Word-Phrase-Entity Recurrent Neural Networks for Language Modeling

Michael Levit, Sarangarajan Parthasarathy, Shuangyu Chang

Microsoft Corporation, U.S.A.

{mlevit|sarangp|shchang}@microsoft.com

Abstract

The recently introduced framework of Word-Phrase-Entity language modeling is applied to Recurrent Neural Networks and leads to similar improvements as reported for n-gram language models. In the proposed architecture, RNN LMs do not operate in terms of lexical items (words), but consume sequences of tokens that could be words, word phrases or classes such as named entities, with the optimal representation for a particular input sentence determined in an iterative manner. We show how auxiliary techniques previously described for n-gram WPE language models, such as token-level interpolation and personalization, can also be realized with recurrent networks and lead to similar perplexity improvements.

Index Terms: RNN, WPE and class-based LM, LM interpolation, LM personalization

1. Introduction

During recent years, the field of Language Modeling has been undergoing a gradual though steady shift away from n-gram models towards more powerful feature-based, continuous-state models such as Exponential and Neural Network LMs [1, 2, 3]. While most industrial-scale ASR systems are still relying on very large n-gram LMs due to their training simplicity, continuous-state LMs do offer better modeling quality given the same amount of data [4, 5, 6].

One important class of language models that received wide acceptance are Recurrent Neural Networks (RNN) [3, 7], which combine the advantage of longer history contexts with the ability to deal internally with word similarity. RNNs have been successfully used in many applications and a number of open-source toolkits are available for experimentation [8, 9, 10].

Another type of language models that achieves similar goals while still relying on n-grams has been introduced recently. Word-Phrase-Entity (WPE) language models offer a unified probabilistic framework for joint modeling of words, common word phrases (e.g. *"i'd+like+to"*) and classes (such as named entities) [11]. For a training sentence, the WPE LM training algorithm will consider a number of its alternative representations (parses) in terms of *tokens*: words, phrases and entities, and then pick the one(s) with the highest likelihood. For example, the sentence *"flight from boston to new york at three p m"* can be represented as sequence of words (trivial parse), or it could include common word phrases *"flight from+boston to+new+york at three p+m"*, or named entities *"flight from* $CITY_{[\text{boston}]}$ *to* $CITY_{[\text{new york}]}$ *at* $TIME_{[\text{three p m}]}$*"*, or various permutations thereof. Which alternative will have the highest likelihood depends on stable word patterns in this and other sentences in the training set, as data parsing and model re-estimation are carried out iteratively [11].

In this work, we unify both modeling paradigms and offer recipes to train Word-Phrase-Entity Recurrent Neural Network Language Models (WPE RNN). As with the n-gram WPE LMs, training happens in an iterative fashion and can be accompanied by optimization of the grammars that define relevant entities. We show that, depending on scenario, WPE n-gram LMs can achieve improvements comparable to word-level RNNs, while WPE RNNs consistently lead to the best results in terms of perplexity and word error rates. In addition, we demonstrate that other WPE modeling techniques such as personalization [13] and (context-dependent) token-level LM interpolation [12] can also be successfully applied to WPE RNN, while the latter can be carried out on an arbitrary mixture of n-gram and RNN language models.

The remainder of this paper is organized as follows. In Section 2 we review the concept of WPE LM and show how it can be transferred to RNN. Section 3 explains personalization and interpolation for WPE RNNs, and Section 4 provides additional insights into our parsing framework. We then present our experiments in Section 5 and conclude with a summary and future work suggestions.

