IMPROVED CONTEXTUAL RECOGNITION IN AUTOMATIC SPEECH RECOGNITION SYSTEMS BY SEMANTIC LATTICE RESCORING

Ankitha Sudarshan[1], *Vinay Samuel*[2], *Parth Patwa*[3], *Ibtihel Amara*[4], *Aman Chadha*[5,6†]

[1]Purdue University [2]Carnegie Mellon University [3]University of California Los Angeles
[4]McGill University [5]Stanford University [6]Amazon AI
[1]sudarsh0@purdue.edu [5,6]hi@aman.ai

ABSTRACT

Automatic Speech Recognition (ASR) has witnessed a profound research interest. Recent breakthroughs have given ASR systems different prospects such as faithfully transcribing spoken language, which is a pivotal advancement in building conversational agents. However, there is still an imminent challenge of accurately discerning context-dependent words and phrases. In this work, we propose a novel approach for enhancing contextual recognition within ASR systems via semantic lattice processing leveraging the power of deep learning models in accurately delivering spot-on transcriptions across a wide variety of vocabularies and speaking styles. Our solution consists of using Hidden Markov Models and Gaussian Mixture Models (HMM-GMM) along with Deep Neural Networks (DNN) models integrating both language and acoustic modeling for better accuracy. We infused our network with the use of a transformer-based model to properly rescore the word lattice achieving remarkable capabilities with a palpable reduction in Word Error Rate (WER). We demonstrate the effectiveness of our proposed framework on the LibriSpeech dataset with empirical analyses.

***Index Terms*—** speech recognition, lattice re-scoring, contextual speech recognition, word lattices

1. INTRODUCTION

Recognizing spoken language accurately and efficiently is a complex task due to variability in the source of speech such as pronunciation, dialects, vocabulary, accents, articulation, etc.

Semantic interpretation of a speech is crucial in ASR systems. Let us take the following example. *"I am going to a bank to deposit a check"*. Without context, the word bank could refer to either a financial institution or either the edge of a river.

To bridge this contextual gap in ASR systems, semantic lattice processing is a key component in contributing to better recognition of situational context conditions. This technique utilizes a lattice structure to represent the relationships between words and phrases in a sentence. It is created by analyzing the audio input and identifying possible word and phrase combinations and their associated probabilities. This information is then used to create a graph-like structure, where each node represents a word or phrase, and the edges represent the relationships between them [1].

In this study, our primary emphasis centers on lattice rescoring, a technique designed to efficiently re-evaluate the likelihood of potential speech hypotheses. While numerous lattice re-scoring techniques have been documented in the literature [2, 3], our research introduces a novel approach tailored to bolster contextual information within ASR systems.

Our key contributions are as follows:
1) Lattice re-scoring, which refines recognition results through the integration of language model probabilities from different language models, enhancing transcription accuracy and overall system performance.
2) Employing a Transformer architecture for our neural language model, enhancing lattice re-scoring with state-of-the-art contextual modeling.
3) Achieving a 1.36% reduction in word error rate when compared to state-of-the-art models sharing a similar architectural framework.

2. RELATED WORK

Many end-to-end (E2E) ASR approaches, such as connectionist temporal classification (CTC) [4], recurrent neural network transducer (RNN-T) [5], attention-based encoder-decoder (AED) [6, 7], have been employed in voice assistants. The E2E model jointly optimizes the overall recognition pipeline during training and generates the output word sequence directly. However, one drawback of this system is its struggle to recognize rarely occurring words in the training data, like songs or person names.

Contextual recognition of ASR systems is important primarily for voice assistants because they need to recognize the names of a user's contacts, the names of artists in a user's music library, etc. Current contextual biasing methods for different E2E ASR models are shallow fusion [8, 9], attention-

[†]Work does not relate to position at Amazon.

