

LARGE-SCALE LANGUAGE MODEL RESCORING ON LONG-FORM DATA

Tongzhou Chen^{*}, Cyril Allauzen^{*}, Yinghui Huang, Daniel Park, David Rybach, W. Ronny Huang, Rodrigo Cabrera, Kartik Audhkhasi, Bhuvana Ramabhadran, Pedro J. Moreno, Michael Riley

Google LLC, USA

ABSTRACT

In this work, we study the impact of Large-scale Language Models (LLM) on Automated Speech Recognition (ASR) of YouTube videos, which we use as a source for long-form ASR. We demonstrate up to 8% relative reduction in Word Error Rate (WER) on US English (en-us) and code-switched Indian English (en-in) long-form ASR test sets and a reduction of up to 30% relative on Salient Term Error Rate (STER) over a strong first-pass baseline that uses a maximum-entropy based language model. Improved lattice processing that results in a lattice with a proper (non-tree) digraph topology and carrying context from the 1-best hypothesis of the previous segment(s) results in significant wins in rescoring with LLMs. We also find that the gains in performance from the combination of LLMs trained on vast quantities of available data (such as C4 [1]) and conventional neural LMs is additive and significantly outperforms a strong first-pass baseline with a maximum entropy LM.

Index Terms: Large-scale language models, N-best rescoring, Fine-tuning

1. INTRODUCTION

Large-scale language models (LLM), such as BERT [2], T5 [3], GPT-3 [4], and PaLM [5], have proven to be successful in natural language processing (NLP) tasks such as, Question Answering, Text Summarization, and other Zero Shot learning applications. These models are trained on vast amounts of text data and have yielded state-of-the-art results across several NLP and search tasks. However, there is very limited work on the use of these LLMs in Automated Speech Recognition (ASR).

Recent research has focused on fine-tuning GPT, GPT-2 and BERT models with small amounts of in-domain data showing that they tend to outperform the performance of conventional Neural LMs such as transformer LMs trained on the same data [6]. The authors in [7] propose the use of pseudo-likelihood scores and show that rescoring N-best hypotheses from an ASR model can yield significant wins on Librispeech but there is always a trade-off between in-domain modeling and fine-tuning a model trained with far more text. An alternate approach to directly predict the oracle hypothesis was originally proposed in [8] and used in [9] to re-rank the N-best hypothesis using scores from BERT.

In this paper, we scale the use of LLMs to ASR on YouTube videos, which we use as a source for long-form ASR. We show the

importance of lattice quality and contextual augmentation for long-form ASR and compare the performance of LLMs with other neural and maximum entropy based LMs using two metrics: Word Error Rate (WER) and *Salient Term Error Rate* (STER).

2. RELATED WORK

Several methods to incorporate LMs in end-to-end sequence models have been proposed in the literature. Decoding algorithms [10, 11, 12] employ fusion strategies, such as *shallow* [13], *cold* [14], *deep* [15] and *component* [16] fusion. However, the wins from incorporating LMs in this fashion have been relatively small for large scale ASR [17]. The Hybrid Autoregressive Transducer (HAT) model introduced in [18] for encoder-decoder models, allowed for the computation of an internal language model component that can be quantified and appropriately interpolated with an external language model (ELM). The density ratio method proposed in [19] offers a theoretically grounded solution to leverage an external language model while separating out the *acoustic likelihood* score and the internal LM score on the source domain. This modular framework lends itself to principled approaches of LM rescoring and adaptation thus overcoming some of the shortcomings of the aforementioned LM integration strategies [18, 20].

ASR systems perform best when the training data is matched to the target domain. However, end-to-end ASR models are trained on large quantities of available speech data and the LM is trained on the limited text data available in the target domain, thus enabling cross-domain transfer. Alternatively, Large LMs are trained on vast quantities of text and subsequently fine tuned on target domain text. In both scenarios, finding an optimal combination of the end-to-end ASR model, with its implicitly trained internal LM and the external LM, is critical for best performance in the target domain. Neural Oracle Search leverages HAT factorization for LM rescoring with an external LM to directly pick the oracle hypothesis [8], while others have explored on-device neural and biasing LM integration [21] and compared rescoring and deliberation [22], demonstrating wins across all tasks.

