Real-time One-pass Decoder for Speech Recognition Using LSTM Language Models

Javier Jorge, Adrià Giménez, Javier Iranzo-Sánchez, Jorge Civera, Albert Sanchis, Alfons Juan

Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València (Spain)

{jjorge,agimenez,jairsan,jcivera,josanna,ajuan}@dsic.upv.es

Abstract

Recurrent Neural Networks, in particular Long-Short Term Memory (LSTM) networks, are widely used in Automatic Speech Recognition for language modelling during decoding, usually as a mechanism for rescoring hypothesis. This paper proposes a new architecture to perform real-time one-pass decoding using LSTM language models. To make decoding efficient, the estimation of look-ahead scores was accelerated by precomputing static look-ahead tables. These static tables were precomputed from a pruned n-gram model, reducing drastically the computational cost during decoding. Additionally, the LSTM language model evaluation was efficiently performed using Variance Regularization along with a strategy of lazy evaluation. The proposed one-pass decoder architecture was evaluated on the well-known LibriSpeech and TED-LIUMv3 datasets. Results showed that the proposed algorithm obtains very competitive WERs with $\sim$0.6 RTFs. Finally, our one-pass decoder is compared with a decoupled two-pass decoder.

Index Terms: ASR, one-pass decoding, real-time, LSTM-LM

1. Introduction

Recurrent Neural Networks (RNNs) and particularly Long-Short Term Memory (LSTM) networks are widely used to build Language Models (LMs) for Large-Vocabulary Continuous Speech Recognition (LVCSR) [1, 2, 3]. An initial recognition step is first applied on the basis of an n-gram LM, from which a set of best hypotheses is produced (e.g. an N-best list or a lattice). Then, a second recognition step is carried out in which an LSTM-based LM is used for hypothesis rescoring [4, 5, 6, 7, 8, 9]. The use of this two-steps approach instead of a more direct, one-pass decoding, is needed to overcome the high computational cost associated with the LSTM-LM. That is, by applying an LSTM-LM to a limited set of best hypotheses, we take advantage of its high accuracy while keeping the decoding time under reasonable levels.

Although the above two-step approach is still the preferred way to develop fast yet accurate ASR systems, we think that it will be soon replaced by one-pass decoding alone, not only to avoid cascade errors but, more importantly, to leverage the full potential of state-of-the-art ASR at real-time. To our knowledge, the direct use of Neural LMs during decoding was first explored in [10], where the authors proposed the use of a Variance Regularization (VR) term along with caching strategies to speed-up this process. Despite using (feed-forward) Neural LMs in decoding early was seen as a big challenge at that time, empirical results showed significant relative improvements both in speed and accuracy. Other relevant contributions addressing this challenge have focused on applying heuristics to reduce model's queries and caching the network's states [11], proposing alternative one-pass decoding strategies such as on-the-fly rescoring [12], improving CPU-GPU communications [13] and, more recently, combining Gated Recurrent Units with more efficient objective functions, such as Noise Contrastive Estimation [14]. On the other hand, and certainly different from these contributions, other authors have explored the idea of converting Neural LMs, either recurrent or not, into n-gram models that can thus be smoothly integrated into the conventional recognition pipeline [15, 16].

This work follows the idea of directly using Neural LMs in decoding and, as in the pioneering work by Shi et al. [10], we advocate the use of the one-pass decoding strategy instead of the conventional two-step approach. It is worth noting, however, that significant progress has been made in ASR since the publication of this pioneering work, and thus the work reported here, based on current ASR technology, differs greatly from it. Generally speaking, we propose the direct use of LSTM-based LMs during one-pass decoding based on a History Conditioned Search (HCS) strategy [17]. To alleviate the computational cost entailed by the use of LSTM-LMs, three main ideas are exploited: static look-ahead tables; accelerated LSTM-LM computation by variance regularization and lazy evaluation; and two new pruning techniques. Results are reported on two standard tasks showing that these ideas are really useful for real-time one-pass decoding using LSTM-LMs.

