INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Joint Grapheme and Phoneme Embeddings for Contextual End-to-End ASR

Zhehuai Chen[1,2,*], *Mahaveer Jain*[2], *Yongqiang Wang*[2], *Michael L. Seltzer*[2], *Christian Fuegen*[2]

[1]SpeechLab, Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Facebook AI, One Hacker Way, Menlo Park, CA 94025, USA

chenzhehuai@sjtu.edu.cn, {jainmahaveer,yqw,mikeseltzer,fuegen}@fb.com

Abstract

End-to-end approaches to automatic speech recognition, such as Listen-Attend-Spell (LAS), blend all components of a traditional speech recognizer into a unified model. Although this simplifies training and decoding pipelines, a unified model is hard to adapt when mismatch exists between training and test data, especially if this information is dynamically changing. The Contextual LAS (CLAS) framework tries to solve this problem by encoding contextual entities into fixed-dimensional embeddings and utilizing an attention mechanism to model the probabilities of seeing these entities. In this work, we improve the CLAS approach by proposing several new strategies to extract embeddings for the contextual entities. We compare these embedding extractors based on graphemic and phonetic input and/or output sequences and show that an encoder-decoder model trained jointly towards graphemes and phonemes outperforms other approaches. Leveraging phonetic information obtains better discrimination for similarly written graphemic sequences and also helps the model generalize better to graphemic sequences unseen in training. We show significant improvements over the original CLAS approach and also demonstrate that the proposed method scales much better to a large number of contextual entities across multiple domains.

Index Terms: End-to-end Speech Recognition, Deep Context, CLAS, Sequence Pooling, Grapheme-to-Phoneme (G2P).

1. Introduction

End-to-end (E2E) automatic speech recognition (ASR) has recently become popular as a result of both advances in neural modeling of context and history in sequences [1, 2], and access to large amounts of labeled training data that improves generalization. In E2E speech recognition, a single model predicts hypotheses directly from speech, unifying the acoustic, language, and pronunciation models in one system. Although E2E training benefits from sequence modeling and simplified inference [3, 4], long and noisy speech utterances are a challenge. In order to perform well in such scenarios, a larger amount of transcribed acoustic data is required, compared to more modular systems based on the combination of Hidden Markov Models (HMM) with Deep Neural Networks (DNN) or Connectionist Temporal Classification (CTC). Moreover, advances in language modeling for modular systems are difficult to apply in E2E systems [5, 6, 7]. In particular, the inability to exploit knowledge from external language models and lexicons hampers the adaptability of E2E systems.

In *contextual speech recognition*, additional context such as device-specific information or personalized user information is used to improve the recognition of a user's query [8]. In such scenarios, the ability to integrate such dynamic information is particularly important.

One method of incorporating context in E2E speech recognition is called *shallow fusion* [9, 10]. In this approach, a contextual language model (LM) is generated on-the-fly and used during recognition to bias the beam search decoding in E2E models. While simple to implement, its effectiveness is limited because the contextual information is not incorporated into the E2E model itself. Contextual Listen-Attend-Spell (CLAS) addresses this drawback by jointly optimizing the ASR components along with embeddings derived from n-grams of contextual entities [11]. However, because the embeddings are learned from graphemic information only, they do not discriminate well among similar sequences of graphemes, nor do they generalize well to unseen pronunciations of words.

To overcome these problems, we propose two extensions to CLAS. First, we extract embeddings based on a grapheme-to-phoneme (G2P) encoder-decoder and experiment with several ways to fuse the grapheme-to-grapheme (G2G) and G2P encoder models. Second, we experiment with different sequence pooling methods that try to better leverage the power of the bidirectional long-short-term memory (LSTM) model that we used as a replacement for the LSTM model proposed in CLAS.

The rest of the paper is organized as follows. In Section 2, prior work of E2E contextual speech recognition is briefly reviewed. Our main contributions are in Section 3: i) the analysis of various encoder-decoder models and sequence pooling methods to obtain the contextual embeddings. ii) the integration of phoneme information into the encoder-decoder. Other related prior work is discussed in Section 2.3. Experimental results are presented in Section 4.

