CONTEXTUAL RNN-T FOR OPEN DOMAIN ASR

Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, Yatharth Saraf

Facebook AI, USA

{jinmahaveer,gilkeren,jaym,gzwieg,fmetze,ysaraf}@fb.com

1. Introduction

Present day ASR models using Deep Neural Networks (DNN) can be broadly classified into two frameworks: hybrid [1] and E2E [2, 3, 4]. A typical hybrid HMM-DNN system consists of three components trained individually: an acoustic model (AM) that estimates the posterior probabilities of Hidden Markov Model (HMM) states, a language model (LM) that estimates probabilities of word sequences, and a pronunciation model (PM) to map phonemes to words. These models are optimized independently [5] and then combined together using a Weighted Finite State Transducer (WFST) [6] for efficient decoding. In an E2E speech recognition model such as the RNN-T [2], a single neural network learns to map audio to text instead of using the distinct components of the hybrid systems. While this generally simplifies overall training and inference pipelines for ASR, E2E models tend to have difficulties with correctly recognizing rare words that are not frequently seen during training, such as entity names. In this paper, we propose modifications to the RNN-T model that allow the model to utilize additional text data with the objective of improving performance on these named entity words. We evaluate our approach on an in-house dataset sampled from de-identified public social media videos, which represent an open domain ASR task. By using an attention model to leverage the contextual metadata that accompanies a video, we observe a relative improvement of about 16% in Word Error Rate on Named Entities (WER-NE) for videos with related metadata.

Index Terms: RNN-T, Deep Contextualization, Context biasing, E2E ASR.

2. Prior Work

Prior work has leveraged contextual words either by on-the-fly (OTF) rescoring [10, 11, 12] or as an additional input to the DNN along with the audio. The first approach is generally referred to as Shallow Fusion whereas the latter as Deep Contextualization [7]. Our work falls in the latter category. It is most closely related to Contextual Listen, Attend And Spell (CLAS) [7], which also used context words from unpaired text to bias an E2E ASR model. The CLAS model was originally evaluated for closed domain ASR tasks like those used for virtual assistants by using entities such as contact names as context words. Further improvements to CLAS were done in [9] and [8] by using representations that leverage phonetic information as well. In this work, different from CLAS, we look at Deep Contextualization in the setting of an RNN-T ASR model, and evaluate our method on an open domain video ASR task using noisy text metadata from videos as context. In a closed domain use case such as making calls through an assistant, there is strong prior information about where entity names can appear in the utterance, whereas in our case the context words may appear anywhere in the conversational speech of the video. Deep con-
3. RNN Transducer

The framework of RNN-T ASR system is illustrated in Fig. 1. RNN-T for ASR has three main components: Audio Encoder, Text Predictor and Joiner.

The Audio Encoder uses audio frame at $x_t$ to produce audio embedding $h^{enc}_t$ (Equation 1). The Audio Encoder used in this work is a stack of bi-directional LSTM (BLSTM) layers.

$$h^{enc}_t = f^{enc}(x_t)$$  \hfill (1)

The Text Predictor uses the last non-blank target unit $y_{u-1}$ to produce embedding $h^{pred}_u$ (Equation 2). The Text Predictor is a stack of LSTM layers in this work. We use sentence pieces as target units.

$$h^{pred}_u = f^{pred}(y_{u-1})$$  \hfill (2)

The Joiner takes in the output of Audio Encoder and Text Predictor and combines them to produce an embedding $z^{t,u}$:

$$z^{t,u} = \phi(U h^{enc}_t + V h^{pred}_u + b)$$  \hfill (3)

$U$ and $V$ are matrices that are used to project audio and text embeddings to the same dimensions. $\phi$ is a non-linear function such as Relu [15] or tanh.

Finally, the joiner’s output, $z^{t,u}$, is passed through a linear transformation followed by a softmax layer to produce a probability distribution over target units (y), i.e. sentence pieces plus a special blank symbol:

$$h^{t,u} = W z^{t,u} + b$$  \hfill (4a)

$$p(y(t,u)) = \text{softmax}(h^{t,u})$$  \hfill (4b)

By incorporating both audio and text for producing $p(y(t,u))$ (Equation 4b), RNN-T can overcome the conditional independence assumption of CTC models [16]. The emission of blank as output unit results in an update of the audio embedding by moving ahead in time axis $t$ whereas emission of non blank results in a change in the text embedding. This results in various possible alignment paths as shown in the lattice of size $T \times U$ in Figure 1 of [2]. The sum of probabilities of these paths gives the probability of an output sequence, $Y$, given the input sequence, $X$, where $Y$ is the sequence of non blank output target units and $X$ is the input sequence of audio frames.

4. Contextual RNN-T

We modify the base RNN-T model described in Section 3 and add three additional components: an Embedding Extractor (EE), an Attention Module (AttModule) and (optionally) a Biasing Module (BiasingModule) as shown in Figure 2.

