CS230

Help me help you: E2E ASR use case-specific optimization with user-provided data

Yikun Chi and Jake Silberg
yikunchi@stanford.edu and jsilberg@stanford.edu

1 Introduction

Today's ASR systems often commit errors that any human with domain knowledge could correct in seconds, for example, transcribing "asymptotic" as "awesome topic". Such errors are particularly likely in newly-favored end-to-end (E2E) systems trained on paired speech and text data [Zheng et al., 2021]. Yet, end-users with such knowledge complain that they have few ways to participate in improving these systems, and often lack the specific data format (paired audio and text) to do so. We instead imagine a world where users have accessible tools to make feasible improvements to a transcription model customized to their needs. For example, a professor might provide a link to a textbook to improve a model's expertise. A religious leader live-streaming services might provide a link to her YouTube channel, so the model can pre-identify challenging utterances.

Towards this vision, we attempted to improve pre-trained E2E ASR models based on data users could feasibly provide, particularly unpaired text-only data that overlaps with the vocabulary to be transcribed, and unpaired audio data where the model can identify difficult-to-transcribe utterances and seek out additional training examples. To mimic real-world uses, we focus on highly specialized vocabularies: Supreme Court hearings, and university lectures.

2 Related work

The literature on improving E2e systems suggests that synthesizing text-only data using TTS is a particularly promising form of data augmentation. Microsoft researchers find that training on paired data generated from TTS was more effective than neural language models while eliminating the need for a second network [Li et al., 2020]. Researchers at ITMO University found that models trained on TTS data performed comparably to semi-supervised systems [Laptev et al., 2020]. We used these findings to focus on synthesizing speech we believe would be closely related to the test set vocabulary. Unfortunately, the Microsoft paper does not make clear what the source and target domains they used were, making it hard to understand exactly how applicable this method is for real-world use. Meanwhile the ITMO research applied the TTS to a subset of a widely used ASR dataset, LibriSpeech. We want to try this approach on real-world data that better approximate users' needs for a customized model.

Amazon researchers also used TTS to synthesize new paired examples from transcripts of the test set and find a 50% reduction in WER for out-of-vocabulary examples [Zheng et al., 2021]. Importantly, the Amazon team discusses that synthetic speech should be mixed with real speech to avoid too much acoustic domain shift, constituting a new hyperparameter, and suggest a ratio of 30% real audio. While a promising avenue, the challenge with their design is that real users would not have access to the target domain transcripts. We want to extend this to training on text data that would be available to an end user. Finally, a Google research paper focusing on the use case of voice assistants is probably closest to our approach, by training on synthesized pronunciations of names a user could provide, such as the contacts in their phone [Sim et al., 2019]. We extend this from names to named entities

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

and specialized vocabularies. We also term this type of related text data "hot text", inspired by "hot docs," a term used by lawyers for highly relevant documents to one's case.

For speaker-specific data, we limit the scope of feasible user input to unlabeled user audio and limited on-demand audio (i.e., user speaks phrases given by the algorithm). Active learning techniques such as [Malhotra et al., 2019] and [Riccardi and Hakkani-Tur, 2005] can identify desired labels from unlabeled examples. Inspired by some work around auto mining transcript data [Hakkani-Tur et al., 2004] and using text to speech to improve ASR systems [Zheng et al., 2021], we extend the active learning result by cross comparing the difficult utterances with a list of key words (either from keyword extraction on hot text, or from sources such as glossary) using phonetic distances (metaphone and dmetaphone). The hypothesis is that certain difficult utterance are the keywords, e.g., "derivative" in an AI lecture. So even though the transcription is incorrect, its phonetic distance would be close to the actual keyword.

The base of our work is building on NVIDIA's Nemo models, which we use for both TTS and ASR [Kuchaiev et al., 2019]. These are convolution models, using the Connectionist Temporal Classification (CTC) loss function pioneered by DeepSpeech Amodei et al. [2015]. This design is covered further in the "Methods" section. Additionally, we are only the second paper to look at ASR using Supreme Court speech, and build on the code base of the first. The previous paper focused just on identifying who was speaking, while we focus on transcription Tumminia et al. [2021].