In this paper, we study the impact of LLMs within the HAT framework for long-form ASR. Using data from two different sources, US English (en-us) and Indian English (en-in) which is heavily code-switched with Hindi and other Indian languages, we show that wins of up to 8% relative can be obtained in long-form ASR while achieving a reduction of up to 30% relative on Salient Term Error Rate (STER) over a strong first-pass baseline that uses a maximum-entropy based language model. We also demonstrate the importance of improved lattice quality that results in a lattice with a proper (non-tree) digraph topology and carrying context from the 1-best hypothesis of the previous segment(s) obtain best performance with LLMs. We find that both Text-to-Text Transfer Transformer

^{*}Equal Contribution.

(T5) [3] and its multilingual counterpart, MT5 [23] are complementary to conventional neural LMs and outperform a strong first-pass baseline that utilizes a maximum entropy LM.

3. LARGE LANGUAGE MODELS

Several LLMs have been proposed to date with significant improvements on varied NLP tasks. In this work, we mainly focus on two LLMs, T5 and PaLM, ranging in size from 3B to 540B parameters, summarized in Table 1. The conventional neural LM used for comparisons is a conformer LM described in Section 4.4 and comprising of 70M parameters.

Conventional LMs	Size	T5 [3]	Size	MT5 [23]	Size	PaLM [5]	Size
Neural LM	70M	S	60M			S	8B
MaxEnt	4.5B	M	220M			M	62B
		L	770M			L	540B
		XL	3B	XL	3.7B		
		XXL	11B	XXL	13B		

Table 1: Comparison of LM sizes.

3.1. T5 and PaLM

Built on an encoder-decoder transformer-based architecture, T5 optimizes the log-likelihood of the target text given input to learn a mapping from the input to target.

While T5 is pretrained on the span corruption task, LM and Prefix LM are two fine-tuning tasks used for language modeling. The LM task predicts the target sequence with null context input while the prefix LM task randomly splits the text into two halves, using the first half as the input to predict the second half. These fine-tuning tasks enable direct computation of log-likelihood of the target text, instead of the estimation of a pseudo log-likelihood as proposed initially in [2] for masked LMs. Thus, given a text sequence Y , similar to the LM task, we can compute its T5 score $S_{T5}(Y)$ by using an empty string ϵ as input and the text sequence Y as target, with the following equation:

$$S_{T5}(Y) = \log P_{T5}(Y|\epsilon; \Theta_{T5}). \quad (1)$$

For longer sequences, we can make better use of the previous context and compute the score in a semi-autoregressive fashion. Therefore, Y can be split into multiple segments $Y_1 \dots Y_S$ and the log-likelihood of the current segment can be computed using the previous segment’s context:

$$S_{T5}(Y) = \sum_{s=1}^S \log P_{T5}(Y_s|Y_{s-1}; \Theta_{T5}), \quad (2)$$

where Y_0 being ϵ .

PaLM is an autoregressive LM with a decoder-only architecture. Hence the score of a text sequence can be computed straightforwardly.

3.2. Integration with ASR Models

In this work, we use a first-pass model based on the conformer architecture [24] that uses HAT factorization [18]. Not only does HAT model provide a posterior score $S_{HAT}(Y|X)$, but it also estimates the

internal LM (ILM) score. As mentioned in Section 2, when interpolating an external LM during rescoring or shallow fusion, estimating and subtracting the internal LM score yields wins. Thus, inference search maximizes:

$$S(Y, X) = S_{HAT}(Y|X) - \mu S_{ILM}(Y) + \nu S_{ELM}(Y), \quad (3)$$

where μ and ν are tunable hyperparameters.

4. EXPERIMENTS

4.1. Data

We conduct experiments with data from two language locales, en-us and en-in. The multi-domain ASR model used in this paper is trained on several thousand hours of long-form utterances derived from YouTube videos[25] and short-form utterances that are anonymized, hand-transcribed and are representative of Google’s Voice Search traffic [26]. The test sets contain long-form utterances derived from 30-minute-long YouTube videos. We set aside a subset containing 5% of the test utterances as the development test to tune the hyperparameters.

The pre-training corpus used to train T5 is the publicly available, Colossal Clean Crawled Corpus(C4), while MT5 is pre-trained on the multilingual variant, MC4 [23]. To address code-switching seen in en-in [27], text data consisting of Indian English and Hindi Wikipedia and CCNet [28] collectively referred to as WEBDOC, is used. This corpus consists of 170M sentences yielding 2.9B word tokens. We use 90% data for training and 10% data for validation. All data in mixed writing systems is transliterated to Latin to be consistent with ASR model training data used for en-in.

