End-to-End Automatic Speech Recognition for Gujarati

Deepang Raval, Vyom Pathak, Muktan Patel, Brijesh Bhatt
Computer Engineering Department, Dharmsinh Desai University, Nadiad
deepangraval2012@gmail.com, angerstick3@gmail.com,
muktan123@gmail.com, brij.ce@ddu.ac.in

Abstract

We present a novel approach for improving the performance of an End-to-End speech recognition system for the Gujarati language. We follow a deep learning based approach which includes Convolutional Neural Network (CNN), Bi-directional Long Short Term Memory (BiLSTM) layers, Dense layers, and Connectionist Temporal Classification (CTC) as a loss function. In order to improve the performance of the system with the limited size of the dataset, we present a combined language model (WLM and CLM) based prefix decoding technique and Bidirectional Encoder Representations from Transformers (BERT) based post-processing technique. To gain key insights from our Automatic Speech Recognition (ASR) system, we proposed different analysis methods. These insights help to understand our ASR system based on a particular language (Gujarati) as well as can govern ASR systems' to improve the performance for low resource languages. We have trained the model on the Microsoft Speech Corpus, and we observe a 5.11% decrease in Word Error Rate (WER) with respect to base-model WER.

1 Introduction

ASR is the process of deriving the transcription (word sequence) of an utterance, given the speech waveform. Speech Recognition has been an active area of research for many decades. Initial work in ASR was based on statistical modeling techniques like Hidden Markov Model (HMM) (Baker, 1975) and used phonemes to represent distinct sounds that make up the word. With the rise of Deep Learning based techniques and the increasing availability of data, the End-to-End speech recognition systems started showing competitive results. Initial deep learning based ASR models, based on Recurrent Neural Network (RNN) and CTC (Graves et al., 2006), overcame the issues of statistical systems and provided the mapping of variable length input to output. With the further advancements in algorithms and resources, various complex deep learning architectures have been introduced for an effective End-to-End speech recognition system. End-to-End speech recognition for low resource languages has not gained significantly from the advancements in deep learning due to lack of training data compared to other high resource languages. Linguistic diversities[1] also makes it difficult to adopt models across languages.

In this paper, we present a speech recognition system for the Gujarati Language. Gujarati is a rich language consisting of 34 consonants and 13 vowels. While the more number of vowels may reduce the homophones, more number of consonants may increase the ambiguity.

The key contributions of this paper are as follows,

- We have adopted the state of the art ASR model described in (Amodei et al., 2015) for the Gujarati Language.
- We present a novel approach of combining two language models, 4-gram word-level language model (WLM) and bi-gram character-level language model (CLM) to improve performance of prefix decoding.
- We propose a *Spell Corrector BERT* based post-processing technique to correct erroneous prediction and further improve the performance of the ASR System.

[1] http://www.cs.cmu.edu/ ytsvetko/jsalt-part1.pdf

The proposed system reduced the WER to 65.54% from the initial 70.65%. We analyzed the system using the testing hypothesis and derived many useful insights on the performance of the system as well as the cause of the errors in the hypothesis. We analyzed that the errors produced in the Gujarati language are mainly because of interchanging/mismatching diacritics ('િ', 'ી', etc.), consonants ('ધ', 'જ', 'ય', etc.), independents ('આ', 'અ', 'ઇ', 'ઈ', etc.) and some homophones.

The remaining of the paper is organized as follows, Section 2 describes the Literature Survey of ASR system architectures. The proposed approach is described in Section 3. Section 4 constitutes the experiments conducted and its observations are followed in Section 5. Section 6 provides the conclusion of our work.

2 Related Work

Since the first ASR circuit developed by Bell Laboratories (Davis et al., 1952) in the 1950s, ASR has remained an active area of research. In early 1960's (Kenichi et al., 1966) presented a phoneme based speech recognition which involved the first use of speech segmenter in different portions of the input utterances. (Vintsyuk, 1968; Sakoe and Chiba, 1978) introduced the concept of the non-uniform time scale for alignment of speech patterns (dynamic wrapping). Both of these works lead to the Viterbi Algorithm (Viterbi, 1967) which had been an indispensable technique in ASR for decades. By the mid-1970s, the basic ideas of applying fundamental pattern recognition technology to speech recognition, based on Linear Predictive Coding (LPC) (Atal and Hanauer, 1971) methods, were proposed by Itakura (Itakura, 1975). CMU's Harpy System (Lowerre and Reddy, 1976), was the first ever system to use the Finite State Network (FSN) to reduce computation for matching in Speech Recognition. However, methods which optimized the resulting FSN did not come about until the early 1990's (Mohri, 1997), which were limited to small to medium vocabulary electronic based solutions for ASR.