2. End-to-end Contextual ASR

The E2E model used in this work is based on Listen-Attend-Spell (LAS), an attention-based encoder-decoder model [12, 13]. The model predicts the posterior probability of label sequences given both a feature sequence and previous inference labels. The encoder is typically an unidirectional or bidirectional LSTM (BLSTM) network while the attention-based decoder is an unidirectional LSTM.

Compared to a traditional decoder in a hybrid system [14], the attention decoder implicitly captures LM information in a way that is jointly trained with the encoder. Because of this tight unification, E2E systems are hard to adapt to new domains or contexts. In contrast, traditional modular systems can do this easily via updates to the language model to bias the decoding [8].

2.1. Contextual speech recognition using shallow fusion

A contextual speech recognizer dynamically incorporates real-time contextual information into its recognition process [8]. A typical example of such information is a user's contact list. *Shallow fusion* [10, 9] solves this by generating on-the-fly

*This work was done when the first author was a Facebook intern.

http://dx.doi.org/10.21437/Interspeech.2019-1434

contextual LMs that are interpolated with E2E neural model's scores to bias the beam search during decoding. This method was further improved by using a token-passing decoder with efficient token recombination to minimize search errors when the number of contextual entities is large [15]. While this showed improvements over standard shallow fusion, there are still gaps between E2E and traditional modular systems in contextual ASR. One problem is that shallow fusion assumes that the exact sequence of words used in the speech utterance is available in the list of contextual entities provided to the system. Another problem is that the provided contextual entities only affect the beam search but are not integrated directly into the E2E neural model.

2.2. Contextual LAS

Another method that tries to integrate contextual information into the E2E models is called Contextual LAS (CLAS) [11]. This technique embeds possible contextual entities, represented as a sequence of graphemes, into fixed-dimensional vectors and utilizes an attention mechanism to model the probability of seeing a particular contextual entity. The disadvantage of this approach is that contextual entities are represented using a fixed dimensional embedding vector. These vectors might not be discriminative enough for common prefixes and suffixes when a large number of entities is used. Moreover, it only encodes sequences of graphemes. For vocabulary words unseen in training, the neural model may by unable to properly match graphemes to acoustics.

2.3. Related Prior Work

The work described in this paper is an extension of CLAS [11]. Here we briefly mention other related prior work. In machine translation, there have been several efforts to map the source language sentence into a fixed-dimensional embedding vector and then map this vector back to the target language sentence [16, 17]. Inspired by this research, we utilize z similar structure as the embedding extractor for deep contextual speech recognition.

In Deep Fusion [18], a seq2seq model and a language model are trained independently and combined together for speech recognition. Cold Fusion [19] trains a seq2seq model jointly with a pre-trained LM. The goal of both Cold and Deep Fusion approaches is to capture generic textual information from large amounts of unsupervised text data whereas our work aims to capture contextual information during inference time.

A modular training framework for general E2E ASR was proposed to separately train an acoustic-to-phoneme model (A2P) and a phoneme-to-word model (P2W) using acoustic and text data respectively, while still performing end-to-end inference in [6]. The system proposed in this paper shares the same motivation of integrating phoneme information into E2E ASR.

A neural G2P has been investigated in [20]. In [20], the implicit alignment between graphemes and phonemes is modeled by LSTMs. We believe that our proposed method shows better performance on the alignment learning because of stronger sequential modeling effects in the encoder-decoder model [16].

Most recently, [21] proposed an approach to contextual ASR in which grapheme and phoneme embeddings are extracted independently and then combined using a special attention mechanism. In contrast, our approach extracts embeddings for both graphemes and phonemes jointly. Moreover, [21] relies on a separate G2P module while we integrate the G2P module into the embedding extractor.