2. One-pass decoder architecture

As previously mentioned, we propose the direct use of LSTM-LMs in one-pass HCS-based decoding. This proposal derives from the large capacity LSTM-LMs have, in contrast to n-gram LMs, to deal with histories of unlimited length [4]. This makes HCS-based decoding perfectly suited for its use with LSTM-LMs, as HCS decoders group hypotheses by history, and thus large decoding sub-networks can be safely removed during search, thereby lowering memory requirements.

Although the HCS approach allows us to deal with potentially infinite LM histories during decoding, LSTM-LMs present other challenges which need to be overcome in order to get a real-time decoder. The most important one is the high computational cost required by these models. In particular, the calculation of the Softmax layer is very expensive for large vocabularies. In what follows we describe the solutions we implemented in our decoder to the problems we encountered.

2.1. Static look-ahead tables

LM look-ahead is a well-known and widespread pruning technique. Basically, this technique consists of adjusting the LM score for each hypothesis h and time t by also taking into account every possible word w to follow [17]. In terms of computational cost, it requires a separate set of look-ahead scores, often referred to as the look-ahead table, for each new history h; that is, it requires the computation of $p(w \mid h)$ for each history

h and word w. Moreover, the cost of this computation, which is already high for conventional n-gram LMs, is even exacerbated when LSTM-LMs are used instead.

A common technique for HCS decoders to keep the look-ahead complexity at a reasonable level is to build look-ahead tables from simplified LMs; that is, if an n-gram LM is being used as the "big" reference LM, look-ahead tables are built from m-gram LMs with $m < n$. The only exception is that, whenever a word-end node is reached, the look-ahead score is replaced by the probability given by the big LM. This way, the number of different look-ahead tables and queries to the big LM are greatly reduced, which is particularly convenient for our LSTM-based one-pass decoder. That is, for fast computation of look-ahead tables, we propose the use of n-gram LMs.

Although the above trick for fast computation of look-ahead tables is really effective, it is worth noting that it can be refined even further, in a straightforward manner. To this end, consider, as we do here, the use of a small n-gram LM such as a pruned 4-gram LM. Then, all look-ahead tables can be precalculated before the actual recognition process begins, and thus the look-ahead complexity during decoding becomes negligible. This is done here, and in order to fit all look-ahead tables in memory, we use an approach similar to the Sparse LM Look-Ahead strategy described in [17].

2.2. Variance regularization and lazy evaluation

As commented before, one of the main drawbacks of using LSTM-LMs is the high computational cost of the Softmax layer. This high cost is mainly due to the estimation of the normalization term. Following the idea posed in [10], including a Variance Regularization (VR) term reduces drastically this computation. In this technique the normalization term is approximated by a constant. Therefore, the probability of a word can be approximated as

$$p(w \mid h) = \frac{\exp(v_L(h)^T \cdot a_w)}{Z(h)} \approx \frac{\exp(v_L(h)^T \cdot a_w)}{D}\,, \quad (1)$$

where h denotes the current history, L is the number of hidden layers, $v_L(h)$ and $Z(h)$ are respectively the input vector to the Softmax layer and the normalization term related to h, a_w is the weight vector for word w, and D is a constant.

In order to speed up the evaluation of the LSTM-LM during decoding, the models were trained based on the VR technique in conjunction with a lazy evaluation strategy in the decoding process. The basic idea behind the VR technique is to avoid computing the full Softmax, while the lazy strategy delays the evaluation of LSTM-LM as much as possible. More precisely, during decoding each LSTM history h is represented as a tuple $h = (w, h', V)$, where w is the last word of h, $h' = (w', h'', V')$ is the state of the previous history (implemented as a pointer), and V is either $\emptyset$ (empty) or $v_1^L(h)$. The term $v_1^L(h)$ refers to the LSTM hidden state for h and can be calculated as $v_1^L(h) = \text{RNN}(w, v_1^L(h'))$. Therefore, during decoding each time a word-end node (w) is reached for a given history $h' = (w', h'', V')$ the following steps are executed:

1. If $V' = \emptyset$ then $V' = \text{RNN}(w', v_1^L(h''))$
2. Estimate $p(w \mid h') = \frac{\exp(v_L(h')^T \cdot a_w)}{D}$
3. Create new state as $h = (w, h', \emptyset)$

Using this approach new histories are created at negligible cost, since the forward step in the model is carried out only when a word-end node is reached for the first time. Once the hidden state is calculated, it is cached in the current state. In practice most of these new histories will be pruned before any hypothesis reaches a word-end node, saving a significant amount of computations. In addition, each time $p(w \mid h')$ is required, it is approximated using Eq. 1 that replaces the full Softmax calculation.

2.3. Novel pruning techniques

Apart from the conventional pruning methods, two new pruning techniques were implemented for the one-pass decoder. On the one hand, in some situations the lack of LM history recombination in conjunction with the histogram pruning (upper bound for the maximum number of active hypothesis at each time frame) resulted in a decrease of performance. More precisely, for some long sentences, most part of the active hypotheses were similar, except for some long term differences in the LM history. Thus, in this case, the decoder behaves similarly to a greedy decoder. In order to avoid this behaviour a LM history recombination (LMHR) parameter was introduced. More precisely, for a given N two different LM histories w_1^M and $\hat{w}_1^L$ are recombined if $w_{M-N+1}^M = \hat{w}_{L-N+1}^L$. This recombination forces the decoder to consolidate word prefixes, and thus, it focuses on the current time frame. It is worth noting that the history length for the LSTM-LM is not limited. Each hypothesis still keeps a reference to the real LSTM state. A similar technique was introduced in [11], although the motivation was different.

Furthermore, the maximum number of new LM histories that can be created at each time frame is limited according to a parameter that we refer to as LM histogram pruning (LMHP).

2.4. Additional remarks

Although our approach is based on an HCS decoder, note that it could be reinterpreted as an on-the-fly composition as other authors have proposed in the past [18, 19]. Indeed, static look-ahead tables resembles the WFST approach, since the information stored in the set of look-ahead tables is similar to that represented in a WFST. Thus, the memory requirements are alike and the most important difference is that in the look-ahead tables the information is organized according to the LM histories.

Regarding this interpretation, this could be seen as performing the composition of a WFST (look-ahead tables) with a LSTM-LM. Nevertheless, it is important to remark that we are still using an HCS decoder. Thus, the structure of the small LM used for look-ahead is not introducing any kind of hypothesis recombination during the search. Its impact is limited to the look-ahead score computation.

Since our approach can be referred to as an on-the-fly composition of a small n-gram model and a big LM, it is possible to use other types of LMs rather than LSTMs. Indeed, an on-the-fly interpolation of n-gram and LSTM-LMs was implemented.

3. Experiments

3.1. Experimental settings

The proposed approach has been evaluated on the LibriSpeech ASR corpus [20], and the third version of the TED-LIUM corpus [21]. Statistics for these datasets are shown in Table 1. The provided vocabulary for LibriSpeech includes 200K words, while TED-LIUM's vocabulary comprises 153K. Regarding the partitions, we have used the **-other* for LibriSpeech and the **-legacy* ones for TED-LIUM.

Table 1: *Statistics of the corpora.*

	LibriSpeech		TED-LIUM	
	Dur.(h)	Words	Dur.(h)	Words
Train	961	884M	452	258M
Dev	5.3	50K	1.59	17K
Test	5.1	52K	2.61	27K

Our acoustic models were based on the hybrid approach [22]. First, we trained a context-dependent feed-forward DNN-HMM with three left-to-right states. State-tying schema follows a phonetic decision tree approach [23], resulting in 8.3K and 10.8K tied states for LibriSpeech and TED-LIUM respectively. We used the transLectures-UPV toolkit (TLK) to train both acoustic models [24].