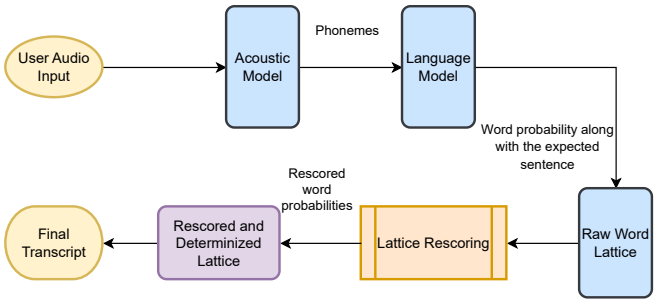

Fig. 1. **Global overview of our framework.** Our framework includes audio input, DNN acoustic model, lattice creation, language model integration, alignment, transformer re-scoring, and transcript generation.

based deep context [10, 11, 1] and trie-based deep biasing [12, 13].

In [1, 11], the confusion between similar phrases is observed and reduced by injecting phoneme information or training with difficult negative examples. The most widely used technique for contextual biasing in ASR is called on-the-fly re-scoring. This method was originally applied to hybrid ASR models in [14]. It involves composition of a weighted finite state transducer (WFST) representing the ASR model with a novel WFST representation of the bias terms, so that the weights on the bias terms are modified "on the fly" at inference time. For E2E models, the bias terms are still compiled into a WFST representation, but the composition of that WFST with the ASR model happens at every decoding time-step - the likelihood of the current hypothesis according to the ASR model is combined with a score from the bias WFST.

A separate re-scoring module can not adequately handle the error introduced by the upstream acoustic network. Some architectures, such as[10, 15] can integrate contextual information into acoustic model to improve the output posterior distribution corresponding to the related contextual words and fit the subsequent external LM well. As the size of the possible contextual word list grows in real applications, the accuracy and latency of the system descend rapidly due to dispersed attention score and heavy attention computation. Moreover, in practice, it is hard to obtain compact and accurate contextual information in advance. Consider music search - we may face a large contextual word list (size in thousands) containing popular songs. In this case, E2E context bias cannot work well due to the large size and low quality of the contextual word list, leading to performance degradation [16].

3. METHODOLOGY

3.1. Background

For ASR systems, the initial decoding process generates a lattice—a directed acyclic graph (DAG) representing a spectrum

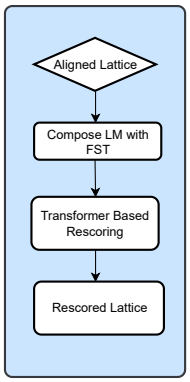

Fig. 2. **Lattice re-scoring strategy.** Alignment is involved, n-gram language model integration, and Transformer-based re-scoring to enhance contextual features in the final transcript.

of potential word hypotheses and their associated scores. The fundamental purpose of lattice re-scoring is to elevate the accuracy of these ASR hypotheses by post-processing and re-ranking them within the lattice. We explore the mathematical and algorithmic aspects of lattice re-scoring, showcasing its role in enhancing ASR accuracy. We use a custom Transformer model with positional encoding, multiple Transformer encoder layers (controlled by n layers), an input embedding layer, and an output linear layer for sequence prediction. Let $A(a)$ represent the acoustic score for arc a in the lattice. This score is typically based on the acoustic model and represents how well the audio aligns with the word associated with the arc. Let $P(w1, w2, ..., wN)$ denote the language model probability for the word sequence $(w1, w2, ..., wN)$ based on the Transformer model. This probability reflects the likelihood of the entire word sequence occurring in the context of the language. Given a path P through the lattice, the word sequence probability $P(P)$ can be computed as the product of acoustic scores and language model probabilities for the words along the path: $P(P) = \prod_a (A(a) \cdot P(w))$, for all arcs a and words w in the path. The path $P*$ through the lattice that maximizes the joint probability of the word sequence: $P* = argmax_P P(P)$

3.2. Lattice Re-scoring

We provide in Figure 1 and Figure 2, respectively, our overall framework and the proposed lattice re-scoring strategy. We use the DNN-refined predictions for creating word lattices as an intermediate representation of the ASR output. This decoding process generates lattice-like outputs that contain phone-level alignments and associated scores and eventual word alignments.