As in [7], each context word, $w_i$, is first represented as a sequence of target sentence piece units, e.g. the word “Jarred” may be mapped to $\{Ja, r, re, d\}$. This sequence is then fed to an BLSTM, and the last state of the BLSTM is used as the embedding of the given context word (shown as $h^{EE}_u$ in Figure 2).

In the vanilla RNN-T system described in Section 3, probabilities over target units $p(y(t,u))$ (Equation 4b)) are conditionally dependent on the outputs of the Audio Encoder, $h^{enc}_u$, and Text Predictor, $h^{pred}_u$. In contextual RNN-T, we would like to make $p(y(t,u))$ conditionally dependent on contextual metadata words as well. This dependency can be achieved by incorporating the context word information into any of the Audio Encoder, Text Predictor and Joiner components. In this work, we explore incorporating the context word information into the Text Predictor and Joiner of the RNN-T.

An Attention Module (AttModule) is used to compute attention for each word in the metadata text. AttModule uses the predictor output for non-blank text history up to $u$ ($h^{pred}_u$) and word embedding, $h^{EE}_u$, to compute attention weight, $e_{u,i}$, as shown in Equation (5b). We use location-aware attention that takes into account the attention weights from the previous predictor state, $\alpha_{u-1}$, while computing alignments at the current step [4].

$$F = Q \ast \alpha_{u-1}$$  \hfill (5a)

$$e_{u,i} = w^T \tanh(A h^{pred}_u + B h^{EE}_u + C f_i + b)$$  \hfill (5b)
\[ \alpha_{u,i} = \exp(e_{u,i}) / \sum_{j=1}^{N} \exp(e_{u,j}) \]  
\[ F \text{ is output of convolution of } \alpha_{u-1} \text{ with matrix } Q \text{ (Equation (5a)). } f_i \text{ in Equation (5b) is used to denote the output for the } i \text{th word in } F. A, B, C \text{ are matrices, } b \text{ and } w \text{ are vectors.} \]

We next compute a context vector, \( c_u \), as shown in Equation (10).

\[ c_u = \sum_{i=1}^{N} (\alpha_{u,i} \ast h_i^{EE}) \]

We concat this context vector, \( c_u \), to the output of the Text Predictor (\( h_u^{pred} \)). This results in a modification of the Joiner equation (3) to equation (7):

\[ z^{Lu} = \phi(Uh_i^{enc} + V[c_u; h_u^{pred}] + b) \]

An additional biasing module (BiasingModule) may be used to find an active subset of the context words that have the same prefix as the last unfinished word in the text history (\( Y_{u-1} \)) to create an additional vector that biases the Joiner towards selecting from the subset of sentence pieces that correspond to the active context words using the attention values \( \alpha_{u,i} \) from Equation (5c). For example, if the decoded form of \( Y_{u-1} \) is "Africa An" and the list of the context word is "Android, Antenna" and Ptyorch then the active context words are "Android" and "Antenna". The BiasingModule computes a biasing value, \( \text{bias}_{u,i,k} \), for each sentence piece \( k \) of word \( w_i \) at a given text history \( Y_{u-1} \) as shown in Equation (8). \( ISP(w_i, Y_{u-1}, k) \) returns 1 if \( w_i \) is active word at \( Y_{u-1} \) and \( k \) is the sentence piece in \( w_i \) followed by the shared prefix, otherwise it returns 0. We then compute \( \text{bias}_{u,k} \) for each sentence piece \( k \) at \( Y_{u-1} \) by summing over all context words as in Equation (9). A vector of all biases, \( b_{u,k} \), with length equal to number of sentence pieces is then linearly projected and passed through an optional Dropout layer before feeding to the Joiner as shown in Equation (10).

\[ \text{bias}_{u,i,k} = \alpha_{u,i} \ast ISP(w_i, Y_{u-1}, k) \]

\[ \text{bias}_{u,k} = \sum_{i=1}^{N} \text{bias}_{u,i,k} \]

\[ z^{Lu} = \phi(Uh_i^{enc} + V[c_u; h_u^{pred}] + \text{Dropout}(Bb_u) + b) \]

Equations 4a and 4b remain unchanged.

5. Experiments

5.1. Dataset

The dataset used for our experiments was sampled from English videos shared publicly on Facebook. The data is de-identified by removing information such as the user who posted the video, and only use the content of the video and the text metadata for training and evaluation.

We segment the data in duration of 10 seconds for training and evaluation. Metadata words are obtained from the title and description text of the video, if available, after doing simple text cleaning and filtering such as removing words with hyperlinks in them. If a word in the text is capitalized then we also add its lower case version as metadata word. We do not preserve the ordering of words in metadata both in training and evaluations. Each segment of the same video shares the same list of contextual words. We use about 8k hours of data for training and about 170 hours for evaluation. We further divide the evaluation test set based on whether there is any word in the reference for that segment that also appears in the metadata words for that video. The segments for which there is at least one common word are referred to below as the CommonNonZero set and the remainder as the CommonZero set. We present results on these two evaluation sets.