The DNN-HMM model was then used to bootstrap a Bidirectional LSTM-HMM model [25], using TLK and TensorFlow [26]. The BLSTM network was composed of eight bidirectional hidden layers with 512 LSTM cells per layer and per direction. We limited the previous history to perform back propagation through time to a window size of 50 frames.

Regarding language modelling, we used n-gram and LSTM-LMs separately and in combination through linear interpolation. For LibriSpeech, we used the 4-gram ARPA LM (*fglarge*) that is provided with the dataset, while for TED-LIUM we trained a standard Kneser-Ney smoothed 4-gram LM model with the same data as it is indicated in [21] using SRILM [27]. The *OOV* ratio on dev sets was 0.57% and 0.17%, while was 0.54% and 0.09% on test sets, for LibriSpeech and TED-LIUM respectively. A pruned version for both models was used to estimate the static look-ahead tables.

LSTM-LMs were trained using Noise Contrastive Estimation (NCE) [28] and VR [10] criterion, in order to reduce the computational cost during both, training and test phases. Training was performed on GPU using the CUED-RNNLM toolkit [29]. We selected those models that provided the lowest perplexity on the dev sets: *dev-other* and *dev-legacy* for LibriSpeech and TED-LIUM, respectively. Both models consisted of a 256-unit embedding layer and a hidden LSTM layer of 1024 units. The output layer is a 200K units Softmax layer in the case of LibriSpeech, while in TED-LIUM the intersection between the provided vocabulary and words in the training set resulted in an output layer of 144K units. As our decoder can combine any number of models during decoding, an offline evaluation of the best linear interpolation of both n-gram and LSTMs using SRILM was performed. The interpolation weights were $w_{ngram} = 0.15, w_{lstm} = 0.85$ for LibriSpeech and $w_{ngram} = 0.22, w_{lstm} = 0.78$ for TED-LIUM.

Regarding the hardware setup, experiments were conducted on an Intel Xeon(R) CPU E5-1620@3.50GHz, and a GPU GTX1080Ti with 12GB. The estimation of the acoustic model scores was performed on GPU, while the estimation of the LM score and the rest of the decoding was carried out on CPU.

3.2. Experimental results

In this section, the impact of the LMHR parameter presented in Section 2.3 is evaluated. After this, a comparative experiment is shown using different LM during decoding. Then, the LMHP parameter is studied to assess the trade-off between Word Error Rate (WER) and Real-Time Factor (RTF). Finally, a comparison between one-pass and two-pass decoders is performed.

Figure 1 shows WER curves as a function of the LMHR parameter for the dev sets of LibriSpeech and TED-LIUM. As mentioned before, the LMHR controls the number of words that were kept during the decoding to perform hypothesis recombination. The LM employed in these figures is a linear interpolation between a large 4-gram LM and a VR-trained LSTM-LM described in Section 3.1.

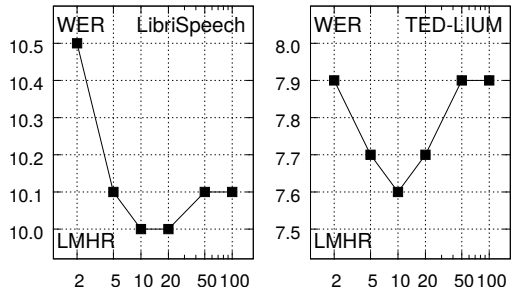

Figure 1: *Evaluation of the impact of the LMHR parameter in terms of WER [%] for LibriSpeech and TED-LIUM.*

As observed in both datasets, there is an optimum WER at around a history length of 10. This means a decrease of 0.5 and 0.3 WER points in LibriSpeech and TED-LIUM, respectively, w.r.t. a history length of 2. History lengths above 20 did not provide further improvements. In what remains, we have determined the optimal LMHR value for each dataset of 10.