The earlier approaches of Electronics based ASR were eventually replaced by statistical approaches with the introduction of HMM based speech recognition. The basic implementation of HMM based speech recognition model was first published in 1975 by Baker (Baker, 1975) at CMU. Further work on HMM continued with the introduction of first ever use of HMM for continuous speech recognition in 1976 (Jelinek, 1976). As the research continued, the HMM model was tried with various machine learning techniques including the HMM/ANN architecture in 1990 (Bourlard and Wellekens, 1990), HMM/GMM architecture in 1997 (Rodríguez et al., 1997) and HMM/SVM architecture in 1998 (Golowich and Sun, 1998). This wave for HMM continued till the introduction of RNN based approaches in early 2005.

The above approaches had major drawbacks such as

- It requires high task-specific knowledge, e.g. to design the state models for HMMs.
- It requires fairly complex parameter tuning as the pipeline contains multiple configurations.

(Graves et al., 2006) introduced a novel method for training RNNs to label unsegmented sequences directly, using CTC, thereby eliminating the above drawbacks and creating an End-to-End ASR system. With further enhancements in algorithms and resources, deep learning based End-to-End ASR systems got better and better and they started outperforming traditional ASR systems (Graves and Jaitly, 2014; Hannun et al., 2014). End-to-End ASR systems with encoder-decoder have shown competitive results (Chan et al., 2016). The RNN encoder-decoder paradigm uses an encoder RNN to map the input to a fixed-length vector and a decoder network to expand the fixed-length vector into a sequence of output predictions (Cho et al., 2014; Sutskever et al., 2014). Adding an attention mechanism to the decoder greatly improves the performance of the system, particularly with long inputs or outputs.

As we saw End-to-End ASR gives great results but at the cost of large data required to train it, which is not feasible for low resource languages. According to *Interspeech 2018, Low Resource Automatic Speech Recognition Challenge*[2], TDNN-based systems (Ped-

[2]https://tiny.cc/Interspeech2018

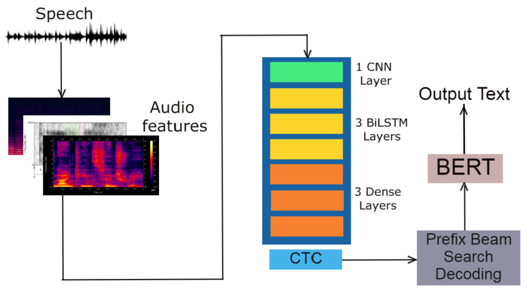

Figure 1: End-to-End Automatic Speech Recognition Process

dinti et al., 2015) are efficient in modelling long temporal context and performed well even in the low-resource setting (Fathima et al., 2018; Pulugundla et al., 2018). Even with the smaller amount of data, with some enhancements, End-to-End systems showed promising results (Billa, 2018).

The work similar to our approach are presented in *speech recognition primer*[3], and *Gujarati Automatic Speech Recognition*[4]. The first approach is based on a combination of CNN (Chua and Yang, 1988) and BiLSTM (Schuster and Paliwal, 1997) the latter approach uses a combination of 3 Gated Recurrent Units (GRUs). The first approach is designed for English, while the second is for Gujarati.

Our approach differs from the above two as follows,

- The model architecture described in our paper constitutes 1 CNN - 3 BiLSTM - 3 Dense layers.
- We present a more effective approach to decode the output using the prefix beam search algorithm along with the combination of the language model.
- Our approach introduces a post-processing technique to improve the performance of the system even more.

BERT is a neural network-based technique for natural language processing pre-training. The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications (Devlin et al., 2018). Multilingual-BERT uses a representation that is able to incorporate information from multiple languages (Pires et al., 2019).

3 Proposed Approach

Figure 1 describes the End-to-End speech recognition system proposed in the paper. This section describes the processing involved in the various stages.

3.1 Audio feature

We have used mel-frequency cepstral coefficients (MFCC) (Motlıcek, 2002) as features to represent the input audio signal. We convert the input audio signal into MFCC. The dimension of these features are *(Time_Steps, MFCCs)* and for the given batch size it is of dimension *(Batch_Size, Time_Steps, MFCCs)*. These features serve as the input to the deep learning model.

3.2 Model Architecture

The model architecture incorporates four major components: CNN layer, BiLSTM layer, Dense Layer, and CTC. Each component has its own importance and components like CNN layer, BiLSTM layer, and Dense layer have to be tuned as per the size of the input data.