3 Dataset and Features

We have primarily used Project Oyez Supreme Court and MIT Open CourseWare data.

For Project Oyez, we used NeMo's CTC segmentation to split Supreme Court opinion announcements into short (<16 seconds) clips with paired transcripts, which were converted from mp3 files to wav files with a sample rate of 16000. These opinion announcements typically consist of a single Justice reading from their opinion. For example, clips include, "This case comes on writ of certiorari to the United States Court of Appeals for the Seventh Circuit," and, "Justice Kagan has filed a dissent in which Justices Ginsburg and Sotomayor have joined." It seems highly likely that both of these contain words the pretrained model probably has not seen before! We considered 2000-2014 opinion announcements to be the universe of training data, giving us up to 52 hours and more than 24,000 labeled training examples. We used 2015 and 2016 announcements (6 hours, 3000 examples) as the dev set, and 2017 and 2018 opinion announcements (6 hours, 3000 examples) as the evaluation set. We then synthesized 44,000 additional paired examples (60 hours) from the "facts of the case" text provided by Project Oyez's API.

For hot-speech implementation, we use one hour of MIT Artificial Intelligence course lecture 12B audio to identify the difficult to transcribe utterance, and we extracted some keyword pronunciations from other lectures to mimic additional on-demand user input. We use the lecture supplemental notes as the hot text. The MIT Opencourse provides transcripts to AI lectures.

4 Methods

For our ASR models, we began with NVIDIA's QuartzNet ASR model [Kriman et al., 2019]. QuartzNet uses blocks of 1D convolutions, with BatchNorm and a ReLU activation function. Our 15x5 pre-trained model has 79 total layers and around 20M parameters and uses a CTC loss function. In E2E CTC models, audio is converted to acoustic features by the encoder. Those features are then provided to the decoder which translates each discrete time period to a single letter or the "blank" token which indicates the start of a new sound. Of course, uttering a single letter may take multiple time periods, so the CTC model then collapses repeat letters not separated by the blank token. For example, the decoder may generate "g-g-g_i-i_n_s-s_b_u_r-r_g" which the CTC model would collapse to "ginsburg." As we did not alter NeMo's TTS neural model (to simulate users lacking paired training data), nor the built-in data augmentation algorithm SpecAugment [Park et al., 2019], these are described in the Appendix.

4.1 Hot Text Methodology

Our theory of improvement for "hot text" is that, while a user will not have the literal transcripts of the test set to use for TTS as in [Zheng et al., 2021], they have text-only data with overlap with the test set. We want to see (1) Can fine-tuning with TTS from "hot text" improve performance over a pre-trained model (2) Are there new error analysis metrics that can help us understand how a "hot text" model is performing and improve it? For (1) our key comparison is to compare the pre-trained model against a model that uses either only TTS data or TTS with some paired speech (as suggested by [Zheng et al., 2021]), and finally against the model with 15 years of paired data – the gold standard of fine-tuning.

For (2) to quantify how this form of data augmentation performs, we propose specific metrics to aid with error analysis and reducing variance. First, drawing on a similar concept in image recognition we propose Intersection over Union (IOU):

$$\frac{\text{Set of "hot text" vocab} \cap \text{Set of target domain vocab}}{\text{Set of "hot text" vocab} \cup \text{Set of target domain vocab}}$$

This penalizes "hot text" datasets with extraneous text. An alternative could be Percentage of Target Vocab (POTV), where the denominator is just the set of target domain vocab. If increasing POTV is more correlated with performance, then adding more TTS text data should help. If IOU is more correlated, then deleting extraneous text would help.

Finally, for Error Analysis, we propose a modification to the Word Error Rate. The typical WER equation is:

$$\frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{\text{Number of words in reference text}}$$

We propose evaluating WER separately on examples where the "hot text" algorithm saw every word in the TTS data – the Hot Text WER – and examples with words it did not see – the Cold Text WER. Like Professor Ng's heuristic for identifying if the model or the beam width is the problem in a language system, these can help diagnose problems. If the model underwhelms on WER, but does well on Hot Text WER, the "hot text" didn't overlap enough with the test set. If the model underperforms even on Hot Text WER, the problem is the TTS or ASR models. A high Cold Text WER could indicate the model is overfitting on synthetic voices, and users should increase the mix of real data.