Regarding the impact of using different LMs, Table 2 shows WER figures, relative improvement of WER ($\Delta\%$) and perplexity (PPL) for LibriSpeech and TED-LIUM test sets on systems that differ in their LM. From top to bottom, the baseline LM is the pruned (small) 4-gram LM used to perform look-ahead, then the large 4-gram LM, next the VR-trained LSTM-LM and finally the interpolated LM mentioned above. Decoding hyperparameters were tuned on the dev set.

As shown in Table 2, one-pass decoding systems including LSTM-LM present relative improvements of $\sim$17% for LibriSpeech and $\sim$13% for TED-LIUM, compared with large 4-gram-based systems. In addition, interpolated LM systems provide an additional $\sim$1% improvement over only-LSTM LM systems. In terms of perplexity w.r.t. large 4-gram LMs, interpolated LMs offer a significant reduction of $\sim$40% across datasets, stressing the consistent relation between perplexity and WER. This interpolated LM was the default model for posterior experiments.

Table 2: *Comparison of WER, relative improvement of WER w.r.t the baseline, and perplexity results using different LM models on LibriSpeech and TED-LIUM test partitions.*

	LibriSpeech			TED-LIUM		
	WER	$\Delta\%$	PPL	WER	$\Delta\%$	PPL
small 4-gram	14.4	-	222.1	10.4	-	176.4
+ 4-gram	12.3	14.6	146.2	9.6	7.7	148.7
+ LSTM	10.2	29.2	89.2	8.3	20.2	91.1
+ interp.	10.1	29.9	86.4	8.2	21.1	88.0

Figure 2 shows WER/RTF curves for LibriSpeech and TED-LIUM dev sets as a function of several selected values for the LMHP parameter (LMHP = 30, 60, 100, *unlimited*), varying the beam width. As remarked in Section 2.3, LMHP allows us to control the number of new LM histories that will be expanded, reducing the queries to the LSTM-LM.

As shown in Figure 2, the LMHP parameter has a strong impact in RTF, allowing us to adjust the WER/RTF trade-off on

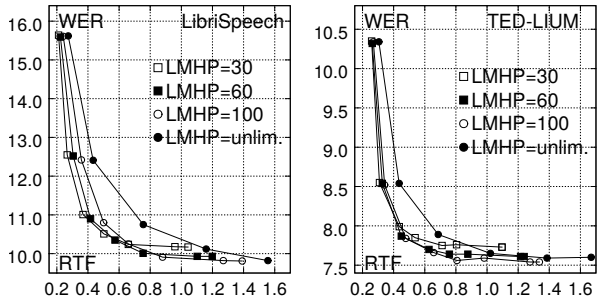

Figure 2: *Comparison of the impact of different values for the LM-histogram-pruning (LMHP) parameter in terms of WER and RTF for LibriSpeech and TED-LIUM.*

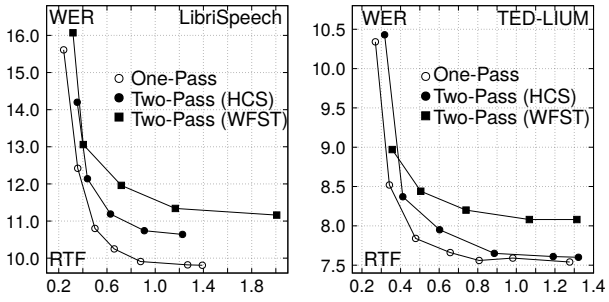

Figure 3: *Comparison of the one-pass HCS decoder and the two-pass HCS and WFST decoders in terms of WER and RTF for LibriSpeech (left) and TED-LIUM (right).*

demand. To this purpose, real-time performance (RTF$\sim$1) can be reached at almost no cost in terms of WER, as differences among LMHP values show. In particular, limiting LMHP to 100 allows us to obtain RTF results below 1 without WER reduction in both datasets.