Convolutions in frequency and time domains, when applied to the spectral input features prior to any other processing, can slightly improve ASR performance (Abdel-Hamid et al., 2012; Sainath et al., 2013). It also attempts to model spectral variance due to speaker variability, which is another reason to use the first layer as convolution (Amodei et al., 2015). We have used a single 1D-convolution layer with 200 filters with *ReLU* activation function with kernel size 11 and stride value as 2. Features extracted are then passed to a deep BiLSTM RNN (Schuster and Paliwal, 1997).

We have used 3 BiLSTM layers, each layer consisting of 200 BiLSTM units (400 LSTM units) and *tanh* as the activation function of each unit. When provided input from the con-

[3]https://github.com/apoorvnandan/speech-recognition-primer
[4]https://github.com/niteya-shah/Gujarati-Automatic-Speech-Recognition

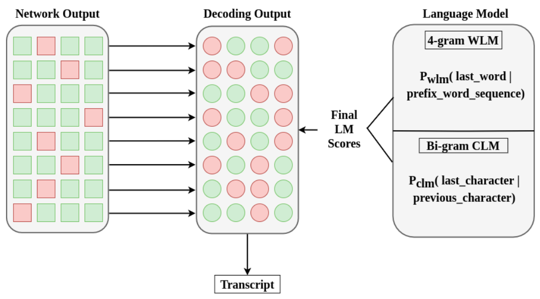
Network Output
Decoding Output
Language Model
4-gram WLM
P_{wlm}(last_word | prefix_word_sequence)
Bi-gram CLM
P_{clm}(last_character | previous_character)
Final LM Scores
Transcript

Figure 2: Working of decoding algorithm using WLM and CLM

volution layer, the BiLSTM layer gives us output as *(Batch_Size, Convolved_Time_Steps, LSTM_blocks)*. Features extracted are then passed to a DNN (Hinton et al., 2006), which consists of 3 layers where the first 2 layers consist of 200 units. The number of units in the last layer is equal to the number of characters in the languages.

While training, a common technique for mapping variable-length audio input to variable-length output is the CTC algorithm (Graves et al., 2006) coupled with an RNN. The CTC-RNN model performs well in End-to-End speech recognition with grapheme outputs (Graves and Jaitly, 2014; Hannun et al., 2014; Maas et al., 2014, 2015). Given the network outputs, CTC maximizes the probabilities of the correct labelings. The CTC objective function is differentiable thus the network can then be trained with standard backpropagation through time (Werbos, 1990).

3.3 Decoding

We propose an enhanced language model based prefix decoding which uses our custom built 4-gram word-level language model (WLM) and a bi-gram character-level language model (CLM) (Brown et al., 1992). Here we refer to it as *Prefix with LMs'*. Both of these models were created using the whole *Gujarati Wikipedia*[5]. Using prefix decoding with two language models (WLM and CLM) makes decent corrections, as it introduces language constraints at both word and character scope. We have compared this approach with greedy decoding (Maas et al., 2014) and prefix decoding (Maas et al., 2014).

As shown in the figure 2, the output from the network is passed through the prefix decoding algorithm which incorporates a WLM and a CLM. The WLM will score the last word given in the prefix word sequence. This score's influence can be controlled by WLM weight (wlm). Similarly, CLM will score the last character to be appended given its previous character. This score's influence on the new prefix is controlled by CLM weight (clm). We encompassed the insertion bonus by multiplying the count of words in prefixes to avoid bias towards shorter prefixes. To control the influence of insertion bonus we used beta (β).

[5]https://gu.wikipedia.org/wiki/

3.4 Post Processing

We propose a BERT based post-processing technique to improve the output of speech recognition systems. The BERT model is used to correct the spelling of the predicted output words. Here we term this technique as *Spell Corrector BERT*. Figure 3 describes the working of sentence correction algorithm using BERT. For a sentence, we iterate through all the words during which we find the replacements for the current word by finding its corresponding zero, one, or two edit substitutes from the Wikipedia corpus. Using the list produced by this approach, we can verify that if the word predicted is correct, it would be already present in the list and needs not to be replaced.