4.2. Training Large Language Models

We experimented with T5 and MT5 models of sizes XL and XXL. Both T5 and MT5 models were pre-trained for 1M steps using the span corruption task and then fine-tuned for 100K steps using the prefix LM task on C4/MC4. To address the heavy code-switching prevalent in en-in and the lack of Hindi data in MC4 corpus, we fine-tune MT5 on the LM task for an additional 300k steps on the WEBDOC corpus.

PaLM models with three different sizes were trained as described in [5] for the en-us task. The corpus used to train these models consisted of filtered web pages, books, Wikipedia, news articles, source code, and social media conversations. We use these pre-trained models as-is with no additional fine-tuning.

4.3. ASR Models

We use a first-pass ASR model based on the conformer architecture [24] that uses HAT factorization [18]. The encoder consists of a convolution subsampling layer and 17-layers of conformer blocks. A conformer block is composed of a feed-forward module, multi-headed self-attention with relative positional encoding module, a convolution and a final feed-forward module, stacked together. The configuration used in this work has an encoder dimension of 512, 8 attention heads, a convolution kernel size of 32 and a decoder dimension of 640 [24]. The decoder at label y_u is only conditioned on the previous two labels y_{u-1} and y_{u-2} , with their embeddings concatenated and projected [29]. The models are trained on 80-dimensional log-mel filter bank coefficients and predict word-piece targets (4096 for en-us and 8192 for en-in). The choice of these parameters was determined by sweeping for best performance within the expected model size.

4.4. Neural and Maximum-Entropy based Language Models

In order to better understand the value of LLMs in ASR, we trained two state-of-the-art LMs, a conventional neural LM and a Maximum Entropy based LM. The conventional Neural LM is a small, unidirectional, conformer LM (CLM) with 70M parameters, originally designed for on-device rescoring [21]. It consists of 12 causal conformer layers, each with a dimension of 384, a feedforward layer dimension of 2048, a convolution kernel of size 15. We use 4-headed self attention with a left context size 31. The model is trained on the same data as the LLMs to predict the same word-piece targets as the first-pass ASR model. Thus, for en-us, we trained it on C4 and for en-in, we trained it on WEBDOC to match the fine-tuning corpus of MT5. The Maximum Entropy based (MaxEnt) LM [30, 31] is a log linear model based on N-gram and skip-gram word contexts, with a size of 4.5B parameters and is comparable to the size of the T5/MT5 XL models. It is also trained on the same data as the conventional Neural LM.

4.5. Decoding and Rescoring

Decoding is performed by a time-synchronous beam search using the breadth-search expansion strategy [32] where the number of active hypotheses at each frame is bounded by a beam size k . A VAD-based segmenter [33] runs in parallel to the beam-search decoder. When the decoder receives an end-of-segment signal from the segmenter, a segment lattice is generated from the currently active hypotheses. If present, a rescoring LM is applied to this segment lattice, with the 1-best hypotheses from previous segments optionally provided as context. Only the best hypothesis in the lattice (eventually after rescoring) is carried forward in the beam-search for the next segment. The final utterance lattice is obtained by concatenating all the segment lattices.

When using an ASR model with unlimited label context, each hypothesis within the beam encodes the full history from the beginning of the utterance. Hence, the segment lattice is a trie with a total number of paths (e.g. hypotheses) bounded by the beam size k .

When using an ASR model where the label context is bound by n [34], beam-search hypotheses sharing the same label context of length n will correspond to the same state in the segment lattice. This results in lattice with a proper (non-tree) digraph topology where the number of paths can grow up to exponentially in the number of states. This was shown to lead to a significant improvement in lattice quality: lattice diversity improvement and oracle WER reduction [34].

The ASR models described in section 4.3 used limited label context with $n = 2$. However when combining these models with the conformer LMs from section 4.4 during the beam search using HAT fusion results in dramatic increase of the label context limit making the resulting combined model to effectively have unlimited label context.

5. RESULTS

5.1. Lattice Quality

The success of a rescoring approach crucially depends on the quality of the hypotheses of the first-pass beam-search decoder. To assess the lattice quality, we computed metrics such as the N -best oracle WER and the average number of paths/hypotheses per segment for our baseline systems on the en-us and en-in development sets as reported in Table 2.

dev	Oracle WER		WER		#paths/segment	
	en-us	en-in	en-us	en-in	en-us	en-in
Baseline	7.3	12.8	12.2	17.2	4e20	4e13
No state merging	8.8	13.1	12.2	17.2	5.7	5.8
Neural LM fusion	8.4	11.0	11.6	15.6	5.2	5.7

Table 2: Lattice quality on the en-us and en-in dev sets.