2. WPE RNN Language Models

WPE language models are trained via an alternating sequence of data parses and LM re-estimations. Given a pre-trained WPE that provides language model probabilities for token-level histories, we can produce a number of alternative parses with corresponding posterior probabilities. A plurality of such parses obtained from a training corpus can be used to estimate a new WPE LM to continue the iterative training. Several heuristics such as parameter regularization and LM initialization are required to achieve optimal modeling power [11]. Formally, the updated probability $\bar{P}(c|h)$ of token c given token history h is computed as:

$$\bar{P}(c|h) := \sum_{\boldsymbol{w},\boldsymbol{c}} P(\boldsymbol{w}) \frac{P(\boldsymbol{w},\boldsymbol{c})}{\sum_{\boldsymbol{c}'} P(\boldsymbol{w},\boldsymbol{c}')} P^{(\boldsymbol{c})}(c|h), \qquad (1)$$

where the outer summation is over training sentences $\boldsymbol{w}$ and their parses $\boldsymbol{c}$. The joint probability of a sentence and one of its parses under the previous WPE LM is:

$$P(\boldsymbol{w},\boldsymbol{c}) = \prod_{c_i \in \boldsymbol{c}} P(c_i|h_i) P(\pi_i|c_i), \qquad (2)$$

where $\boldsymbol{\pi} = (\pi_1 \dots \pi_n)$ is the segmentation of $\boldsymbol{w}$ induced by $\boldsymbol{c} = (c_1 \dots c_n)$ and probabilities $P(\pi_i|c_i)$ are either class-instance probabilities for class tokens (such as named entities) or 1.0 for words and word phrases. Sentence priors $P(\boldsymbol{w})$ are usually explicitly provided or can be ignored altogether. Finally, $P^{(\boldsymbol{c})}(c|h)$ stands for an estimator of $P(c|h)$ based on a single linear token sequence $\boldsymbol{c}$. In the case of n-gram LMs, its value can be obtained directly as a maximum likelihood estimate $\frac{\#ch}{\#h}\Big|_{(\boldsymbol{c})}$. However, a more practical interpretation of

Eq. (1) suggests training a token-level language model from a set of weighted parses of all training sentences with the weights

$$L'(\boldsymbol{w},\boldsymbol{c}) = P(\boldsymbol{w})\frac{P(\boldsymbol{w},\boldsymbol{c})}{\sum_{\boldsymbol{c}'} P(\boldsymbol{w},\boldsymbol{c}')}. \quad (3)$$

For n-gram LMs, this interpretation buys the benefit of smoothing (e.g. back-offs) but, more importantly, it also implies that computation of estimates $P(c|h)$ can be delegated to arbitrary language model types, including RNNs that we are focusing on in this paper.

Before we proceed, let us recap how RNNs operate. A typical (Elman) RNN is a network with three layers: normalized activations of the output layer approximate probabilities of the next word given history. The relatively small hidden layer underneath reflects the network's current state. It is fed from a concatenation of the input layer (a one-hot vector of input words) and the network state from the previous time step. The network is trained via Back Propagation Through Time. Since brute force training and evaluation require the computation of activations for all nodes of the output layer, the architecture can be optimized by organizing the output layer as a hierarchy of (possibly sub-optimal) word classes. Other popular ways to speed up last layer computations include Noise Contrastive Estimation [14] and Importance Sampling [15]. In [19], an extension of RNN is proposed that allows for additional features in the input layer (such as identities of the preceding words) and direct connections between the added nodes and nodes in the output layer.

In the sections below, the following nomenclature will be used for brevity: "W-N-LM" for word n-gram language model; "T-N-LM" for WPE n-gram LM ("T" as in "*T*oken"); "W-R-LM" for word RNN LM and "T-R-LM" for WPE RNN. In addition, subscript "$_\text{p}$" will indicate personalized nature of the model (e.g. T-R-LM$_\text{p}$). T-N-LM and T-R-LM should be both viewed as combinations of token-level language models and the corresponding sets of entity defining grammars.