Table 3 shows WER figures, RTF and relative increase of RTF ($\Delta\%$) on LibriSpeech and TED-LIUM test sets comparing the best performing LMHP value and *unlimited* using the same pruning parameters. As observed, RTF was drastically reduced while maintaining similar WER figures.

Table 3: *Comparison of WER, RTF and relative increase of RTF w.r.t to LMHP=100 on LibriSpeech and TED-LIUM test sets.*

	LibriSpeech			TED-LIUM		
LMHP	WER	RTF	$\Delta\%$	WER	RTF	$\Delta\%$
unlimited	10.12	1.56	-	8.18	1.38	-
100	10.20	0.88	43.6	8.19	0.87	37.0

As mentioned in Section 2, our decoder can be considered as an on-the-fly composition of the pruned LM and the LSTM/n-gram interpolation. This motivates another set of experiments that assesses the impact of performing this on-the-fly composition in contrast to the decoupled approach. For the decoupled approach, the standard two-pass decoding strategy was adopted based on generating lattices and then rescoring them with the LSTM/n-gram interpolated LM. For lattice generation two options were considered: using the pruned (small) n-gram model, already used for static look-ahead tables estimation, and the large LM. In the case of the small n-gram, a WFST decoder was used. In the second option, the large LM was used in our HCS decoder replacing the LSTM-LM to generate lattices. The posterior lattice rescoring was carried out in both cases using the CUED-RNNLM toolkit, which imports this function from HTK [30]. It is important to remark that this software was extended to include the VR normalization. The same acoustic and language models were used in both cases.

Figure 3 compares the WER/RTF curves for one-pass using LMHR=10 and LMHP=100 versus both two-pass approaches on LibriSpeech (left) and TED-LIUM (right) datasets. Results show that the one-pass decoder produces significant improvements in WER compared to WFST, especially when RTF is greater than 0.4. Considering a very similar RTF performance, the one-pass decoding approach achieves relative improvements in WER $\sim$12% and $\sim$6% in LibriSpeech and TED-LIUM, respectively. Comparing HCS decoders, one-pass shows a consistent improvement in RTF, reducing WER ($\sim$6%) in the case of LibriSpeech and obtaining a similar accuracy in TED-LIUM.

Table 4 shows WER figures for similar RTFs on test partitions, considering the aforementioned decoders, one-pass and two-pass HCS, and the two-pass WFST decoder. As shown in development sets, the one-pass HCS significantly improves over the WFST approach. Finally. the one-pass HCS obtains a better WER/RTF trade-off than the two-pass HCS.

Table 4: *Comparison of WER for similar RTFs between one-pass HCS decoder and the two-pass HCS and WFST decoders on LibriSpeech and TED-LIUM test partitions.*

	LibriSpeech		TED-LIUM	
	WER	RTF	WER	RTF
one-pass HCS	10.20	0.88	8.19	0.87
two-pass HCS	10.95	0.90	8.46	0.88
two-pass WFST	11.39	1.16	8.70	1.06

4. Conclusions and future work

A novel one-pass decoder that seamlessly combines the use of static look-ahead tables and LSTM-LMs has been presented. This decoder has been evaluated on reference ASR datasets, such as LibriSpeech and TED-LIUM, obtaining competitive WER/RTF results. Indeed, RTF figures are reported well below one that makes this decoder specially suitable for a real-time streaming setup.

Moreover, two new pruning parameters, LMHR and LMHP, were introduced allowing us to adjust the trade-off between WER and RTF according to our requirements. In addition, the one-pass and two-pass decoders were directly compared to demonstrate how the one-pass decoder clearly benefits from the integration of all LMs in the first stage of the decoding process.