4.2 Hot Speech Methodology

Overall, our goal is to identify difficult word utterances from the speaker by analyzing additional unlabeled audio from the speaker (aka "hot speech"). By cross referencing with content specific keywords, we can build an additional training dataset to help the model better transcribe the key words.

We first identified difficult-to-transcribe word utterances. We feed the unlabeled audio clip into the model, and split the output audio based on the CTC matrix output's prediction of spaces. We also split the output transcriptions by spaces. This way we can obtain the word level audio clips, and its transcription (often incorrect). We use BeamSearch path probability [Malhotra et al., 2019] as an indicator to identify words that the model finds it difficult to transcribe. NeMo model does not perform well when predicting a single word's utterance. So we build series of four word sub clips, and feed the sub clip transcription CTC matrix into BeamSearch to examine the top 1000 candidates as well as the max path log probability. For example, if one input segment's transcription is the sentence "Today we will discuss neural networks", then first we obtained the utterance of each word, and build the following clips: "Today we will discuss", "we will discuss neural", "will discuss neural networks". We examine the prediction confidence of each subclip by looking at the path probability. For the low confident clips, we split the transcription into 4 words, and add each transcription to the unconfident word set.

We then establish a list of domain specific keywords, this can be done through statistics based keyword extraction process, e.g. YAKE, or through alternative sources such as textbook glossary (education setting), legal terms (law setting) and etc.

Given the unconfident word set and the keyword set, we cross-compare every pair of words in those set, and if their phonetic distance (based on metaphone and dmetaphone) is highly similar, we keep both words (i.e.: we think that the true label for the mis-transcribed could be the keyword).

At the end, we generate a training dataset using NeMo TTS and data perturbation on 1) the specific keyword, 2) the keyword and its mis-transcribed results, and 3) all keywords and all mis-transcribed results. We also simulate user providing additional speech examples of certain keywords by extracting search word's pronunciation from hold-out hidden dataset.

5 Experiments/Results/Discussion

5.1 Hot Text Results

For "hot text" hyperparameters, increasing the number of epochs from 5 to 10 benefited paired models more than "hot text" models, and a very low learning rate (.0001) performed best. We hoped weight decay might reduce overfitting on synthetic audio, but it rarely improved performance on the dev set. Freezing the encoder also degraded performance. A table of model performance is in the Appendix.

The most important hyperparameter was the mixture of synthetic to real data. Using 17 years of synthetic data significantly underperformed the pre-trained model, and even fine-tuning on just "hot text" overlapping our dev set underperformed. Only models with 30% real data (e.g., 2 years of TTS and 1 year of real opinion announcements) or more outperformed the pre-trained model.

Hot Text key results: Experimental configuration Control configuration

Model	Dev set WER	Test set WER	Hot Text WER (Dev)
Pre-trained	0.1236	0.1027	0.1134
2 years of TTS	0.1419	N/A	0.1223
2 years of TTS + 1 year paired	**0.0939 (-1.5%)**	**0.0801 (+2.5%)**	**0.072 (-10%)**
1 year paired	0.0953	0.0783	0.0804
15 years paired ("gold standard")	0.0749	0.0661	0.0644

Note: 2 years TTS + 1 year paired are two different models for dev and test sets, as dev is trained on 2015-2016 "hot text" and test on 2017-2018 "hot text", though both share the 2014 paired real audio

The best performing "hot text" models are above. On our dev set, a model trained on 2 years of "hot text" and 1 year of paired real audio outperformed both the pre-trained model and a "control" model fine-tuned on just 1 year of paired audio. Most interestingly, when we evaluated the models on just the Hot Text WER, the hot text model performs 10% better than the control model. This suggests that with a better overlap of "hot text" and the target domain, we could see significant gains.

We also looked for patterns in the transcriptions models generated. The combination of TTS and paired data was critical for names. The pretrained model transcribed "Justice Ginsburg" as "justis kensberg." The model trained just on 2 years of "hot text" wrote "justice ginsburg" but also "justice sodmayor" instead of "sotomayor". The model with 2 years of "hot text" and 1 year of paired data correctly transcribed both. For legal terminology the TTS model fared better. While the pre-trained model transcribes "federal lor", both the TTS and TTS+paired models correctly transcribe "federal law." It seems likely the pre-trained model had seen "law" in training, but fine-tuning successfully upweighted it. This suggests that "hot text" may work best for improving performance of less frequent terms, but may not be as promising for names.