Assume now that the latest T-R-LM produced a number of parses (token sequences) for each training sentence. To train a new iteration T-R-LM according to Eq. (1), we need to introduce weights $L'(\boldsymbol{w},\boldsymbol{c})$ in the RNN training procedure. Unlike n-grams, we may not pool occurrence counts for identical training samples for RNN training, and need to present each sentence individually, simplifying $L'(\boldsymbol{w},\boldsymbol{c})$ into parse posteriors with the upper bound of $L'(\boldsymbol{w},\boldsymbol{c}) \leq 1$. Next, we scale the cross-entropy error function by $L'(\boldsymbol{w},\boldsymbol{c})$ before computing gradient in the back-propagation algorithm. Randomizing training samples on each iteration (and in fact even each RNN training epoch) produced better results than sorting all parses by their length as a proxy for their difficulty [17], probably because it helped avoiding repetition of partially identical parses [16].

The training diagram is shown in Figure 1. Just like the T-N-LMs, T-R-LMs are trained in an iterative fashion by parsing the training set, re-estimating the model from these (weighted) parses, producing new parses of the training set, etc. Unlike T-N-LMs, the issue of initialization is now easier to address since the first T-R-LM can be trained from parses of the training data with the final T-N-LM. Also note that each iteration of T-R-LM training is also an iterative process of several epochs. The overall training costs are therefore substantially higher than with T-N-LM. Finally, similarly to T-N-LMs, entity definitions can be refined on each iteration following this formula:

$$P(\pi|c) = \sum_{\boldsymbol{w},\boldsymbol{c}} L'(\boldsymbol{w},\boldsymbol{c})\left.\frac{\#(c,\pi)}{\#c}\right|_{(\boldsymbol{c})} \quad (4)$$

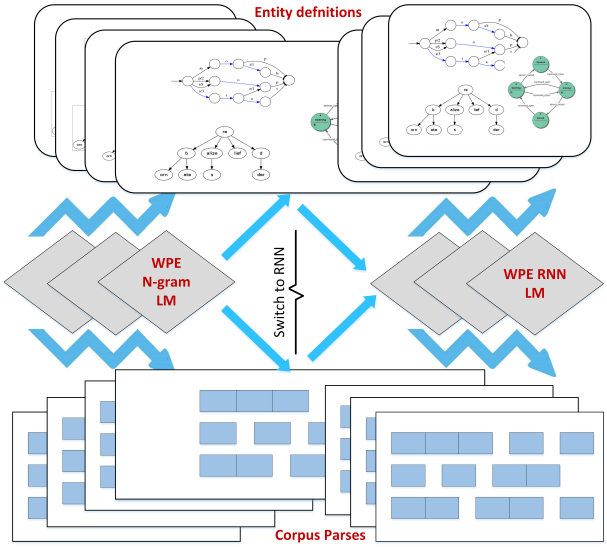

Figure 1: *Training of T-R-LM. We start by iteratively training T-N-LM to convergence. The set of final parses and entity definitions are used to start T-R-LM training that comprises alternating RNN training and parsing.*

3. Language Model Interpolation and Personalization

In [12] it was shown how several class-based n-gram LMs, and T-N-LMs in particular, can be interpolated using context-dependent interpolation weights $\lambda_m(h)$. The basic idea is to update weight of the mth component in token context h as the normalized cumulative of the relative contributions of this component to probabilities of all training parses that contain h. Parse posteriors are used to weight these contributions. Formally, if we express probability of token c given token history h as a linear interpolation of probabilities in the component LMs:

$$P(c|h) = \sum_m \lambda_m(h) P_m(c|h), \quad (5)$$

The context-dependent interpolation weights $\lambda(h)$ can be iteratively updated via:

$$\bar{\lambda}_m(h) := -\frac{1}{\gamma(h)} \sum_{\boldsymbol{w},\boldsymbol{c}} \left(L'(\boldsymbol{w},\boldsymbol{c}) \sum_{c:h(c)=h} S_m(c,h) \right) \quad (6)$$

with sufficient statistics

$$S_m(c,h) = \frac{P_m(c|h)\lambda_m(h)}{\sum_{m'} P_{m'}(c|h)\lambda_{m'}(h)} \quad (7)$$

and normalization factor $\gamma(h)$. Summation index $h(c) = h$ means that we only consider those tokens in parse $\boldsymbol{c}$ whose history is h. For more information, see [12].