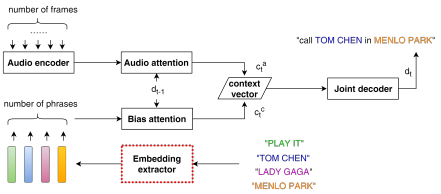

Figure 1: *The Framework of CLAS.*

3. Proposed Methods

3.1. CLAS framework

Figure 1 shows the CLAS framework from [11]. As the LAS model, the contextual acoustic vector c_t^x is obtained as a function of the previous decoder state d_{t-1} and the output of an audio encoder model using the attention mechanism, *audio attention*. We refer to this network as *main network* (black boxes in Figure 1) while reporting experiments in section 4. Another encoder model, which we refer to as the *embedding extractor*, is used to extract the contextual embedding for each user entity z. The contextual entity vector c_t^z is obtained as a function of the previous decoder state d_{t-1} and the extracted contextual embeddings of all z in an attention mechanism, *bias attention*. The concatenation of both c_t^z and c_t^x represents the combined context vector and is fed into a decoder to infer the decoder state d_t with probabilities over all possible grapheme output units. Beam search is used to identify the best hypothesis.

We propose two changes in this work that improves upon CLAS. These changes are solely focused on the *embedding extractor*. First, to better convey contextual information, a more discriminative *embedding extractor* is pre-trained. Second, to obtain better discrimination for similar graphemic sequences and generalize to unseen pronunciations of words, a grapheme-to-phoneme (G2P) module is introduced into the *embedding extractor*.

3.2. Grapheme embeddings

One naive method to obtain an embedding of contextual entity is to encode the sequence of graphemes into a fixed-dimensional vector with the help of an LSTM language model (LM) and use the last encoder state as embedding. We refer to this method as *NNLM*. The problem with this approach is that the LM criterion uses cross entropy to maximize the posterior probability of the current grapheme given the history of previous graphemes [22] and this can only capture the context around the current grapheme but fails to model the whole entity. This serves as one baseline in our experiments.

In addition, we propose several methods to extract embeddings of contextual entities that utilize different encoder models and sequence pooling methods. We first train a grapheme-to-grapheme encoder-decoder model using the same grapheme sequence as input and output. Once the encoder-decoder model is trained, we discard the decoder and use the encoder as an *embedding extractor*. All proposed entity extractors use a bi-directional LSTM (BLSTM) as the encoder as it has better sequential modeling effects compared to an unidirectional LSTM.

The first set of sequence pooling methods learn the attention weights over the encoder states. This encoder-decoder model was trained using location-based attention. After that, we fix the parameters of the model and use its encoder as the *embedding extractor*. The attention mechanism is discarded.and the encoder states are averaged to represent the embedding of the

entity. We refer to this method as *EncoderAttention*.

The second set of sequence pooling methods use deterministic and not learned attention weights over the encoder states. We call them *fixed encoder-decoder* approaches. The simplest such method uses the last state of a BLSTM encoder. We refer to this as *EncoderLast*. This is similar to the original CLAS approach [11], except that the unidirectional LSTM encoder is replaced with a bidirectional one. A variation of this is *EncoderBoundary* where the first and the last BLSTM states of the encoder are concatenated as the embedding of entity. Finally, in *EncoderAverage*, the dimension-wise average over all BLSTM states of the encoder is used as embedding of entity.

3.3. Joint grapheme and phoneme embeddings

The grapheme embedding extractors described above have the disadvantage that they can only encode graphemic sequences usually seen in the training data. This may impair performance on unseen grapheme sequences especially in use cases with large lists of entities that are unseen during training, such as contact lists. In this scenario, it is challenging for the system to learn the mapping between graphemes and acoustics.

The use of grapheme-to-phoneme embeddings helps with this problem because it provides additional information to discriminate similar grapheme sequences with different pronunciations. In addition, it scales better to unseen entities because it maps grapheme sequences into a space more correlated with the acoustics and better covered by the training data.

There are several ways in which grapheme and phoneme information can be combined and leveraged in the CLAS framework. Figure 2 shows the variations explored in this work. For learning the grapheme-to-phoneme embeddings, we use a separate pronunciation dictionary.