As the contribution to first-pass model’s posterior and internal LM at label y_u depends only on the previous two labels, our baseline systems can leverage the state merging benefits of limited context models described in Section 4.5 as demonstrated by the relatively low oracle WER and high number of paths per segments.

Lattice quality can be improved by improving first-pass modeling by integrating a neural LM in the beam-search decoding using HAT fusion. Table 2 shows this results in a significant improvement in 1-best WER. However, this causes the loss of the state merging benefits and results in an increase of oracle WER in en-us. However, this is still a significant improvement compared to disabling state merging in the baseline systems.

5.2. Comparison of LMs

In this Section, we consider the impact of LM integration on the en-us task. Table 3 demonstrates the value of providing longer context to Large LMs. Each row contains the result of rescoring with the T5 XXL model when carrying over contexts of different lengths, i.e., of carrying over the 1-best hypotheses from different number of previous segments. We observe that carrying over previous context outperforms no context. However, longer contexts do not seem to provide additional wins. The rest of this paper thus uses contextual information from just the previous segment.

WER	dev
Baseline	12.2
+ T5 rescoring, carrying 0 segment	11.6
+ T5 rescoring, carrying 1 segment	11.5
+ T5 rescoring, carrying 2 segments	11.5

Table 3: WER comparison on the en-us test set for different lengths of carried over context

Table 4 presents the rescoring and fusion results on the en-us development and evaluation test sets for various LMs. First we observe that a small Neural LM edges out over the performance of a Maxent LM. Moreover, though the T5 S model, whose size is slightly smaller than the NLM, was slightly behind NLM, increasing the size of T5 leads to better results. It is also interesting to note that the NLM and T5 XXL models are complementary, as fusion can give a better 1-best WER. In addition, we experimented with more enormous PaLM LMs and they are able to bring the power of larger capacity and large amounts of training text, yielding better results than T5.

5.3. Code-switching Task

In this Section, we present the performance of LLMs on a more challenging en-in task dominated by heavy code-switching.

Although MT5 is meant to be a multilingual LM, the amount of training data from the different languages is unbalanced. The training data consists of 5.67% English, but only 1.21% is Hindi in the Devanagari script [23]. This imbalance between en-in and

WER	dev	eval
Baseline	12.2	16.1
+ MaxEnt rescoring	12.2	16.4
+ NLM rescoring	11.8	15.8
+ T5 S rescoring	11.9	15.9
+ T5 M rescoring	11.7	15.8
+ T5 XL rescoring	11.6	15.7
+ T5 XXL rescoring	11.5	15.7
+ PaLM S rescoring	11.5	15.5
+ PaLM M rescoring	11.3	15.4
+ PaLM L rescoring	11.3	-
+ NLM fusion	11.6	15.6
+ NLM fusion & T5 XXL rescoring	11.4	15.5

Table 4: en-us WER comparison between T5 and other LMs

Hindi fails to capture the frequent code switches between English and Hindi predominant in the en-in test sets. To address this issue, we finetune both XL and XXL MT5 models on the WEBDOC corpra with the LM task. We evaluate the raw MT5 model and these fine-tuned models on the en-in development set to study the effect of fine-tuning. These results are tabulated in Table 5.

en-in dev	MT5 XL	MT5 XXL
Baseline		17.2
Raw	16.6	16.8
Fine-tuned	16.1	16.3

Table 5: WER comparison on en-in dev set with raw and fine-tuned MT5 models of sizes XL and XXL

It can be seen that rescoring with the fine tuned models outperforms rescoring with the raw MT5 model. This can be attributed to the lack of sufficient Hindi data in the MC4 corpus which can be fixed with data balanced fine-tuning. When compared to en-us, the wins from LLMs on en-in are less. We hypothesize that this could be related to the small size of the WEBDOC corpus compared to MC4, in line with the data-hungry nature of LLMs [35, 36].

5.4. Comparison of LMs on the code-switching task

WER	dev	eval
Baseline	17.2	16.4
+ MaxEnt rescoring	16.5	15.9
+ NLM rescoring	16.2	15.4
+ MT5 XL rescoring	16.1	15.2
+ NLM fusion	15.6	15.0
+ NLM fusion & MT5 XL rescoring	15.4	14.6

Table 6: en-in WER comparison between MT5 and other LMs

Table 6 presents the rescoring results from various LMs. The MT5 XL model is the best performing model with a WER reduction of 7.3% relative on the evaluation test set. On the other hand, the Conformer LM when used in shallow fusion in the first-pass shows additional wins. Since we fine-tuned MT5 on the same training data as Conformer LM, we also report the perplexity of MT5 and Conformer LM on the 10% validation part of WEBDOC. MT5 has a log perplexity per word of 4.15, slightly higher than the Conformer LM at 2.98 and MaxEnt at 3.69.