Observe that the above formulations do not make any assumptions about the nature of the participating LM components, and can therefore be applied to T-R-LMs and T-N-LMs alike and in any combination. It is worth noticing, however, that context for interpolation weights can be different from context for language model probabilities, and could in fact end up being the only explicit fixed-length context if only T-R-LMs (and W-R-LMs) are interpolated.

As training progresses, some class definitions can be updated using Eq. (4) while others can be defined on a per-user basis. We call this scenario personalized WPE training [13]. For user u, this amounts to turning Eq. (2) into

$$P(\boldsymbol{w}, \boldsymbol{c}, u) = \prod_{c_i \in \boldsymbol{c}} P(c_i|h_i)P(\pi_i|c_i, u) \quad (8)$$

and then adjusting $L'(\boldsymbol{w}, \boldsymbol{c}, u)$ accordingly. Again, these modifications can be used with T-R-LM$_\text{p}$ in a similar fashion as they were used with T-N-LM$_\text{p}$.

4. Parser Implementation

In this section, we provide more insights into how our parser is implemented. At each point in time, our trellis parser maintains a number of competing states, and decoding is done using Dynamic Programming algorithm. For WPE language models, such a "state" incorporates the state of the token-level language model (n-gram or RNN) and, for non-trivial tokens, also current word position within the active token (defined as a word trie or FSM). Some auxiliary bookkeeping information needs to be included as well. For n-grams, the state in a token-level LM is given by the n-gram token history. For RNNs, we store the activations of the hidden layer. If direct connections are used [19] (we found them to be crucial for modeling success), the identities of the preceding tokens are also saved[1]. In this setup, consuming the next word of an input sentence means either advancing within a token definition, or switching to the next token, or both. For a typical RNN implementation this approach calls for separation between RNN advancement step (read the next token and update the hidden state as well as features for direct connections) and evaluation step (given hidden state activations and features, compute token output probabilities). Furthermore, we found it useful to make a number of other adjustments such as separating begin- and end-of-sentence symbols. This formulation makes parser LM-agnostic and available for any types of language models, including interpolations.

5. Experiments and Results

5.1. Domain, Data and Tools

Like our previously reported experiments, the latest investigations are centered around audio interactions of Windows Phone users with an automated conversational agent (Cortana) and cover a wide variety of topics and styles, such as voice search, command-and-control, chit-chat, message dictation and others. All class definitions remain unchanged since our last investigations and include 21 shared named entities such as ACTOR, TIME etc. and two personalizable entities CONTACT_NAME and PLACE [13]. However, for various reasons including user privacy concerns, new data sets had to be assembled. Transitioning to case-sensitive language models is another difference from our previous experiments. There are two scenarios that we are focusing on:

- C1: command-and-control and message dictation
- C2: voice search and chit-chat (complement of C1).

The separation is justified by different interaction styles and personalization opportunities (medium in C1, low in C2). Table 1 shows separation of the corresponding corpora into training, validation and test sets. We train our language models on

[1]In reality, trellis states are expressed as integer arrays, so certain type conversions are necessary.