Grapheme-to-phoneme (Figure 2a) takes a grapheme sequence as input and produces a phoneme sequence as output. *Phoneme-to-grapheme* (Figure 2b) takes a phoneme sequence as input and produces a grapheme sequence as output. *Phoneme-to-phoneme* (Figure 2c) takes a phoneme sequence as input and produces a phoneme sequence as output.

Phonemes and graphemes have complementary discriminative power. The use of phonetic information helps discriminate homographs or near homographs, while the use of graphemic information helps discriminate homophones or near homophones. Thus, we explored two ways to create representations based on both. The first approach is inspired by [23, 24]. We append each word with its corresponding phoneme sequence and predict the combined grapheme and phoneme sequences from a grapheme input, as shown in Figure 2d. In the second approach, the phoneme and grapheme embeddings are learned jointly using multi-task learning as shown in Figure 2e.

Note that for all approaches, we discard the decoder and only use the encoder's output as the entity embedding. Also, after the embedding extractor has been trained, only the encoders in Figure 2(b,c) requires a pronunciation dictionary during inference time. In Figure 2(d,e), the pronunciation information has been implicitly learned by the model thus offering some advantages during inference time.

4. Experiments

4.1. Setup

The proposed embedding extraction methods are evaluated on an in-house corpus which contains a range of spoken commands for multiple domains, such as music, weather, timer,

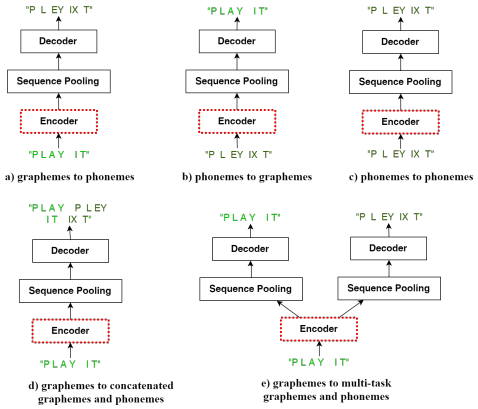

Figure 2: *Embedding Extractors that uses phonetic information during their training.*

and calling. There are 5 million ASR training utterances (approximately 4,000 hours) which have at least one named entity (tagged by an in-house entity tagger), referred to as *train-full*. Among these domains, calling (referred to as *call*) is the most challenging domain due to the vast variety of name pronunciations (usually consists of more than one word). For faster turn around of experiments, we shuffle out two subsets from this training dataset: the first, referred to as *train-multi-100K*, has about 100,000 utterances sampled from all the domains, while the second, referred to as *train-call-100K* consists of 100,000 utterances sampled from the calling domain only. For each of these two subsets, 1,000 utterances are held out as test sets, referred to as *test-multi* and *test-call*. Care was taken to make sure entities in the test sets did not appear in the training set.

In the training stage, unless explicitly mentioned, the *embedding extractor* is trained on all named entities of *train-full*. We use 400-dimensional embeddings in all experiments. After the *embedding extractor* is trained, it is used to extract entity embeddings in both the training and inference stages. While training the *main network*, we first extract the ground truth named entity for each utterance and then randomly sample 40 other named entities from the list of all named entities of the domain of utterance. The embeddings for these 41 named entities (except the experiments in Table 2) are extracted using the *embedding extractor* and fed to the *main network*. Since the alternative 40 entities are sampled from the same domain, the *main network* is trained to utilize contextual information and distinguish confusing entities. The *main network* follows the LAS architecture [12], with a 2-layer BLSTM with 1400 hidden nodes per layer as the encoder and 2-layer LSTM with 700 hidden nodes as the decoder. More details can be found in [15]. We built the system with PyTorch [25] based on Espnet [26], and implemented Block-momentum SGD [27] to enable distributed training with linear speedups and no performance degradation.

4.2. Embedding Extractor

In the first set of experiments, we investigate the effectiveness of the embedding extraction methods proposed in Section 3.2. The *main network* is trained on *train-call-100K* set and tested on *test-call* set. For the sequence-to-sequence embedding extractors, we use a two-layer LSTM or BLSTM with 512 hidden nodes per layer as the encoder and a 1-layer LSTM with 512 hidden nodes as the decoder.