On the other hand, our proposed metrics to predict how "hot" given text is did not predict performance. Details are in the appendix.

5.2 Hot Speech Results

5.2.1 Difficult Word Prediction and Label Recovery

Our active learning application and cross reference difficult utterance transcription with keywords yield promising results. We were able to identify words that the model will have a high chance of mis-transcribing, and were able to successfully recover the true label of many difficult to transcribe utterance. Some of the pairs identified include:

- "aneral", "annual", "neral" -> "neural"
- "terevitive" -> "derivative", "performes" -> "performance"
- "norin" -> "neuron" , "wet" -> "net"

In above examples, each word on the left represents one or more utterance instances where the pre-trained model is mis-transcribing, and the word on the right is the recovered true label.

5.2.2 Impact of On-Demand User Input

User provided input on keywords pronunciation dramatically decreased the WER on model's performance on those keywords. One example showcased in the code is the word "neural net". Our active learning component successfully predicted that "neural net" will have a high error rate, which turns out to be 95%. Training on 5 additional unseen instances of user saying "neural net" (simulating on-demand user input), the WER dropped to 45% (learning rate = 0.001, trained for 10 epochs). The WER on overall lecture (1 hour of lecture video) increased from 20% to 25%.

5.2.3 Impact of synthetic input

The impact of synthetic training dataset is less than ideal. First we tested the effectiveness of using TTS generated specific keywords. We generated 6 clips of the word "neural" with varying speed perturbation. However, after training, the model has a WER of 1 when predicting the utterance of "neural". We did notice as we increased the number of epochs from 10, to 20, 50, 100, and eventually to 500, the output of the post training neural net is close to the truth (e.g.: output "eural" instead of "neural"). So we hypothesized that additional training data and epochs could achieve better result. We also experimented with training data that includes a keyword and its mis-transcribed results (e.g.: "aneral", "annual", "neral" , "neural") and yielded similar results. Finally, training the model on all TTS generated keywords and their mis-transcribed results increased the lecture level WER from 20% to 57%. All training results are listed in the last section of the code notebook.

6 Conclusion/Future Work

For "hot text", we agree with the literature on the need for a mix of real and synthetic data, and find our model outperformed the "control" best when the "hot text" contained all the vocabulary in an example. Perhaps adding better overlapping text can improve performance further, potentially in combination with more synthetic speakers or other ways to limit the audio shift of TTS.

For "hot speech", we were able to successfully identify difficult to transcribe segments, and recover its true label. But the training result of using synthetically generated training data is not ideal. One immediate next step that could be promising is instead of training on synthetically generated training data, we could train on the difficult to transcribe segments with recovered labels. This process is essentially the active learning training loop, but replacing the human transcribing process with an unsupervised guessed label through phonetic distance cross comparing with keyword set. In the synthetic training data direction, more exploration should be done in terms of the number of samples, sample length, sample ratio of synthetic data and user on-demand data to produce better performance.

Finally, we believe the legal speech and MIT lectures datasets are interesting testing areas for improving ASR on technical domains with unusual vocabulary, and hope our making this data available in a format for ASR will be of use to other researchers.

7 Contributions

Jake Silberg led the work on "hot text" and the Oyez dataset. Yikun Chi led the work on "hot speech" and the MIT dataset. We also thank the NeMo team, who provided guidance on several bug reports and questions via Github Issues threads.