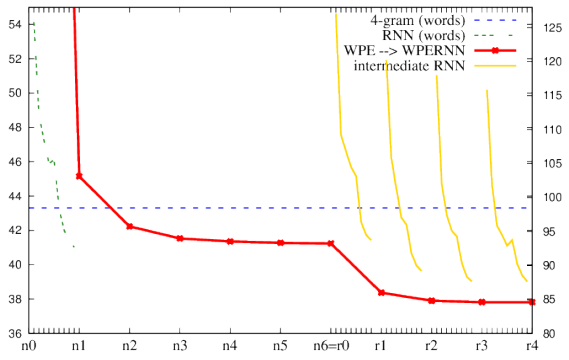

Figure 2: *Perplexity on* C2 *vali1 set for 4-gram W-N-LM, W-R-LM, T-N-LM (iterations n1–n6) and subsequently trained T-R-LM (r1–r4); perplexity numbers of the intermediate token-level RNNs trained on alternative parses are also included on the right y-axis.*

TRAIN, employ perplexity change on VALI1 as a stopping criterion, optimize interpolation weights on VALI2 and measure final perplexity/WER numbers on TEST-TEXT and TEST-AUDIO respectively. The average sentence length is close to five words.

Table 1: *Sizes of data sets for the covered scenarios.*

scenario	TRAIN	VALI1	VALI2	TEST-TEXT	TEST-AUDIO
C1	360K	25K	23K	10K	2.4K
C2	1M	45K	45K	10K	4K

To train n-gram language models we use SRILM toolkit [18]; we also borrow the core trellis decoder from it. Our RNN implementation is based on [8] with modifications as outlined in Section 4. We use 4-gram token history for direct connections to the output layer, the number of classes and size of hidden layer is set to 100 as we saw that larger networks did not provide any significant improvement.

5.2. Experiments

As illustrated in Figure 1, the first stage of our training process is estimating a T-N-LM. The observed improvements due to this stage are in-line with our previous reports (about 5% relative perplexity reduction). Once T-N-LM convergence has been reached, we continue iterations, but instead of token-level n-gram LMs, token-level RNN LMs are trained and used to generate parsing alternatives for each training sentence. The process continues till convergence. Figure 2 shows how this process changes perplexity on the validation set VALI1 in C2. It takes T-N-LM training six iterations (n1–n6) to converge, and the consecutive T-R-LM training takes another 4 iterations (r1–r4) yielding additional significant improvements. The token-level perplexity results along the right y-axis illustrate token-level RNN training progress for consecutive epochs within each WPE iteration. These numbers are not directly comparable to the word-level perplexity because they do not take into account probabilities of particular token realizations, but also because a single token replaces many words. Even though in the present setup entities do not impact the OOV rate significantly, in what follows, we set P(OOV)=1e-7 to enable direct perplexity com-

parison. The final perplexities on the test sets for both scenarios are shown[2] in Table 2. The table shows that while W-R-LM outperforms 4-gram W-N-LM, it only beats 4-gram T-N-LM in the C2 scenario, but not in C1 that is heavy on named entities such as TIME and (in the absence of personalized grammars) FIRSTNAME. Furthermore, combining the two techniques leads to the best perplexity numbers for both scenarios (10-15% relative to W-N-LM and 4-8% relative to W-R-LM).

Table 2: *Perplexities of different language models on the test sets.*

scenario	W-N-LM	T-N-LM	W-R-LM	T-R-LM
C1	69.7	65.9	62.0	**59.3**
C2	42.6	40.6	41.0	**38.0**

For speech recognition experiments, W-N-LMs can be used just like regular class-based language models [11], but rescoring is required when RNNs get involved. For each scenario/WPE combination, we therefore run first pass recognition with the corresponding n-gram LM (W-N-LM or T-N-LM) and then rescore the resulting 10-best hypotheses, sending each of them through the WPE parser to compute lattice probabilities for all alternative parses of this hypothesis and substituting the first pass language model probability with it. The oracle WER on 10-best is measured at about half the corresponding 1-best measure. Table 3 shows the obtained WERs for both scenarios. We suspect that the relatively modest WER improvements due to WPE for both n-grams and RNN (≈2%) are caused, in part, by the somewhat non-random nature of the pre-existing set TEST-AUDIO and transcriber bias whose effect we described in [20].