Each path in the lattice has a score based on the ASR system's confidence. However, we can improve the transcription by reevaluating these paths using a neural LM, which captures

the likelihood of different word sequences based on linguistic context. We use our custom-trained transformer to perform re-scoring. A Transformer-based re-scoring approach, as opposed to traditional n-gram methods, introduces novelty by leveraging advanced neural network architectures that are designed to handle sequences more effectively and capture complex language patterns.

This transformation is done by computing the conditional probability of the word sequence in each path given the language model and combining it with the original acoustic likelihood score. The result is a new lattice where paths have modified scores that reflect both acoustic and language model information, enhancing transcription accuracy.

The scores from the neural LM are

converted to log-likelihoods to be combined with the original lattice scores. Once we have the lattice paths re-scored using the neural LM and have converted the scores to log-likelihoods, we combine these scores with the original lattice scores. This step helps integrate the language model probabilities into the lattice.

By combining these scores, the lattice paths that were previously assigned lower scores by the ASR system but have higher probabilities according to the LM are promoted, resulting in a better transcription of the input audio than the original input.

A prominent mathematical formula in the lattice creation process in ASR is related to the computation of the overall likelihood or score of a path through the lattice. This score is typically calculated as the sum of individual acoustic and language model scores along the path.

$$\text{Path Score} = \sum_{i=1}^{N} \Big(\log\big(P(\text{word}_i|\text{word}_{i-1})\big) + \log\big(P(\text{Acoustic features}_i|\text{word}_i)\big) \Big) \quad (1)$$

where, N represents the number of words in the path, word_i is the i^{th} word in the path, $P(\text{word}_i|\text{word}_{i-1})$ is the conditional probability of transitioning from word_{i-1} to word_i using the language model, and $P(\text{acoustic features}_i|\text{word}_i)$ is the probability of observing acoustic features at position i given word_i using the acoustic model.

4. EXPERIMENTAL DETAILS

4.1. Data Corpus and Preprocessing

We use the LibriSpeech dataset [17], which consists of approximately 1000 hours of read English speech with a sampling rate of 16 kHz. This dataset provides a substantial and high-quality source of audio data, ensuring the robustness and generalizability of our proposed method, For data preprocessing, we utilize the Kaldi toolkit. We prepared the data in the Kaldi format organizing it into training, validation, and test sets. The format consists of two main components: the archive file (.ark) and the corresponding index file (.scp).file contains binary data and is typically organized as a sequence of key-value pairs, where the key is a unique identifier (usually a string) associated with the data, and the value is the binary data itself.

Example: `<ark>`
`<key1> <binarytoken1> <data1>`
`<key2> <binarytoken2> <data2> </ark>`

This involved creating text transcriptions and corresponding acoustic features for each segment of the audio data. The .scp file provides a mapping between the keys in the archive file and their corresponding file positions. Each line contains a key followed by the offset (in bytes) of the corresponding entry in the .ark file.

Example: `<key1> <offset1>`

4.2. Acoustic Model

Post preprocessing, we implemented the GMM-HMM acoustic framework. The GMM-HMM model was trained using the extracted acoustic features. The model provided posterior probabilities over subword units (e.g., phonemes or context-dependent acoustic states) for each frame of audio. These probabilities were represented as a likelihood matrix.

4.3. Language Model: Deep Neural Network (DNN)

Following the GMM-HMM stage, we incorporate a DNN to refine and improve the predictions made by the GMM-HMM model.

The DNN model generated enhances posterior probabilities over subword units. This DNN-refined output provides more accurate representations of the spoken audio, building upon the GMM-HMM predictions.

We use a neural LM – a custom transformer trained on the same Librispeech dataset for the rescoring task. We then compose Lattice FSTs with Language Model FST. The FST created using Kaldi's tools represents the language model, lexicon, and any other components of the speech recognition system. We then compile the HCLG FST (Hidden Markov Model, Context, Lexicon, Grammar) using the trained acoustic and language models, lexicon, and other necessary components.