8 Our code

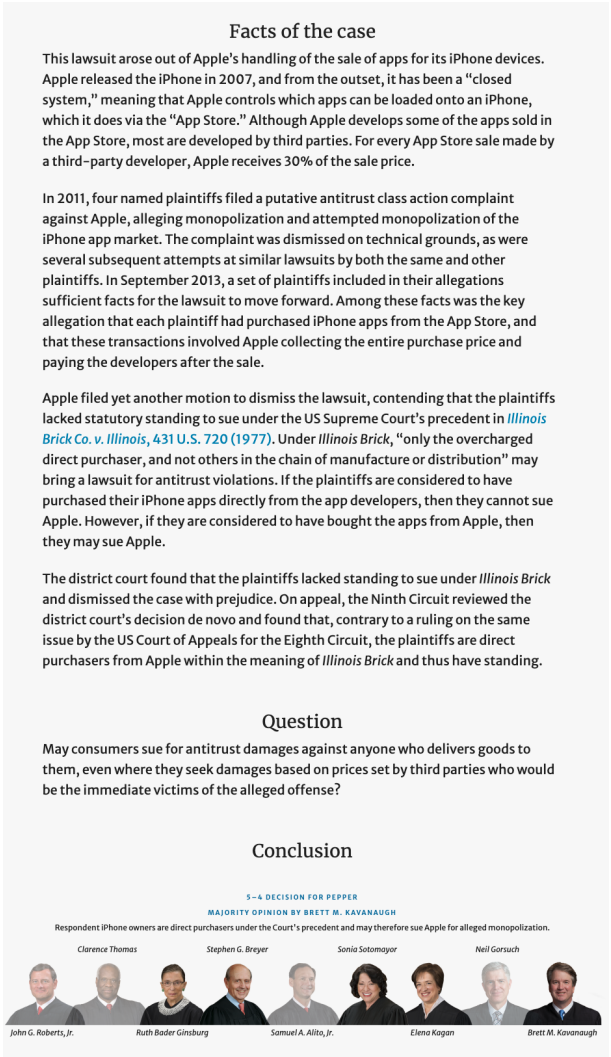
https://drive.google.com/drive/folders/1skmTwQROCDzx9VcjQWaKob4YUBEzKL-4?usp=sharing

9 Appendix

This appendix further describes the data we used for hot text, the TTS models we used, SpecAugment, as well as more details on our hyperparameter tuning and results.

9.1 Hot Text Data

An example of the "facts of the case" used for generating speech is below. It should help provide many of the names and terms that will later appear in the transcript of the opinion announcement.



Facts of the case

This lawsuit arose out of Apple's handling of the sale of apps for its iPhone devices. Apple released the iPhone in 2007, and from the outset, it has been a "closed system," meaning that Apple controls which apps can be loaded onto an iPhone, which it does via the "App Store." Although Apple develops some of the apps sold in the App Store, most are developed by third parties. For every App Store sale made by a third-party developer, Apple receives 30% of the sale price.

In 2011, four named plaintiffs filed a putative antitrust class action complaint against Apple, alleging monopolization and attempted monopolization of the iPhone app market. The complaint was dismissed on technical grounds, as were several subsequent attempts at similar lawsuits by both the same and other plaintiffs. In September 2013, a set of plaintiffs included in their allegations sufficient facts for the lawsuit to move forward. Among these facts was the key allegation that each plaintiff had purchased iPhone apps from the App Store, and that these transactions involved Apple collecting the entire purchase price and paying the developers after the sale.

Apple filed yet another motion to dismiss the lawsuit, contending that the plaintiffs lacked statutory standing to sue under the US Supreme Court's precedent in *Illinois Brick Co. v. Illinois*, 431 U.S. 720 (1977). Under *Illinois Brick*, "only the overcharged direct purchaser, and not others in the chain of manufacture or distribution" may bring a lawsuit for antitrust violations. If the plaintiffs are considered to have purchased their iPhone apps directly from the app developers, then they cannot sue Apple. However, if they are considered to have bought the apps from Apple, then they may sue Apple.

The district court found that the plaintiffs lacked standing to sue under *Illinois Brick* and dismissed the case with prejudice. On appeal, the Ninth Circuit reviewed the district court's decision de novo and found that, contrary to a ruling on the same issue by the US Court of Appeals for the Eighth Circuit, the plaintiffs are direct purchasers from Apple within the meaning of *Illinois Brick* and thus have standing.

Question

May consumers sue for antitrust damages against anyone who delivers goods to them, even where they seek damages based on prices set by third parties who would be the immediate victims of the alleged offense?