Table 3: *Word error rates (%) on the test sets.*

scenario	W-N-LM	T-N-LM	W-R-LM	T-R-LM
C1	15.10	14.76	13.96	**13.59**
C2	11.58	11.21	10.58	**10.38**

Next, for the C1 scenario, Table 4 summarizes the effect of entity personalization as described in Section 3. It shows that two additional entities CONTACT_NAME and PLACE defined on a per-sentence basis, can lead to significant perplexity improvements (7-9% relative), even though just about 10% of all test sentences actually had a matching entry in their personalized grammars. The corresponding WERs for the recognition experiments drop to 14.5% and 13.3% for n-grams and RNNs respectively.

Table 4: *Personalization improves perplexities for RNN WPE LMs, just like it does for n-gram WPE LMs.*

personalization	n-grams	RNN
no	T-N-LM⇒65.9	T-R-LM⇒59.3
yes	T-N-LM$_p$ ⇒**61.2**	T-R-LM$_p$ ⇒**53.9**

To measure effect of language model interpolation from Section 3, we separately consider word-level, WPE and personalized WPE setups for both scenarios. Table 5 summarizes these results. It shows that context-dependent token-level interpolation of n-gram and RNN language models achieves between 6% and 11% improvement relative to RNN language models alone, and up to 17% improvement relative to n-grams. WPE language models benefit from the interpolation just as much as the word-level LMs do, and the same holds for personalized scenarios.

Table 5: *Token-level interpolation of n-gram and RNN language models.*

	C1			C2	
LM	words	WPE	WPE-pers.	words	WPE
n-gram	69.7	65.9	61.2	42.6	40.6
RNN	62.0	59.3	53.9	41.0	38.0
interp.	57.9	54.8	**50.7**	36.5	**34.7**

On average, the estimated optimal weights assigned to the RNN ended up between 0.65 (word-level) and 0.75 (WPE). We discovered that the space of interpolation weights for RNNs is fairly flat and, on unseen data, the optimized weights contribute no more than additional 1% in comparison to the context-independent and equal weights. We believe that it is RNN's inherent ability to impose similarity metric on words that renders context-dependent interpolation weights superfluous. Still, additional small improvements could be achieved by interpolating all four types of language models (W-N-LM, W-R-LM, T-N-LM and T-R-LM).

In terms of word error rates, we saw no consistent improvement due to interpolation w.r.t. rescoring with RNNs alone, despite strong indications to the contrary from the perplexity numbers. The WER improvements were on the order of at most 1%. We then ran a parameter sweep for sentence-level interpolation and saw an almost monotonic increase in WER as more probability mass was moved from RNNs to n-grams. Having investigated connections between changes in language model score and changes in WER as we transition from RNN LMs to the corresponding interpolated versions, we have noticed that there was a large number of sentences that experienced a substantial increase in LM score due to interpolation while the n-best order remained unaffected. On the other hand, the range of small LM-score changes had no visible correlation with WER moves. We did see, however, that small improvements could be achieved when rescoring was accompanied by increase in the language model scale (relative to acoustic model's contributions).

6. Future Work and Conclusions

We have demonstrated that the Word-Phrase-Entity language modeling technique can be successfully applied to other types of language models, such as popular Recurrent Language Models. The RNNs turned out to benefit from the joint modeling of words phrases and entities, just as n-grams did, achieving up to 7% perplexity improvements. We also ported two other previously introduced WPE techniques to RNNs: personalization and token-level interpolation. On the corpus with personal entity definitions another 9% reduction in perplexity was achieved, and token-level context-dependent interpolation accounted for an additional 6-8% drop. The improvements were accompanied by modest decreases in word error rates, except for token-level interpolation where no consistent sizable improvements were observed. Our next goals are to scale the experiments to larger data sets and to extend the concept of classes to data-driven word clusters and thus obtain much higher coverage compared to the opportunistically defined entities.

[2]The relatively low perplexity of C2 is due to a high proportion of chit-chat.