These word-level alignments are then converted into Finite State Transducers (FSTs). These FSTs provide a structured representation that allow for efficient manipulation of word-level alignments. To this end, we generate four types of lattices:
Type 1: DNN based lattice (with phone alignments followed by word alignments).
Type 2: GMM based lattice (with phone alignments followed by word alignments).
Type 3: DNN-based lattice (with direct word alignments).
Type 4: GMM-based lattice (with direct word alignments).

4.4. Transformer Model For Lattice Re-scoring

We trained a custom six-layer transformer model with 512 hidden embedding on an NVIDIA h100 GPU using the LibriSpeech[17] dataset. We trained the model over 4 epochs with a learning rate of 0.1 and a dropout rate of 0.1.

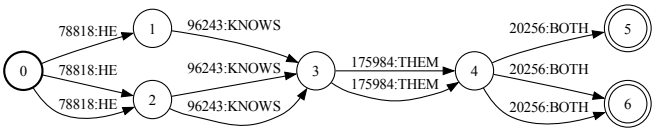

Fig. 3. Example of pre-rescoring lattice given an input utterance.

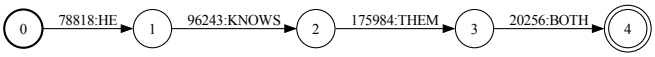

Fig. 4. Example of post-rescoring lattice given an input utterance.

5. RESULTS AND DISCUSSION

We conducted experiments on the four types of lattices mentioned in Section 4.3 and observed that their performance in the pipeline was identical. Hence we decided to make a comprehensive analysis on Lattice Type 1 which is representative of the other types. The below tables represent results for Lattice Type 1 on the test sets `'test-clean'` and `'test-other'`.

LM Scale	WIP 0		WIP 0.5		WIP 1	
	(a)	(b)	(a)	(b)	(a)	(b)
7	6.67	17.35	6.65	17.32	6.67	17.41
10	6.75	17.67	6.76	17.68	6.79	17.72
13	7.08	18.35	7.11	18.39	7.17	18.43

Table 1. WERs of Lattice Type 1 post-rescoring with three levels of word insertion penalties (WIPs) on both test datasets clean (a) and other (b).

From the results, we see a significant improvement in the WER post rescoring. Considering the LM Scale-WIP combination of 7 and 0.5 respectively, we see a decrease of 14.88% in the WER. This implies that our system performs slightly better than the SoTA of the HMM-(SAT) GMM model which has a WER of 8.01% on the *test-clean* dataset and also beats the WER of 22.49% on the `test-other` dataset.

In our results and discussion, it is evident that our rescoring architecture has played a pivotal role in achieving improved performance.Our rescorer, based on the Transformer architecture, proved to be a crucial component in improving Word Error Rate (WER). By leveraging the Transformer's attention mechanism, our model effectively captured long-range dependencies and contextual information within the input data. This enabled more accurate and contextually relevant predictions during the rescoring process, resulting in a notable reduction in WER. The model's ability to consider broader linguistic context played a pivotal role in achieving these enhanced ASR results.

LM Scale	WIP 0		WIP 0.5		WIP 1	
	(a)	(b)	(a)	(b)	(a)	(b)
7	7.56	21.08	7.64	21.03	7.67	20.07
10	8.16	21.58	8.21	21.59	8.31	21.61
13	9.08	23.04	9.19	23.12	9.34	23.16

Table 2. WERs of Lattice Type 1 pre-rescoring (raw lattice) with three levels of word insertion penalties (WIPs) on both test datasets clean (a) and other (b).

LM Scale = 7 WIP = 0.5				
Test Set	Utterances (#)	WER (%)	Rescored WER (%)	Change (%)
`test-clean`	2620	7.64	6.65	**-14.88**
`test-other`	2939	21.03	17.32	**-21.42**

Table 3. WER on LibriSpeech for Kaldi ASR baseline.