Results are presented in Table 1. Without any bias attention (the LAS baseline), the WER on *test-call* is 28.7%. Following

Table 1: *Comparisons of Embedding Extractors.*

Embedding Enc. Arch.	Embedding Extractor	WER
n/a	n/a	28.7
LSTM	*EncoderLast*	20.2
	NNLM	28.7
BLSTM	*EncoderAttenion*	19.5
	EncoderLast	12.5
	EncoderAverage	10.3
	EncoderBoundary	6.7

[16], we trained a sequence-to-sequence auto-encoder and used the last state of the LSTM encoder as the entity embedding, i.e., *EncoderLast*. With this embedding extractor, the WER is improved to 20.2%. We tried to replace the entity auto-encoder by an NNLM, using the last state of the NNLM as the embedding. However, this did not yield any improvement over the baseline. With *EncoderAttention*, WER improves to 19.5% from 20.2%. In this case, when the embedding extractor is trained, an attention-weighted encoder output pooling is used at each decoding step, while an average pooling of encoder output is used at inference time.

To address this inconsistency, we evaluated the proposed *fixed encoder-decoder* approaches, where the output of encoder is pooled by averaging (*EncoderAverage*) or by concatenating the first and last output vector of the encoder (*EncoderBoundary*) output, both in training and inference. This yields significant gains, as seen in the last two rows of Table 1. Since *EncoderBoundary* with a BLSTM encoder yields the best performance, we use it in the remainder of the experiments.

The embedding extractors used in the previous experiments are all trained on the named entities extracted from the 5 million utterances in *train-full*. Table 2 shows the effect of varying the number of utterances used to train the *EncoderBoundary* embedding extractor on the WER in the *test-call* set. It is observed that the generalization power of embedding extractor increases as the number of entities to train it increases. Note that since the embedding extractor only requires limited effort to collect named entities, it is possible that we can scale the embedding extractor training data beyond the acoustic training data and possibly achieve even better generalization. This will be investigated in the future.

Table 2: *Utterances used in training embedding extractor*

# utterances	100K	500K	1M	2.5M	5M
WER	24.0	18.1	14.9	11.0	6.7

4.3. Integrating Phonetic Representations

In the second set of experiments, we investigate the impact of integrating phonetic information into the *embedding extractor*. An in-house G2P model based on [28] is used to get the best estimated pronunciation for all named entities. As before, the *embedding extractor* is trained on the named entities from *train-full*, while the *main network* is trained on the *call* subset and tested on the *call-test* subset. We will present evaluation results with this method in scenario of multiple domains in the next section. Results of the various architectures shown in Figure 2 are compared in Table 3. Using phoneme sequences as input results in worse performance compared to the grapheme-to-grapheme baseline. We suspect that this may be attributed to the nature of names, where there are usually multiple pronunciations for the same graphemic form but only one phoneme sequence can be selected as the input. Comparing grapheme-to-phoneme with phoneme-to-grapheme approaches, the former explicitly encodes the grapheme information while the latter one only implicitly infers graphemes. Since the final objective of E2E ASR is to recognize graphemic sequences, explicitly encoding grapheme sequences is helpful. To model graphemic and phonetic information jointly, we use *Grapheme* $\oplus$ *Phoneme* extractor that appends graphemic and phonetic sequences together as shown in Figure 2(d) in Section 3.3. The phoneme and grapheme embeddings are learned jointly using multi-task learning in *Grapheme* $\otimes$ *Phoneme* as shown in Figure 2(e). The latter one shows better performance and significantly improves the original CLAS method.

Table 3: *Methods to Integrate Phonemes*

Input Seq.	Output Seq.	WER
Grapheme	Grapheme	6.7
Phoneme	Phoneme	14.3
Phoneme	Grapheme	10.0
Grapheme	Phoneme	4.2
Grapheme	Grapheme $\oplus$ Phoneme	4.7
Grapheme	Grapheme $\otimes$ Phoneme	3.7

Integrating phonetic information also helps when more confusion is introduced in the bias list by increasing the number of alternative named entities during inference time. Table 4 shows the WER trend when the number of entities presented in the bias list increases during inference time.