If the current word is not present in the replacements list, then that word is replaced with $[MASK]$ and the sentences are generated by replacing the $[MASK]$ with the word replacements from the replacements list. Further, the sentences are tokenized and passed to the BERT model. As an output, BERT returns the list of replaced words and their respective probabilities w.r.t. the sentence. From this list, we select K word replacements with the highest probability and append this list of word replacements to the *output_list*. Once all the words are iterated, a final *output_list* is generated which contains a list of all words with at most K replacements for each word. Given a sentence containing 3 words, અમદાવાદર એપોર્ટ પર where અમદાવાદર and એપોર્ટ are incorrect, this process gives the *output_list*= [[અમદાવાદ, અમદાવાદમા,...] , [એરપોર્ટ, પોર્ટ,...] , [પર]]. This *output_list* is passed to a combinator which generates sen-

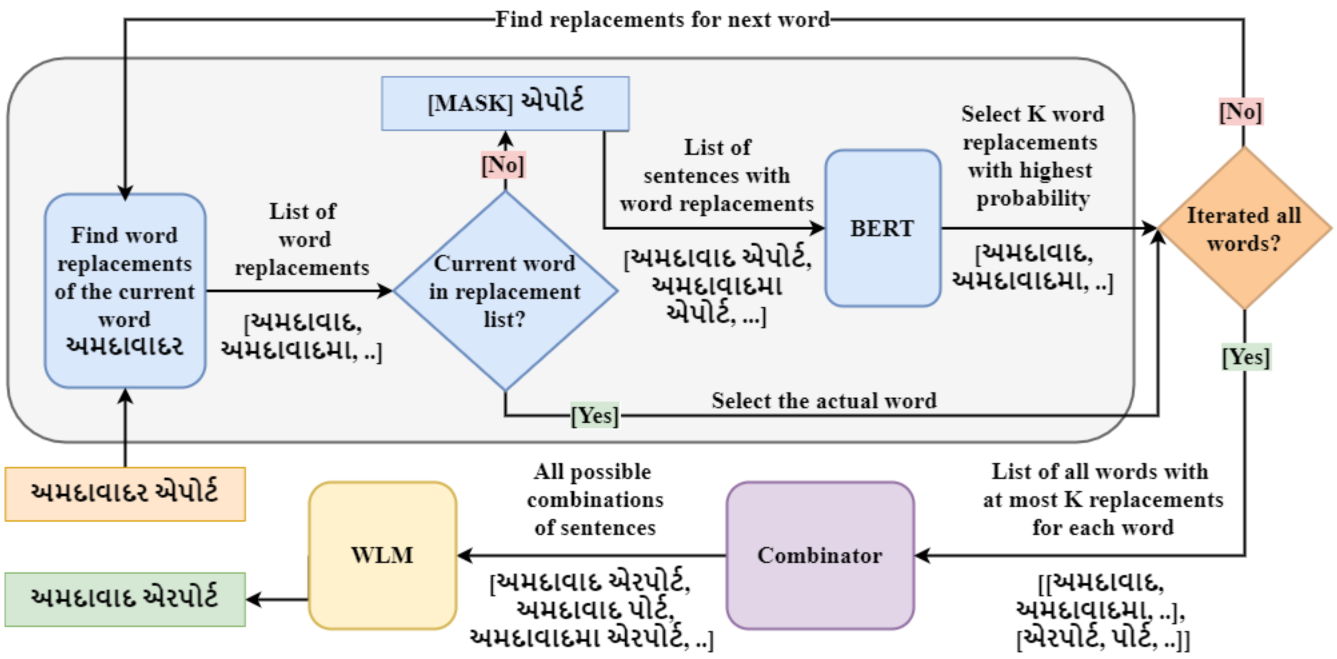
Find replacements for next word
[MASK] એપોર્ટ
[No]
List of sentences with word replacements
Select K word replacements with highest probability
[No]
Find word replacements of the current word અમદાવાદર
List of word replacements
[અમદાવાદ, અમદાવાદમા, ..]
Current word in replacement list?
[અમદાવાદ એપોર્ટ, અમદાવાદમા એપોર્ટ, ...]
BERT
[અમદાવાદ, અમદાવાદમા, ..]
Iterated all words?
[Yes]
[Yes]
Select the actual word
અમદાવાદર એપોર્ટ
All possible combinations of sentences
List of all words with at most K replacements for each word
WLM
Combinator
અમદાવાદ એરપોર્ટ
[અમદાવાદ એરપોર્ટ, અમદાવાદ પોર્ટ, અમદાવાદમા એરપોર્ટ, ..]
[[અમદાવાદ, અમદાવાદમા, ..], [એરપોર્ટ, પોર્ટ, ..]]