Conclusion

5–4 DECISION FOR PEPPER

MAJORITY OPINION BY BRETT M. KAVANAUGH

Respondent iPhone owners are direct purchasers under the Court's precedent and may therefore sue Apple for alleged monopolization.

Clarence Thomas *Stephen G. Breyer* *Sonia Sotomayor* *Neil Gorsuch*

John G. Roberts, Jr. *Ruth Bader Ginsburg* *Samuel A. Alito, Jr.* *Elena Kagan* *Brett M. Kavanaugh*

9.2 TTS Model

For generating text-to-speech for "hot text" we used NVIDIA's Tacotron2 as a spectrogram generator and WaveGlow as the audio generator. Tacotron2 is a recurrent sequence-to-sequence model with Long Short Term Memory (LSTM). It consists of 3 convolutional layers, followed by bidirectional LSTM, 2 further LSTM layers, and 5 more convolution layers to generate Mel Spectrograms (spectrogram images modified to better mimic how humans hear noise). WaveGlow is a generative model that learns the mapping from a dataset distribution into a Gaussian distribution. To then generate audio, samples are generated from the Gaussian distribution. The default configuration includes 512 residual channels. The hot speech TTS uses NVIDIA's FastPitch Hifigan E2E model. This model is chosen due to its capability of producing out of vocabulary pronunciation. We did not train or modify either model, but further details can be found at the link in "Code citations."

9.3 SpecAugment

NeMo ASR default models use the SpecAugment data augmentation method developed by [Park et al., 2019]. This algorithm warps features, masks blocks of frequency channels, and masks time steps to create perturbed features. For "hot text" we used default SpecAugment settings.

9.4 Hyperparameter tuning for hot text

We describe in more detail our full process for choosing hyperparameters other than Data Mix, which is fully described in the main body given its importance:

- Number of epochs: We generally found that TTS-only models tended to move away from the target domain, so 5 epochs outperformed 10. For TTS+paired models, 10 outperformed 5 but with minimal gains. For paired models, we saw noticeable improvement between 5 to 10. In several cases, the TTS+paired model would outperform after 5 epochs, but the model with only paired data would outperform after 10 epochs.
- Dropout: The pre-trained model had 0 dropout. We briefly experimented with dropout, but found it significantly increased both bias and variance.
- Learning rate: We initially managed to get some "fine-tuned" models with over 100% WER by using the .01 learning rate that NVIDIA suggests for training from scratch, which was concerning! While NVIDIA suggests .001 for fine-tuning, given how well the pre-trained model was already performing, we found .0001 to work best.
- Weight decay: Weight decay, which some consider identical to L2 regularization in that it penalizes larger parameters so it shrinks weights towards 0 but does not set them exactly to 0 (as L1 regularization would), seemed a promising avenue because all the TTS models had significantly better performance on the training set than the test set. While NVIDIA recommends weight decay of around .001, we tried everything from .0005 to .01, and nearly always ended up degrading performance (only one configuration performed better at .003 than .001). This is likely due to the fact that the training and dev sets were from different distributions, making regularization challenging – regularize too little and we would overfit to the synthetic audio voices, but regularize too much and we would underperform the carefully pre-trained model. This suggests Data Mix (adding real audio) is likely a more promising form of preventing overfitting to the TTS data.
- Freezing the encoder: Because an ASR system consists of an encoder that recognizes audio features and a decoder that transcribes it to text, we experimented with freezing the encoder, hoping that training this way would improve the vocabulary of the decoder without shifting the encoder towards the single TTS voice. Unfortunately, this also degraded performance.

The full combination of hyperparameter experimentation, including Data Mix, is below. Note that once an avenue did not seem promising, we did not continue to test it, so the table does not contain every combination of hyperparameters. For example, the learning rate of .01 immediately resulted in loss jumping from 30 to 300, so we did not complete training.