Table 4: *Number of entities in the bias list vs WER*

# entities	5	40	100	300	500	1000
EncoderBoundary	1.5	6.7	7.0	8.0	9.0	12.0
+Phoneme	2.7	3.7	3.1	3.3	4.0	4.6

4.4. Test on Multiple Domains

In the last set of experiments, we compare the baseline and proposed models on the multiple-domain test set (*multiple-test*). The *main network* is trained on either the subset *train-100K* or the full training set *train-full*. Results are presented in Table 5. The proposed method shows consistent and significant improvements over LAS and CLAS. The improvement comes from both better *EncoderBoundary* embedding in Section 4.2 and integrating phonemes in Section 4.3.

Table 5: *WER evaluation on the multiple-domain test set.*

Systems	Training Set	
	train-100K	*train-full*
LAS	34.1	22.6
CLAS [11]	28.0	18.5
+ *EncoderBoundary*	25.1	13.6
+ Phoneme	23.2	12.5

5. Conclusions

We extended the CLAS framework by incorporating a grapheme-to-phoneme encoder-decoder model and showed that leveraging both phonetic and graphemic information obtains better discrimination for similarly written graphemic sequences. It also helps the model to generalize better to graphemic sequences unseen in training. In combination with a BLSTM based embedding extractor that concatenates the first and last BLSTM state of the encoder into an entity embedding, we showed significant improvements over CLAS.

6. References

[1] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[2] Z.-H. Chen, J. Droppo, J. Li, and W. Xiong, "Progressive joint modeling in unsupervised single-channel overlapped speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 1, pp. 184–196, 2018.

[3] Z.-H. Chen, Y. Zhuang, Y. Qian, and K. Yu, "Phone Synchronous Speech Recognition With CTC Lattices," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 86–97, Jan 2017.

[4] Z.-H. Chen, Y. Qian, and K. Yu, "A unified confidence measure framework using auxiliary normalization graph," in *International Conference on Intelligent Science and Big Data Engineering*. Springer, 2017, pp. 123–133.

[5] T. Hori, S. Watanabe, and J. R. Hershey, "Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017, pp. 287–293.

[6] Z. Chen, Q. Liu, H. Li, and K. Yu, "On Modular Training of Neural Acoustics-to-word Model for LVCSR," in *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, Calgary, Canada, April 2018.

[7] Y. Wang, X. Fan, I.-F. Chen, Y. Liu, T. Chen, and B. Hoffmeister, "End-to-end anchored speech recognition," *arXiv preprint arXiv:1902.02383*, 2019.

[8] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays *et al.*, "Personalized speech recognition on mobile devices," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5955–5959.

[9] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[10] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. N. Sainath, "Contextual speech recognition in end-to-end neural network systems using beam search," *Proc. Interspeech*, pp. 2227–2231, 2018.

[11] G. Pundak, T. N. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: end-to-end contextual speech recognition," *arXiv preprint arXiv:1808.02480*, 2018.

[12] W. Chan, "End-to-end speech recognition models," Ph.D. dissertation, Carnegie Mellon University Pittsburgh, PA, 2016.

[13] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[15] Z. Chen, M. Jain, Y. Wang, M. Seltzer, and C. Fuegen, "End-to-end contextual spebech recognition using class language models and a token passing decoder," in *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, Brighton, UK, May 2019.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[17] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[18] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[19] S. S. A. Sriram, H. Jun and A. Coates, "Cold fusion: Training seq2seq models together with language models," *arXiv preprint arXiv:1708.06426*, 2017.

[20] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4225–4229.

[21] A. Bruguier, R. Prabhavalkar, G. Pundak, and T. N. Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6171–6175.

[22] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[23] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4759–4763.

[24] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing Acoustic-to-Word CTC Model," *arXiv preprint arXiv:1803.05566*, 2018.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[26] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "Espnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.

[27] K. Chen and Q. Huo, "Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5880–5884.

[28] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.