Recurrent Neural Network Based Language Model Personalization by Social Network Crowdsourcing

Tsung-Hsien Wen[1]*,Aaron Heidel*[1]*, Hung-yi Lee*[2]*, Yu Tsao*[2]*, and Lin-Shan Lee*[1]

[1]National Taiwan University,
[2]Academic Sinica, Taipei, Taiwan

r00921033@ntu.edu.tw, lslee@gate.sinica.edu.tw

Abstract

Speech recognition has become an important feature in smartphones in recent years. Different from traditional automatic speech recognition, the speech recognition on smartphones can take advantage of personalized language models to model the linguistic patterns and wording habits of a particular smartphone owner better. Owing to the popularity of social networks in recent years, personal texts and messages are no longer inaccessible. However, data sparseness is still an unsolved problem. In this paper, we propose a three-step adaptation approach to personalize recurrent neural network language models (RNNLMs). We believe that its capability to model word histories as distributed representations of arbitrary length can help mitigate the data sparseness problem. Furthermore, we also propose additional user-oriented features to empower the RNNLMs with stronger capabilities for personalization. The experiments on a Facebook dataset showed that the proposed method not only drastically reduced the model perplexity in preliminary experiments, but also moderately reduced the word error rate in n-best rescoring tests.

Index Terms: Recurrent Neural Network, Personalized Language Modeling, Social Network, LM adaptation

1. Introduction

Personalization, which is an indispensable aspect in several real world applications nowadays, has been studied in a wide range of research fields, such as personalized web search [1, 2] and recommendation systems [3, 4, 5, 6]. In the speech research community, acoustic model (AM) adaptation [7, 8, 9], which has been proven to provide an impressive word error rate reduction, can be viewed as one of the major efforts made for personalization in this field. However, although the personalized AM has been well studied, there is little work done on language models (LM) personalization. Traditionally, LM adaptation [10, 11, 12] stressed the problem of cross-domain linguistic mismatch, in which the cross-individual linguistic mismatch is often ignored. Likely due to the lack of sufficient personal corpora at the time, it was difficult to realize the idea. Therefore, we have to aggregate the corpora produced by many different individuals but on similar domains to perform a domain-oriented LM adaptation. There are two major trends that strongly push the progress of LM personalization. First, owing to the mass production and rapid proliferation of smartphones in recent years, each individual has the capability of possessing a smartphone, which makes its use a very personal experience [13, 14, 15]. Therefore, only a speaker-dependent recognizer is needed. Second, large quantities of texts are available over the social networks with known authors and given relations among the authors. It is possible to train personalized LMs because it may be reasonable to assume that users with close relationships may share common subject topics, wording habits, and linguistic patterns.

N-gram-based LMs [10, 16, 17], including various adaptation techniques, have been proven to work very well for many applications. In our previous work, we adapted an N-gram-based LM toward one particular user's wording patterns by incorporating the social texts that the target user and other users had posted considering different aspects of similarities between users [18]. However, the lack of natural strategy to model long-range dependencies [19, 20, 21], and the potentially error-prone back-off behaviors [22] seem to limit the prediction power of N-gram models. Recently, several studies have shown that neural-network-based LMs (NNLM) [23, 24, 25] are an improvement over the N-gram-based models by taking advantage of the learned distributed representations of word histories. Moreover, RNNLMs [26, 27, 28], which memorizes arbitrary length of histories in a recurrent structure, can model long range dependencies in an elegant way. Considering our LM personalization task, we think that it is beneficial to adopt RNNLMs for the following three reasons: (1) the number of social posts we can obtain from each individual are still relatively sparse compared to the number to which we can robustly apply an N-gram LM adaptation. However, the distributed representation of word histories can extenuate this problem by addressing the similarities of a combinatorial number of originally disjoint sentences in the new continuous space. (2) Online messages or social posts tend to be relatively casual, and thus, do not usually obey traditional grammar rules strongly. As a consequence, short-term dependencies may not be evident enough for predicting the next word anymore. The recurrent structure may help mitigate this issue since it memorizes longer dependencies than N-gram LMs. (3) Besides posts and messages, social networks also maintain a significant amount of information such as user profiles, relationships, interactions, preferences, and interests. These are all valuable information we can use while predicting one's linguistic habits. As mentioned in some previous works [28, 29], adding additional auxiliary features into RNNLMs is relatively easy and intuitive. As a result, we value the meet of LM personalization and RNNLMs as an indispensable attempt.