Hyperparameter experimentation (Bold models advanced to test set)

Data Mix	Epochs	LR	WD	Training WER	Dev WER
17 years TTS	5	.0001	0.001	.07	0.396
2 years TTS	5	.0001	.001	.0727	0.148
2 years TTS	10	.0001	.001	0.0541	0.158
2 years TTS	5	.0001	.003	.0702	0.142
2 years TTS + 1 year paired (33% real)	5	.0001	.001	0.0676	0.0979
2 years TTS + 1 year paired (33% real)	**10**	**.0001**	**.001**	**0.0524**	**0.0939**
2 years TTS + 1 year paired (33% real)	5	.0001	.005	0.0665	0.0980
2 years TTS + 1 year paired (33% real)	5	.0001	.009	N/A	0.0997
2 years TTS + 1 year paired (33% real)	5	.0001	.0005	N/A	0.0963
2 years TTS + 2 years paired (50% real)	5	.0001	.001	0.057	0.0888
2 years TTS + 2 years paired (50% real)	10	.0001	.001	0.046	0.0864
1 year paired	5	.0001	.001	0.0643	0.1052
1 year paired	**10**	**.0001**	**0.001**	**0.0366**	**0.0953**
2 years paired	5	.0001	0.001	0.0554	0.0904
2 years paired	10	.0001	0.001	0.03542	0.08623
15 years paired	**10**	**0.0001**	**0.001**	**0.0453**	**0.0749**
15 years paired	10	.0001	0.01	0.1490	0.1949

9.5 Metrics performance

Below is how our proposed IOW and POTV words compared to WER. The "hot text" for the test set had a lower IOU and POTV than the "hot text" for the development set, but our model performed better on the test set. Still, the fact that 17 years of TTS data had high POTV but low performance suggests a penalty for too much "hot text" is worthwhile. A measure of auditory distance of the TTS from the original speakers must also be important.

Hot Text Error Analysis Metrics: Dev Set had higher IOU and POTV than Test Set, but worse WER

Model	Intersection Over Union	Percentage of Target Vocabulary	WER
Test Set TTS + 1 year paired	0.4064	71.96%	**0.08**
Dev Set TTS + 1 year paired	**0.4244**	**72.41%**	**0.094**
17 years TTS	0.242	82.06%	0.396

Finally, below are some samples of how the different models transcribed Supreme Court audio:

Sample of transcriptions from the pre-trained model:

- we now vacate the decision of the corpolau and remand
- justis thomas has filed the concurring opinion justis kensberg has filed a discending opinion in which justice todmayor has joined
- at that time we will announce all remaining opinions ready during this term of the court
- federal lor has long prohibited people convicted of felonies from possessing guns

Sample of transcriptions from a TTS-only model:

- we now vacate the decision of the court allow and remand',
- justice thomashas filed the concurring opinion justice ginsburg has filed a dissending opinion on which justice sodomyor has joined
- at that time we will announce all remaining opinions ready during this term of the court
- federal law has long prohibited people convicted of felonies from possessing guns

Sample of transcriptions from 2 years of "hot text" and 1 year of paired data:

- we now vacate the decision of the court alow and remand
- justice thomas has filed a concurring opinion justice ginsburg has filed a dissenting opinion in which justice sotomayor has joined
- at that time we will announce all remaining opinions ready during this term of the court
- federal law has long prohibited people convicted of felonies from possessing guns

10 Citations

Bibliography

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. *arXiv:1512.02595 [cs]*, December 2015. URL http://arxiv.org/abs/1512.02595. arXiv: 1512.02595.

D. Hakkani-Tur, G. Tur, M. Rahim, and G. Riccardi. Unsupervised and active learning in automatic speech recognition for call classification. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–429, May 2004. doi: 10.1109/ICASSP.2004.1326014. ISSN: 1520-6149.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. *arXiv:1910.10261 [eess]*, October 2019. URL http://arxiv.org/abs/1910.10261. arXiv: 1910.10261.

Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, Patrice Castonguay, Mariya Popova, Jocelyn Huang, and Jonathan M. Cohen. NeMo: a toolkit for building AI applications using Neural Modules. *arXiv:1909.09577 [cs, eess]*, September 2019. URL http://arxiv.org/abs/1909.09577. arXiv: 1909.09577.

Aleksandr Laptev, Roman Korostik, Aleksey Svischev, Andrei Andrusenko, Ivan Medennikov, and Sergey Rybin. You Do Not Need More Data: Improving End-To-End Speech Recognition by Text-To-Speech Data Augmentation. *2020 13th International Congress on Image and Signal*

Processing, BioMedical Engineering and Informatics (CISP-BMEI), pages 439–444, October 2020. doi: 10.1109/CISP-BMEI51763.2020.9263564. URL http://arxiv.org/abs/2005.07157. arXiv: 2005.07157.