Figure 3: Working of Spell Corrector BERT

tences by combining the words. If there is a list contain a single word in it, it is selected as it is and if there is a list of word replacements then various combinations are produced using the combinator. The output of this process will be *output_sentences*= [અમદાવાદ એરપોર્ટ પર , અમદાવાદ પોર્ટ પર , અમદાવાદમા એરપોર્ટ પર ,...]. To select the best sentence out of all the sentences, a WLM is used for sentence scoring and the sentence with the highest score is selected as the $final\_output$ = અમદાવાદ એરપોર્ટ પર.

4 Experiments

4.1 Dataset

We have used Microsoft Speech Corpus available for Gujarati[6] which contains approximately 22,807 training examples and 3,075 testing examples. The dataset contains an eclectic collection of speakers where the length of a single audio utterance is 6.35±2.33 seconds. Out of 22,807 training examples, we have used 16,000 utterances (≈28.2 Hours) for training and 4,807 (≈8.5 Hours) utterances for validation and all the testing examples i.e. 3,075 (≈5.4 Hours) utterances for inferencing.

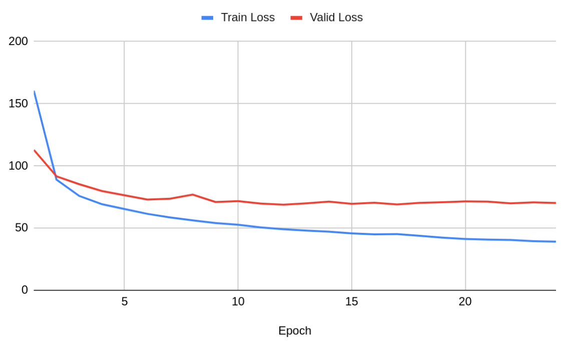

Figure 4: Training and Validation Loss for the model

4.2 Training

The training data, as well as validation data, was divided into 7 batches and the model was trained for 24 epochs and 92 hours on T4 GPU with 16 GB of GPU-memory. The model consists of 2,744,676 total parameters. We used *Adam* optimizer for gradient descent, and for calculating the loss we used CTC loss function.

4.3 Decoding

We have used Gujarati data scraped from Wikipedia containing 2,501,841 words with a vocabulary size of 163,170 words. We used this corpus to create the statistical word-level language model and produced 1,570,614 4-grams. We also used the same corpus to create the sta-

[6]https://msropendata.com/datasets/7230b4b1-912d-400e-be58-f84e0512985e

Table 1: Sentence and its corresponding output through various decoding techniques.

Actual	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમામ તૈયારીઓ કરી દેવાઈ છે
Greedy	અમદાવાદાર પર પણ સુરક્ષાઅને લઈને તમમ ટેરે કરી જવોાય છે
Prefix with LMs'	અમદાવાદાર પર પણ સુરક્ષાને લઈને તમમ કેરે કરી જવાય છે
Prefix with LMs' & Spell Corrector BERT	અમદાવાદ પર પણ સુરક્ષાને લઈને તમામ કેરે કરી જવાય છે

Table 2: Distribution of single letter error words

Technique	Consonants	Diacritic	Independents
Greedy	66.41% (1,962)	28.23% (834)	5.34% (158)
Prefix with LMs'	66.52% (1,824)	27.97% (767)	5.5% (151)
Prefix with LMs' & Spell Corrector BERT	52.90% (829)	38.67% (606)	8.4% (132)

Table 3: Techniques and their corresponding WER

Techniques	Word Error Rate (%)
Greedy	70.65
Prefix without Language Model	69.95
Prefix with WLM	69.53
Prefix with CLM	68.64
Prefix with LMs'	68.23
Prefix with LMs' & Spell Corrector BERT	65.54

tistical character-level language model to create bi-grams for each alphabet of the Gujarati language. For prefix decoding, the beam width was taken as 50 and all other parameters were decided using cross-validation. The algorithm of *Prefix with LMs'* recorded a 2.42% decrease in WER w.r.t. system using greedy decoding technique.

4.4 Post Processing

Spell Corrector BERT is used to further improve the output produced by *Prefix with LMs'*. We have used a pre-trained BERT multilingual model by Google[7] combined with a 4-gram WLM as the core components of *Spell Corrector BERT*. The algorithm of *Spell Corrector BERT* recorded a 2.69% decrease in WER w.r.t. standalone *Prefix with LMs'*. The table 3 shows the comparison of WER for different techniques.

