNN (spectrogram)	98.34	97.43

Fig. 10. Model accuracy.



(a) Accuracy

(b) Loss

Fig. 11. Performance of mfcc feature in CRNN model.

works better.
Figure 11, we show how the accuracy and loss of mfcc feature converge by our CRNN model.

7 CHALLENGES AND FUTURE WORKS

In this project we only worked with a small IEEE corpus and on small amount of manually recorded audio files. The main challenge of this project is no appropriate speech dataset is available for question classification, specially for echo question. So, the dataset that we used is too small for applying deep neural networks. As this dataset is labeled by us so the ground truth can vary depending on the context and perspective. Also the word chunking method that we used based on silence is not 100% accurate because of not having enough silence between each word. So this may have an effect on our final result. In future we should consider stress and intonation in speech for better classification and need to create the dataset on our own. Also using text annotation with speech .wavs can be interesting to investigate. Another obvious next step is to consider other features to train the model, RMSE, pitch, Chromagram of a STFT, Tonnetz, etc.

8 CONCLUSION

In this report, we have presented two end-to-end deep learning-based speech systems: recurrent neural networks [5] and convolutional recurrent neural networks [9] which give state-of-the-art results in question classification on the IEEE database in two challenging scenarios: clear, conversational speech and speech in noisy environments. All components of the model are automatically trained, and are thus applicable to other domains for which labeled data is available. Detection accuracies achieved so far are highly encouraging, relative to the inherent difficulty of the task as measured by human labeler performance. We investigated several modeling alternatives for the components of the model (fully connected neural networks, convolutional neural networks).

We found performance largely independent of these choices, indicating on the one hand that our current system does about as well as possible given current modeling techniques and the inherent difficulty of the task and our limited representation of it. We believe this approach will continue to improve as we capitalize on increased computing power and dataset sizes in the future.

REFERENCES

[1] Nathaniel Blanchard, Patrick J Donnelly, Andrew Olney, Borhan Samei, Sean Kelly, Xiaoyi Sun, Brooke Ward, Martin Nystrand, and Sidney K D'Mello. 2016. Semi-Automatic Detection of Teacher Questions from Human-Transcripts of Audio in Live Classrooms.. In *EDM*. 288–291.

[2] Kofi Boakye, Benoit Favre, and Dilek Hakkani-Tür. 2009. Any questions? Automatic question detection in meetings. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*. IEEE, 485–489.

[3] Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, and Xiaofei He. 2017. Dialogue Act Recognition via CRF-Attentive Structured Network. *arXiv preprint arXiv:1711.05568* (2017).

[4] Antonio Gulli and Sujit Pal. 2017. *Deep Learning with Keras*. Packt Publishing Ltd.

[5] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

[6] Gil Keren and Björn Schuller. 2016. Convolutional RNN: an enhanced model for extracting features from sequential data. In *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 3412–3419.

[7] Anna Margolis and Mari Ostendorf. 2011. Question detection in spoken conversations using textual conversations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 118–124.

[8] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 512–519.

[9] Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence* 39, 11 (2017), 2298–2304.

[10] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26, 3 (2000), 339–373.

[11] Quang Vu, Laurent Besacier, and Eric Castelli. 2007. Automatic question detection: prosodic-lexical features and crosslingual experiments. (01 2007), 2257-2260 pages.

[12] Jiahong Yuan and Dan Jurafsky. 2005. Detection of questions in Chinese conversational speech. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*. IEEE, 47–52.