Technique	
Models with similar architecture	WER
AmNet [18]	8.60
HMM-SAT-GMM [19]	7.19
Snips [20]	6.40
Models with different architecture	WER
Deepspeech2 [21]	5.83
CTC + policy learning [22]	5.42
Li-GRU [23]	6.20
Ours	6.65

Table 4. A comparative study between our framework and the current SoTA for Different ASR Models.

We conduct an identical experiment on the `test-other` dataset. The Word Error Rate (WER) values were less favorable compared to the current dataset. In Table 3, we observed a 21.42% decrease in WER, but it's important to note that the initial values on the `test-other` set were already higher and our results are better than the SOTA [19].

6. CONCLUSION

In summary, our paper introduced a contextual ASR system with semantic lattice rescoring, achieving a substantial 14% reduction in Word Error Rate (WER) on librispeech test set post rescoring. We also explored different lattice types. Future work includes expanding to music domains, improving contextual cues, and optimizing beam length and DNN hyperparameters for enhanced performance.

7. REFERENCES

[1] Zhehuai Chen, Mahaveer Jain, Yongqiang Wang, et al., "Joint grapheme and phoneme embeddings for contextual end-to-end asr.," in *Interspeech*, 2019.

[2] Leonid Velikovich, Ian Williams, Justin Scheiner, Petar Aleksic, Pedro Moreno, and Michael Riley, "Semantic lattice processing in contextual automatic speech recognition for google assistant," 09 2018, pp. 2222–2226.

[3] Shankar Kumar, Michael Nirschl, Daniel Holtmann-Rice, Hank Liao, Ananda Theertha Suresh, and Felix Yu, "Lattice rescoring strategies for long short term memory language models in speech recognition," 2017.

[4] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[5] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv:1211.3711*, 2012.

[6] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[7] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *ICASSP*, 2018.

[8] Petar Aleksic, Mohammadreza Ghodsi, Assaf Michaely, et al., "Bringing contextual information to google speech recognition," 2015.

[9] Ding Zhao, Tara N Sainath, David Rybach, et al., "Shallow-fusion end-to-end contextual biasing.," in *Interspeech*, 2019.

[10] Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," in *SLT*, 2018.

[11] Antoine Bruguier, Rohit Prabhavalkar, Golan Pundak, and Tara N Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *ICASSP*, 2019.

[12] Duc Le, Gil Keren, Julian Chan, et al., "Deep shallow fusion for rnn-t personalization," in *SLT*, 2021.

[13] Duc Le, Mahaveer Jain, Gil Keren, et al., "Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion," *arXiv:2104.02194*, 2021.

[14] Keith Hall, Eunjoon Cho, Cyril Allauzen, et al., "Composition-based on-the-fly rescoring for salient n-gram biasing," 2015.

[15] Feng-Ju Chang, Jing Liu, Martin Radfar, et al., "Context-aware transformer transducer for speech recognition," in *ASRU*, 2021.

[16] Minglun Han, Linhao Dong, Zhenlin Liang, et al., "Improving end-to-end contextual speech recognition with fine-grained contextual knowledge selection," in *ICASSP*, 2022.

[17] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *ICASSP*, 2015.

[18] Jonathan Macoskey, Grant P. Strimel, Jinru Su, and Ariya Rastrow, "Amortized neural networks for low-latency speech recognition," 2021.

[19] Kaldi Speech Recognition Toolkit, "Kaldi speech recognition toolkit," `https://github.com/kaldi-asr/kaldi`.

[20] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," 2018.

[21] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew 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, "Deep speech 2: End-to-end speech recognition in english and mandarin," 2015.

[22] Yingbo Zhou, Caiming Xiong, and Richard Socher, "Improving end-to-end speech recognition with policy learning," 2017.

[23] Mirco Ravanelli, Titouan Parcollet, and Yoshua Bengio, "The pytorch-kaldi speech recognition toolkit," 2019.