[7] https://github.com/google-research/bert/blob/master/multilingual.md

5 Observation

5.1 Comparison of various decoding and post-processing techniques

We have tested the performance of the decoding technique and post-processing by observing the distribution and frequency of the single letter error words. Table 1 shows a sample testing sentence as well as the hypothesis generated by our model using various decoding techniques and post-processing techniques. Table 2 describes the distribution of single-letter error words observed in different approaches. Here, the count of errors due to consonants/diacritics/independents w.r.t. the total count of single-letter error words in each decoding technique is shown as a percentage. It shows that, *Prefix with LMs'* and *Prefix with LMs' & Spell Corrector BERT* post-processing, both help in reducing single letter error words.

Table 2 also shows count of single letter error words. Subsequently, from this count we can conclude that, percentage decrease of the error in consonant, diacritic and independents using *Prefix with LMs'* is 7%, 8% and 4% respectively w.r.t greedy decoding, while using *Prefix with LMs' & Spell Corrector BERT*, the percentage decrease of the error in consonant, diacritic and independents is 57.74%, 27.00% and 16.45% respectively w.r.t. greedy decoding.

We observe that, with a lesser number of incorrect characters, *Spell Corrector BERT* either retains WER or in the majority of the cases, will improve WER significantly. Table 4 shows sample sentences with a different number of erroneous words for comparison of the performance of *Spell Corrector BERT*.

5.2 System Analysis

The system analysis is performed on the model hypothesis decoded using *Prefix with LMs' & Spell Corrector BERT*. We have evaluated the performance of the proposed model on 3,075 test examples. Out of total erroneous words, 7.19% words have one letter error with similar sounding alphabets of letters interchange. e.g. 'શ' → 'સ', 'ઇ' → 'ઈ', 'િ' → 'ી'. The interchange of consonants/diacritics/independents is due to the factors like, noise, channel variability, speaker variability, anatomy of the vocal tract, speed of speech, regional and social dialects, homophones (Forsberg, 2003). Any incorrect word in the sentence is replaced on the basis of probability and hence WLM, CLM, or BERT is not solely responsible for the selection of any word, it is the combination of the probabilities that results in the final output.

5.2.1 Error due to consonants/independents/diacritics

Table 5 displays examples of words which have a single-letter error due to consonants, independents, and diacritics. *Ref.* depicts the actual word in the sentence, *Hyp.* denotes the output word, *Ref. Freq.* shows the count of reference word in the corpus, *Hyp. Freq.* is the count of hypothesis word in the corpus, *Ref. → Hyp.* shows the character that is replaced and *Type* shows the type of error in the inference word. From this table, we can observe that, despite the hypothesis word being infrequent in the corpus, the similar sounding letters in the reference word gets replaced. This advocates the idea that the replacement of the similar sounding letters from the words is also one of the factors inducing the errors in the system, irrespective of the words' frequency in the corpus.

Out of the total 1,567 one letter error words, 52.90% errors are due to single consonant mismatch. The top three incorrectly predicted consonants are 'ક', 'ર', 'ત', with frequencies 102, 92, and 79 respectively. Together they contribute to 32.93% of total errors due to consonants.

In figure 5, the connection between two consonants represents the error of interchanging/misplacing these consonants with each other. For example, the connection between 'શ' and 'સ' indicates that these consonants are generally interchanged/misplaced with each other, in a word predicted by the system.

Out of the total 1,567 one letter error words, 8.40% errors are due to single-independent mismatch. The top three incorrectly predicted independents are 'ઈ', 'ઇ', 'અ', with frequencies 81, 21, and 13 respectively. Together they contribute to 87% of total errors due to independents. We observe that (['ઇ', 'ઈ'],['ઉ', 'ઊ']) are more vulnerable to being misplaced/interchanged.

Out of the total 1,567 one letter error words, 38.67% errors are due to single-diacritic mismatch. The top three incorrectly predicted independents are 'ી', 'ે', 'ા', with frequencies 213, 120, and 76 respectively. We observe that frequency of diacritic 'ી' and 'ે'is greater than the sum of the frequencies of remaining diacritics, and they constitute to 67% of total incorrectly predicted diacritics.

The output ([શરૂ, સરૂ],[સાડા, સાળા],[કોઈ, કોએ]) are interesting cases as the hypothesis words are not present in the corpus. The predicted output misplaced (['શ', 'ડ', 'ઈ']) with similar sounding (['સ', 'ળ', 'અ']) respectively without any prior knowledge of the word. This is due to the character by character prediction approach of our model.