We observe that the Conformer LM and MT5 are complementary and the combination results in a best WER reduction of 8% relative.

6. ERROR ANALYSIS

To analyze the effectiveness of large LM, we select unigrams and bigrams with the highest Term Frequency Inverse Document Frequency (TF-IDF) values from the evaluation test sets (*salient terms*) for the two languages studied in this paper. In general, such terms capture the topic presented in the video. On the one hand, they are important for indexing or information retrieval; on the other hand, they are more difficult to be recognized compared to frequently occurring function words (such as, "the", "of", etc.). We analyzed the performance of the baseline and the various large LMs on these *salient terms* to study the impact on rare words. The *Salient Term Error Rate (STER)* is reported in Table 7, defined as the number of deletion and substitution errors on the *salient terms* divided by the total number of *salient terms*. Out of a total of 600K words, approximately, 10% words are tagged as *salient terms* for en-in and 5% for en-us. First we observe that almost all rescoring and fusion can reduce the error made on these *salient terms*. In en-us, as reflected by the WER reported in Table 4, T5 outperforms other LMs. In en-in, however, NLM fusion in the first pass has a bigger impact on the *salient terms* than any rescoring method similar to what has been reported in [37]. Although MT5 has been fine tuned on the same data as the NLM, we find that it is less impactful by itself on the *salient terms* in en-in.

Although MT5 has been fine tuned on the same data as the neural LM, we find that it is less impactful by itself on the *salient terms*. However, in both languages, the combination of these two LMs through interpolation is additive (last row in Table 6) resulting in the best performance. As noted in [35, 36] scaling to larger and larger datasets is only beneficial when the data is high-quality and larger models require larger data sets. This can explain some of the differences seen between these two relatively high resource languages.

STER	en-us	en-in
Baseline	28.8	20.0
+ MaxEnt rescoring	28.8	17.4
+ NLM rescoring	27.4	16.7
+ T5/MT5 rescoring	26.7	17.6
+ NLM fusion	27.2	15.4
+ NLM fusion & T5/MT5 rescoring	26.4	12.1

Table 7: Errors analysis on *salient terms* of en-us and en-in.

7. CONCLUSION

In this study, we presented the impact of LLMs (up to 350B parameters) on long-form ASR. We demonstrated up to 8% relative reduction in Word Error Rate (WER) on US English (en-us) and code-switched Indian English (en-in) long-form ASR test sets and a reduction of up to 30% relative on Salient Term Error Rate (STER) over a strong first-pass baseline that uses a maximum-entropy based language model. We also find that the gains in performance from the combination of LLMs trained on vast quantities of available data (such as C4 [1]) and conventional neural LMs is additive and significantly outperforms a strong first-pass baseline with a maximum entropy LM. To the best of our knowledge, this is the first study that scales LLMs to long-form ASR.