5.2. Model

The architecture of the Contextual RNN-T model from Section 4 used for the experiments in this paper is as follows. The Audio Encoder is a 4-layer BLSTM with 604 dimensions. We use subsampling of 2 across the time dimension after the first and second BLSTM layers. The output of the last layer of the BLSTM is projected to 1024 dimensions. The Text Predictor is a 2-layer LSTM of 256 dimensions whose output is also projected to 1024 dimensions. We used a token set consisting of 200 sentence pieces, trained using the sentence piece library [17]. The Embedding Extractor is a 2-layer BLSTM of size 100. The Attention Module has the following parameters: 1) Convolution (Q) with 2 out channels and kernel of size 1, 2) Attention is computed over 64 dimensions with A being of size 1024 \( \times 64 \), B of size 200 \( \times 64 \) and C of size 2 \( \times 64 \) (in Equations (5a), (5b) and (5c)).

As seen in Table 2, the Contextual RNN-T (Equation (7)) model (row 2) improves on WER-NE by about 13% relative compared to the baseline model (row 1) on the CommonNonZero evaluation set. Introducing BiasingModule (as in Equation (10)) improves WER-NE further by another 3%. As shown in Table 3, both WER and WER-NE for the CommonZero test set does not get significantly impacted by the Contextual RNN-T models when there is no intersection between the metadata words and the reference.

We also measure robustness of our system using precision and recall of the emission of context words in the model's hypotheses. A True Positive occurs when a context word from the metadata of the video is correctly output by the model as compared to the reference. A False Positive occurs if the model outputs a context word but it does not appear in the reference. We show aggregated precision and recall over both test sets for
triggering of the context words in Table 4. We see an improvement in recall of 8.3% and degradation in precision by 1.3% relative for the best contextual model compared to the baseline.

Table 1: Comparing outputs generated by the baseline and Contextual RNN-T models. Named entities are represented with bold font in these examples.

<table>
<thead>
<tr>
<th>Reference Snippet</th>
<th>Baseline Output</th>
<th>Contextualization Output</th>
<th>Metadata Words (truncated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>from the <strong>Africa Android</strong> Challenge</td>
<td>from the <strong>Africa and red</strong> challenge</td>
<td>from the <strong>Africa Android</strong> challenge</td>
<td>innovative, School, language, Plus, app, tutorial, products, educational, startup</td>
</tr>
<tr>
<td>its very intuitive so when you look at <strong>PyTorch</strong> itself</td>
<td>its very intuitive so when you look at pie towards itself</td>
<td>its very intuitive so when you look at <strong>PyTorch</strong> itself</td>
<td>experiences, novel, PyTorch updates, Facebook, machine, AI, language, research, ...</td>
</tr>
</tbody>
</table>

Table 2: WER and WER-NE results on CommonNonZero test set

<table>
<thead>
<tr>
<th>Model</th>
<th>WER</th>
<th>WER-NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>16.04</td>
<td>24.69</td>
</tr>
<tr>
<td>Contextual RNN-T w/o BiasingModule</td>
<td>15.45</td>
<td>21.41</td>
</tr>
<tr>
<td>Contextual RNN-T with BiasingModule</td>
<td>15.37</td>
<td>20.66</td>
</tr>
</tbody>
</table>

Table 3: WER and WER-NE results on CommonZero test set

<table>
<thead>
<tr>
<th>Model</th>
<th>WER</th>
<th>WER-NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>23.07</td>
<td>29.18</td>
</tr>
<tr>
<td>Contextual RNN-T w/o BiasingModule</td>
<td>22.95</td>
<td>29.71</td>
</tr>
<tr>
<td>Contextual RNN-T with BiasingModule</td>
<td>22.89</td>
<td>29.93</td>
</tr>
</tbody>
</table>

Table 4: Precision and Recall for context words across both test sets

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.934</td>
<td>0.844</td>
</tr>
<tr>
<td>Contextual RNN-T w/o BiasingModule</td>
<td>0.920</td>
<td>0.898</td>
</tr>
<tr>
<td>Contextual RNN-T with BiasingModule</td>
<td>0.922</td>
<td>0.914</td>
</tr>
</tbody>
</table>

6. Analysis

To understand better what the Contextual RNN-T model is doing, we visualize attention values for a few test segments where it correctly recognizes named entities that the baseline model makes errors on. These examples are shown in Table 1.

For the example shown in row 1 of Table 1, both the contextual and baseline models are able to recognize common entities such as **Africa**. However, the baseline model has difficulties in recognizing entities that are not frequent in training data set, such as **Android** and **PyTorch**. Since **Android** appears in the metadata, the Contextual RNN-T model is able to attend to it and transcribe it correctly. This can be seen in the visualization of attention given to the words in the metadata at each output target unit (u) of the Contextual RNN-T Model in Figure 3.

7. Conclusion

We show that contextual metadata text, even if it is noisy, can be used to improve recognition of named entities for a challenging open domain ASR task such as social media videos within the framework of an E2E RNN-T ASR model. Some future explorations could be: i) Using contextual embeddings from other modalities such as images from video, ii) Using semantic embeddings to represent the metadata.
8. References