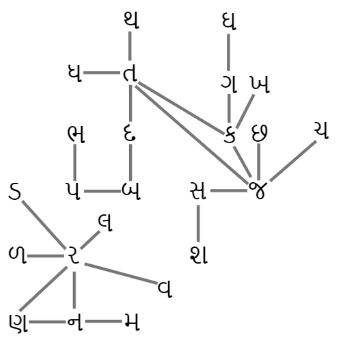

Figure 5: Interchanging consonants which results in erroneous prediction.

Table 4: Sample sentences for comparing performance of BERT

Actual	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમામ તૈયારીઓ કરી દેવાઈ છે
At most one error per word	અમદાવદ એપોર્ટ પર સુરક્ષામે લઈ તમાન તૈયરીઓ જરી દેરાઈ છે
Spell Corrector BERT Output	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમામ તૈયારીઓ કરી દેવાઈ છે
At most two errors per word	અમદવા એરપર્ પર સરક્ષને લઈ તમ તૈયરઓ કરી દેવ છે
Spell Corrector BERT Output	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમે તૈયારીઓ કરી દેવ છે
At least 1 word with error greater than 2	અમદાવાદાર પર પણ સુરક્ષાઅને લઈને તમમ ટેરે કરી જવોાય છે
Spell Corrector BERT Output	અમદાવાદ પર પણ સુરક્ષાને લઈને તમામ કેરે કરી જવાય છે

Table 5: Examples of words which have single letter error

Ref.	Hyp.	Ref. Freq.	Hyp. Freq.	Ref.→Hyp.	Type
શરૂ	સરૂ	282	0	શ → સ	Consonant
ત્યારે	ક્યારે	377	15	ત → ક	Consonant
ધરાઈ	ધરાઇ	9	15	ઈ → ઇ	Independent
ઊમેર્યું	ઉમેર્યું	2	11	ઊ → ઉ	Independent
પ્રારંભિક	પ્રારંભીક	4	0	◌િ → ◌ી	Diacritic
ચૂંટણીમાં	ચુંટણીમાં	50	12	◌ૂ → ◌ુ	Diacritic

5.2.2 Error due to homophones

Out of a total of 606 words that had a single diacritic error, only 2.97% of errors were due to homophones which is a small fraction of the total amount of errors in the inference from the ASR system. This might be because Gujarati is mostly a phonetic language with only a few exceptions. Also, the number of alphabets (vowels and consonants) in Gujarati are more than that in English. By reducing diacritic errors, we can resolve errors due to homophones. This helps us to understand that our system is not much affected by error due to homophones. Table 6 shows the incorrectly predicted homophones.

5.2.3 Effect of word frequency on error

To understand the effect of frequency on error, we calculated the frequency of the predicted words in the test dataset. We categorized the words into three different categories based on the correctness of the word referenced to their occurrence in the testing dataset. The three different categories are,

- ACPW: Words that are always predicted correctly.
- AIPW: Words which are always predicted incorrectly
- CAIPW: Words which are predicted correctly as well as incorrectly at times.

Count of ACPW, AIPW and CAIPW is 1,069, 7,809 and 1,604 respectively. Mean frequencies/occurrences of ACPW, AIPW and CAIPW is 1.11, 1.34, and 15.38 respectively. Words which are ACP, AIP and CAIP in testing, are shown in training with a mean frequency/occurrence of 4.75, 4.86, and 77.81 respectively with a count of 857, 5,764 and 1,594 respectively. This gives us a rationalization for the fact that our system is able to learn from the utterances shown in the training and can infer unseen examples too.

Table 6: Examples of words which are incorrectly predicted homophones

Reference Word	કર્તા (Actor)	રવિ (Sun)	પીતા (Drinking)
Hypothesis Word	કરતા (Than)	રવી (Winter Crop)	પિતા (Father)

Table 7: Sample words which are predicted correctly as well as incorrectly

Words in Testing	Total Count	Wrong count	Right count	Correctness(%)
આજે	81	32	49	60.49
રાજકોટના	2	1	1	50.00
તાકાત	2	1	1	50.00
વિભાગ	7	4	3	42.86
આવે	67	24	43	64.18
Average				53.50

We also analyzed the correctness of the words from the set CAIPW. 8.32 out of 15.38 mean frequencies of CAIPW are correct and the remaining 7.06 out of 15.38 are incorrect. Table 7 shows some examples of the correctly as well incorrectly predicted words with the amount of correctness. This gives an explanation for how words are predicted correctly as well as incorrectly with the same proportion.