As future work, the current decoder will be evaluated on a real streaming scenario providing recognition hypothesis as the audio signal data is ingested. In addition, the current LSTM-LM evaluation will be moved from CPU to GPU in order to further alleviate the computational cost.

5. Acknowledgements

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 761758 (X5gon) and the TIN2015-68326-R and RTI2018-094879-B-I00 (MINECO/MCIU/AEI/FEDER,UE) research projects, and the grant FPU14/03981 from the Spanish Ministry of Education, Culture and Sport.

6. References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[2] 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.

[3] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[4] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *Twelfth annual conference of the international speech communication association*, 2011.

[5] Y. Si, Q. Zhang, T. Li, J. Pan, and Y. Yan, "Prefix tree based n-best list re-scoring for recurrent neural network language model used in speech recognition system." in *Interspeech*, 2013, pp. 3419–3423.

[6] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[7] X. Chen, X. Liu, A. Ragni, Y. Wang, and M. J. Gales, "Future word contexts in neural network language models," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 97–103.

[8] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition," 2018.

[9] A. Ogawa, M. Delcroix, S. Karita, and T. Nakatani, "Rescoring n-best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6099–6103.

[10] Y. Shi, W.-Q. Zhang, M. Cai, and J. Liu, "Efficient one-pass decoding with nnlm for speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 377–381, 2014.

[11] Z. Huang, G. Zweig, and B. Dumoulin, "Cache based recurrent neural network language model inference for first pass speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6354–6358.

[12] T. Hori, Y. Kubo, and A. Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6364–6368.

[13] K. Lee, C. Park, I. Kim, N. Kim, and J. Lee, "Applying gpgpu to recurrent neural network language model based fast network search in the real-time lvcsr," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[14] K. Lee, C. Park, N. Kim, and J. Lee, "Accelerating recurrent neural network language model based online speech recognition system," *arXiv preprint arXiv:1801.09866*, 2018.

[15] E. Arısoy, S. F. Chen, B. Ramabhadran, and A. Sethy, "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 184–192, 2014.

[16] M. Singh, Y. Oualil, and D. Klakow, "Approximated and domain-adapted lstm language models for first-pass decoding in speech recognition." in *INTERSPEECH*, 2017, pp. 2720–2724.

[17] D. Nolden, "Progress in decoding for large vocabulary continuous speech recognition," Ph.D. dissertation, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, Apr. 2017.

[18] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[19] H. Sak, M. Saraclar, and T. Güngör, "On-the-fly lattice rescoring for real-time automatic speech recognition," in *Eleventh annual conference of the international speech communication association*, 2010.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[21] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Estève, "Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation," in *International Conference on Speech and Computer*. Springer, 2018, pp. 198–208.

[22] H. Bourlard and C. J. Wellekens, "Links between Markov models and multilayer perceptrons," in *Advances in Neural Information Processing Systems I*, D. Touretzky, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1989, pp. 502–510.

[23] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proc. Workshop on Human Language Technology*, Plainsboro, NJ, USA, Mar. 1994, pp. 307–312.

[24] M. del Agua, A. Giménez, N. Serrano, J. Andrés-Ferrer, J. Civera, A. Sanchis, and A. Juan, "The translectures-UPV toolkit," in *Advances in Speech and Language Technologies for Iberian Languages*, Nov. 2014, pp. 269–278.

[25] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2462–2466.

[26] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[27] A. Stolcke, "SRILM – an extensible language modeling toolkit," Denver, CO, USA, Sep. 2002, pp. 901–904.

[28] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," *arXiv preprint arXiv:1206.6426*, 2012.

[29] X. Chen, X. Liu, Y. Qian, M. J. F. Gales, and P. C. Woodland, "CUED-RNNLM – An open-source toolkit for efficient training and evaluation of recurrent neural network language models," Mar. 2016, pp. 6000–6004.

[30] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey *et al.*, "The htk book," *Cambridge university engineering department*, vol. 3, p. 175, 2002.