8. REFERENCES

- [1] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer.” *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
- [2] A. Wang and K. Cho, “Bert has a mouth, and it must speak: Bert as a markov random field language model,” *arXiv preprint arXiv:1902.04094*, 2019.
- [3] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.
- [4] T. Brown *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [5] A. Chowdhery *et al.*, “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*, 2022.
- [6] X. Zheng, C. Zhang, and P. C. Woodland, “Adapting gpt, gpt-2 and bert language models for speech recognition,” in *2021 IEEE ASRU*, 2021, pp. 162–168.
- [7] J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff, “Masked language model scoring,” in *2020 ACL*, Jul. 2020.
- [8] E. Variani *et al.*, “Neural oracle search on n-best hypotheses,” in *ICASSP*, 2020, pp. 7824–7828.
- [9] S.-H. Chiu and B. Chen, “Innovative bert-based reranking language models for speech recognition,” in *SLT*. IEEE, 2021, pp. 266–271.
- [10] T. Hori, Y. Kubo, and A. Nakamura, “Real-time one-pass decoding with recurrent neural network language model for speech recognition,” in *ICASSP*. IEEE, 2014, pp. 6364–6368.
- [11] J. Chorowski and N. Jaitly, “Towards better decoding and language model integration in sequence to sequence models,” *arXiv preprint arXiv:1612.02695*, 2016.
- [12] T. Hori, J. Cho, and S. Watanabe, “End-to-end speech recognition with word-based rnn language models,” in *SLT*. IEEE, 2018, pp. 389–396.
- [13] C. Peyser *et al.*, “Improving tail performance of a deliberation e2e asr model using a large text corpus,” *arXiv preprint arXiv:2008.10491*, 2020.
- [14] A. Sriram, H. Jun, S. Satheesh, and A. Coates, “Cold fusion: Training seq2seq models together with language models,” *arXiv preprint arXiv:1708.06426*, 2017.
- [15] C. Gulcehre *et al.*, “On using monolingual corpora in neural machine translation,” *arXiv preprint arXiv:1503.03535*, 2015.
- [16] C. Shan *et al.*, “Component fusion: Learning replaceable language model component for end-to-end speech recognition system,” in *ICASSP*. IEEE, 2019, pp. 5361–5365.
- [17] A. Kannan *et al.*, “An analysis of incorporating an external language model into a sequence-to-sequence model,” in *ICASSP*, 2018, pp. 1–5828.
- [18] E. Variani, D. Rybach, C. Allauzen, and M. Riley, “Hybrid autoregressive transducer (hat),” in *ICASSP*, 2020, pp. 6139–6143.
- [19] E. McDermott, H. Sak, and E. Variani, “A density ratio approach to language model fusion in end-to-end automatic speech recognition,” in *ASRU*, 2019, pp. 434–441.
- [20] C. Allauzen, E. Variani, M. Riley, D. Rybach, and H. Zhang, “A hybrid seq-2-seq ASR design for on-device and server applications,” in *Interspeech 2021*, 2021, pp. 4044–4048.
- [21] T. N. Sainath *et al.*, “An efficient streaming non-recurrent on-device end-to-end model with improvements to rare-word modeling,” in *Interspeech*, 2021, pp. 1777–1781.
- [22] K. Hu *et al.*, “Improving deliberation by text-only and semi-supervised training,” *arXiv preprint arXiv:2206.14716*, 2022.
- [23] L. Xue *et al.*, “mT5: A massively multilingual pre-trained text-to-text transformer,” in *2021 NAACL: Human Language Technologies*, Jun. 2021.
- [24] A. Gulati *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” *arXiv preprint arXiv:2005.08100*, 2020.
- [25] H. Liao, E. McDermott, and A. Senior, “Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription,” in *ASRU*, 2013, pp. 368–373.
- [26] A. Narayanan *et al.*, “Recognizing long-form speech using streaming end-to-end models,” in *ASRU*, 2019, pp. 920–927.
- [27] J. Emond, B. Ramabhadran, B. Roark, P. Moreno, and M. Ma, “Transliteration based approaches to improve code-switched speech recognition performance,” in *SLT*. IEEE, 2018, pp. 448–455.
- [28] G. Wenzek *et al.*, “Ccnets: Extracting high quality monolingual datasets from web crawl data,” *arXiv preprint arXiv:1911.00359*, 2019.
- [29] R. Botros *et al.*, “Tied & reduced rnn-t decoder,” *arXiv preprint arXiv:2109.07513*, 2021.
- [30] F. Biadsy, K. Hall, P. Moreno, and B. Roark, “Backoff inspired features for maximum entropy language models,” 2014.
- [31] F. Biadsy, M. Ghodsi, and D. Caseiro, “Effectively building tera scale maxent language models incorporating non-linguistic signals,” 2017.
- [32] A. Tripathi, H. Lu, H. Sak, and H. Soltau, “Monotonic recurrent neural network transducer and decoding strategies,” in *ASRU*, 2019, pp. 944–948.
- [33] R. Zazo, T. N. Sainath, G. Simko, and C. Parada, “Feature learning with raw-waveform cldnns for voice activity detection,” in *Interspeech*, 2016, pp. 3668–3672.
- [34] R. Prabhavalkar *et al.*, “Less is more: Improved rnn-t decoding using limited label context and path merging,” in *ICASSP*, 2021, pp. 5659–5663.
- [35] J. Kaplan *et al.*, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.
- [36] J. Hoffmann *et al.*, “Training compute-optimal large language models,” *arXiv preprint arXiv:2203.15556*, 2022.
- [37] V. Ravi *et al.*, “Improving accuracy of rare words for rnn-transducer through unigram shallow fusion,” *arXiv preprint arXiv:2012.00133*, 2020.