7. References

[1] Chen S.: *Performance prediction for exponential language models*, in NAACL-HLT, 2009.

[2] Bengio, Y., Ducharme, R. and Vincent, P.: *A Neural Probabilistic Language Model*; in Journal of Machine Learning Research, No. 3, pp.1137–1155, 2003.

[3] Mikolov, T., Karafiát, M., Burget, L., Černocký, J. and Khudanpur, S.: *"Recurrent Neural Network Based Language Model"*; in proc. of Interspeech, Chiba, Japan, 2010.

[4] Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P. and Robinson, T.: *"One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling"*; arXiv preprint arXiv:1312.3005, 2013.

[5] Kim, Y., Jernite, Y, Sontag, D., and Rush, A. M. *"Character-aware Neural Language Models"*; arXiv preprint arXiv:1508.06615, 2015.

[6] Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: *"Exploring the Limits of Language Modeling"*; in arXiv:1602.02410, 2016.

[7] Mikolov, T., Kombrink, S., Burget, L, Černocký, J, Khudanpur, S.: *"Extensions of Recurrent Neural Network Language Model"*; in proc. of ICASSP, Prague, Czech Republic, 2011.

[8] Mikolov, T., Kombrink, S., Deoras, A., Burget, L, Černocký, J.: *"RNNLM – Recurrent Neural Network Language Modeling Toolkit"*, in ASRU, Hawaii, 2011.

[9] Chen, X., Liu, X., Qian, Y., Gales, M. and Woodland, P.: *"CUED-RNNLM – An Open-Source Toolkit for Efficient Training and Evaluation of Recurrent Neural Network Language Models*; in proc. of ICASSP, Shanghai, China, 2016.

[10] *"Faster RNNLM Toolkit"*; available from *https://github.com/yandex/faster-rnnlm*.

[11] Levit, M., Parthasarathy, S., Chang, S., Stolcke, A. and Dumoulin, B.: *"Word-Phrase-Entity Language Models: Getting More Mileage out of N-grams"*; in proc. of Interspeech, Singapore, 2014.

[12] Levit, M., Stolcke, A., Chang, S., Parthasarathy, S.: *"Token-Level Interpolation for Class-Based Language Models"*; in proc. of ICASSP, Brisbane, Australia, 2015.

[13] Levit, M., Stolcke, A., Subba, R., Parthasarathy, S., Chang, S., Xie, S., Anastasakos, T. and Dumoulin, B.: *"Personalization of Word-Phrase-Entity Language Models*; in proc. of Interspeech, Dresden, Germany, 2015.

[14] Chen, X., Liu, X., Gales, M., Woodland, P.: *"Recurrent Neural Network Language Model Training with Noise Contrastive Estimation for Speech Recognition"*; in proc. of ICASSP, Brisbane, Australia, 2015.

[15] Bengio, Y. and Senécal, J.-S.: *Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model*; in IEEE Transactions on Neural Networks, 19(4), pp.713-722, 2008.

[16] Yu, D. and Deng, L.: *"Automatic Speech Recognition – A Deep Learning Approach"*; pp.72, Springer, 2015.

[17] Bengio, Y., Louradour, J., Collobert, R. and Weston, J.: *"Curriculum Learning"*; in proc. of ICML, 2009.

[18] Stolcke, A.: *"SRILM an Extensible Language Modeling Toolkit"*; in proc. of Interspeech, 2002.

[19] Mikolov, T., Deoras, A., Povey, D., Burget, L, Černocký, J.: *"Strategies for Training Large Scale Neural Network Language Models*; in proc. of ASRU, Hawaii, 2011.

[20] Levit, M. Chang, S., Alonso, O. and Yadavalli, A. K.: *"End-to-End Crowdsourcing Approach for Unbiased High-Quality Transcription of Speech*; in proc. of HCOMP, AAAI, San Diego, 2015.