5.2.4 Error due to half-conjugates

This type of error occurs due to the mismatch in the speed of utterance. The fast-conjugate error occurs when a word is uttered too quickly but the hypothesis word is predicted slow, e.g., (મુખર્જી → મુખરજી). When a word is uttered slowly but the hypothesis word is predicted fast then this type of error is called slow-conjugate error (ગયો → ગ્યો). Out of total erroneous predicted words, 2.4% of them have half-conjugate error and out of those 2.4% words, 10% words consist of pure half-conjugate erroneous words. This justifies the fact that the error due to half-conjugate is trivial and thus the variation in speaker speed is not a significant factor due to which error occurs in inference by our system.

6 Conclusion

In this paper, we have presented an End-to-End speech recognition system for Gujarati. We propose a prefix decoding technique that uses two language models to improve the performance of the system. We have also used a BERT based spelling corrector model in a post-processing step to further improve the performance. We observe that the proposed approach reduces the overall WER by 5.11%.

While deep learning models require a lot of training data for better results, in this paper we showed that without increasing the training data we can improve the performance. This is particularly important for a resource-constrained language like Gujarati. We are optimistic that with an increase in data our optimizations would perform even better.

We explored and analyzed the inferences from our ASR system to gain key insights which consist of checking the correctness of the word error due to consonants, diacritics, independents, half-conjugates, and homophones. These insights can help to understand our ASR system based on a particular language (Gujarati) as well as can govern ASR systems' to improve the performance for low resource languages.

Acknowledgement

Param Shavak supercomputer was used to perform some experiments of this research work. We are thankful to GUJCOST and Department of Science and Technology, GoG for establishing supercomputer facility.

References

O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn. 2012. 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)*, pages 4277–4280.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2015. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595.

Bishnu S Atal and Suzanne L Hanauer. 1971. Speech analysis and synthesis by linear prediction of the speech wave. *The journal of the acoustical society of America*, 50(2B):637–655.

J. Baker. 1975. The dragon system–an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29.

Jayadev Billa. 2018. Isi asr system for the low resource speech recognition challenge for indian languages. pages 3207–3211.

H. Bourlard and C. J. Wellekens. 1990. Links between markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

L. O. Chua and L. Yang. 1988. Cellular neural networks: theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272.

Ken H Davis, R Biddulph, and Stephen Balashek. 1952. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Noor Fathima, Tanvina Patel, C Mahima, and Anuroop Iyengar. 2018. Tdnn-based multilingual speech recognition system for low resource indian languages. In *INTERSPEECH*, pages 3197–3201.

Markus Forsberg. 2003. Why is speech recognition difficult.

Steven E. Golowich and Don X. Sun. 1998. A support vector/hidden markov model approach to phoneme recognition, in. In *Center for Media Technology (RCMT), School of Creative Media, City University of Hong Kong, Hong Kong*, pages 125–130.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. volume 2006, pages 369–376.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1764–II–1772. JMLR.org.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567.

Geoffrey E. Hinton, Simon Osindero, and Y. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

Fumitada Itakura. 1975. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on acoustics, speech, and signal processing*, 23(1):67–72.

F. Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Maeda Kenichi, Sakai Toshiyuki, and Doshita Shuji. 1966. Phonetic typewriter system. US Patent 3,265,814.

B Lowerre and R Reddy. 1976. The harpy speech recognition system: performance with large vocabularies. *The Journal of the Acoustical Society of America*, 60(S1):S10–S11.

Andrew Maas, Ziang Xie, Dan Jurafsky, and Andrew Ng. 2015. Lexicon-free conversational speech recognition with neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 345–354, Denver, Colorado. Association for Computational Linguistics.

Andrew L. Maas, Awni Y. Hannun, Daniel Jurafsky, and Andrew Y. Ng. 2014. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *CoRR*, abs/1408.2873.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.

Petr Motlıcek. 2002. Feature extraction in speech coding and recognition. Technical report, Technical Report of PhD research internship in ASP Group, OGI-OHSU,< http ....

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? *CoRR*, abs/1906.01502.

Bhargav Pulugundla, Murali Karthick Baskar, Santosh Kesiraju, Ekaterina Egorova, Martin Karafiát, Lukás Burget, and Jan Cernockỳ. 2018. But system for low resource indian language asr. In *INTERSPEECH*, pages 3182–3186.

Elena Rodríguez, Belén Ruíz, Ángel García-Crespo, and Fernando García. 1997. Speech/speaker recognition using a hmm/gmm hybrid model. In *Audio- and Video-based Biometric Person Authentication*, pages 227–234, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tara Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. 2013. Deep convolutional neural networks for lvcsr. pages 8614–8618.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Taras K Vintsyuk. 1968. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

P. J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.