Jinyu Li, Rui Zhao, Zhong Meng, Yanqing Liu, Wenning Wei, Sarangarajan Parthasarathy, Vadim Mazalov, Zhenghao Wang, Lei He, Sheng Zhao, and Yifan Gong. Developing RNN-T Models Surpassing High-Performance Hybrid Models with Customization Capability. *arXiv:2007.15188 [cs, eess]*, July 2020. URL http://arxiv.org/abs/2007.15188. arXiv: 2007.15188.

Karan Malhotra, Shubham Bansal, and Sriram Ganapathy. Active Learning Methods for Low Resource End-to-End Speech Recognition. In *Interspeech 2019*, pages 2215–2219. ISCA, September 2019. doi: 10.21437/Interspeech.2019-2316. URL http://www.isca-speech.org/archive/Interspeech_2019/abstracts/2316.html.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *Interspeech 2019*, pages 2613–2617, September 2019. doi: 10.21437/Interspeech.2019-2680. URL http://arxiv.org/abs/1904.08779. arXiv: 1904.08779.

G. Riccardi and D. Hakkani-Tur. Active learning: theory and applications to automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(4):504–511, July 2005. ISSN 1558-2353. doi: 10.1109/TSA.2005.848882. Conference Name: IEEE Transactions on Speech and Audio Processing.

Khe Chai Sim, Françoise Beaufays, Arnaud Benard, Dhruv Guliani, Andreas Kabel, Nikhil Khare, Tamar Lucassen, Petr Zadrazil, Harry Zhang, Leif Johnson, Giovanni Motta, and Lillian Zhou. Personalization of End-to-end Speech Recognition On Mobile Devices For Named Entities. *arXiv:1912.09251 [cs, eess, stat]*, December 2019. URL http://arxiv.org/abs/1912.09251. arXiv: 1912.09251.

Jeffrey Tumminia, Amanda Kuznecov, Sophia Tsilerides, Ilana Weinstein, Brian McFee, Michael Picheny, and Aaron R. Kaufman. Diarization of Legal Proceedings. Identifying and Transcribing Judicial Speech from Recorded Court Audio. *arXiv:2104.01304 [cs, eess]*, April 2021. URL http://arxiv.org/abs/2104.01304. arXiv: 2104.01304.

Xianrui Zheng, Yulan Liu, Deniz Gunceler, and Daniel Willett. Using Synthetic Audio to Improve The Recognition of Out-Of-Vocabulary Words in End-To-End ASR Systems. *arXiv:2011.11564 [cs, eess]*, February 2021. URL http://arxiv.org/abs/2011.11564. arXiv: 2011.11564.

Code citations

- NVIDIA's NeMo framework:
  https://github.com/NVIDIA/NeMo. Individual NeMo notebooks we used are also cited in our code.
- Soundfile:
  https://github.com/bastibe/python-soundfile
- PyTorch:
  https://github.com/pytorch/pytorch
- TacoTron2 and WaveGlow:
  https://ngc.nvidia.com/catalog/resources/nvidia:tacotron_2_and_waveglow_for_pytorch.
- Rd-diarization API:
  https://github.com/JeffT13/rd-diarization
- Walkderdb supreme_court_transcripts API:
  https://github.com/walkerdb/supreme_court_transcripts
- NeMo Colab tutorial on segmenting raw video file :
  https://colab.research.google.com/github/NVIDIA/NeMo/blob/main/tutorials/tools/CTC_Segmentation_Tutorial.ipynb
- NeMo Colab tutorial on using ASR:
  https://github.com/NVIDIA/NeMo/blob/main/tutorials/asr/Offline_ASR.ipynb

- Nemo Colab tutorial on using TTS module:
  https://github.com/NVIDIA/NeMo/blob/main/tutorials/tts/1_TTS_inference.ipynb
- NeMo Colab tutorial on ASR:
  https://colab.research.google.com/github/NVIDIA/NeMo/blob/main/tutorials/asr/01_ASR_with_NeMo.ipynb