In this paper, we propose a new paradigm for personalized LMs where we use the RNNLMs as our base model. The system is built on a voice access of cloud application [18]. A crowdsourcing mechanism is designed to collect texts and other information on social networks from users. We also propose a three-step training for personalized RNNLMs considering the feasibility and complexity of the training process. Some user-oriented features were incorporated into the model for better word prediction capabilities. The experiments showed that the proposed method not only drastically reduced the model perplexity in preliminary experiments, but also moderately reduced the word error rate in n-best rescoring tests.

2. The Crowdsourcing Platform

In order to obtain personal acoustic and language data, we implemented a cloud-based application that can help users access their social network via voice, as shown in Fig. 1. Consequently, this cloud-based application can also be viewed as a crowdsourcing platform for collecting personal data. Crowdsourcing [30, 31] has been applied for several purposes in many different fields. For example, a crowdsourcing approach [32] was proposed to collect queries in an information retrieval system considering temporal information. The MIT movie browser [33, 34] relied on Amazon Mechanical Turk, the most famous crowdsourcing platform, to build a crowd-supervised spoken language system. The definition of crowdsourcing varies depending on the scenario. In our case, an implicit crowdsourcing [31] is at play: the user logs into his/her Facebook account and grants our application the authority to collect his/her acoustic and linguistic data for adaptation for our voice access service. At the same time, the user enjoys the benefits of better accuracy brought by the personalized recognizer that we build from the crawled data.

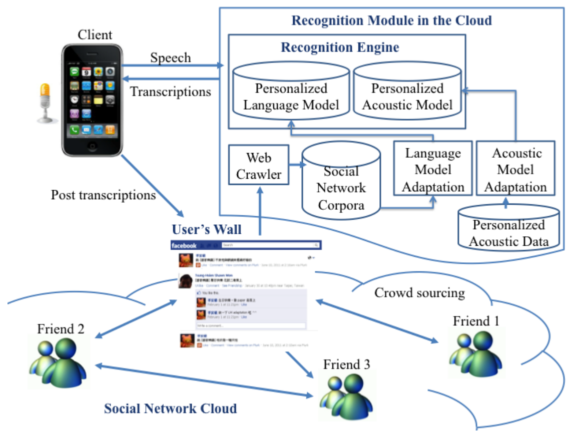

Figure 1: *The ecosystem of our crowdsourcing-based voice access of cloud application. A speaker-dependent recognizer is obtained by AM and LM personalization. A web crawler was implemented for collecting social posts from users for LM personalization.*

3. Proposed Approach

As shown in Fig. 2, the RNNLM [26] comprises three layers: the input layer, the hidden layer, and the output layer. The input word vector $w(t)$ represents the word at time t using a 1-of-N encoding. The context vector $s(t)$ uses the distributed representation for arbitrary length histories at time t, with a recurrent connection considering the time-delayed context vector $s(t-1)$. The output layer $y(t)$ then generates the probability distribution of the next word. In order to provide complementary information such as part-of-speech tags, topic information, or morphological information to the input vector $w(t)$, a context-dependent RNNLM variant [28] adds an additional feature layer $f(t)$ to the network through which it connects to both the hidden and output layers. Therefore, the network weights that must be learned are the matrices $\mathcal{W}$, $\mathcal{F}$, $\mathcal{S}$, $\mathcal{G}$, and $\mathcal{O}$. The learning process maximizes the likelihood of the training data using the back-propagation through time (BPTT) algorithm. Typically, a validation set is also used to control the training epochs and learning rates.

3.1. RNNLM Personalization

Considering the fact that the collected personal corpora are small, it is impossible to train a standalone RNNLM using only this small amount of training data. As a result, we fall back to the idea of LM adaptation [10]. The basic framework for LM

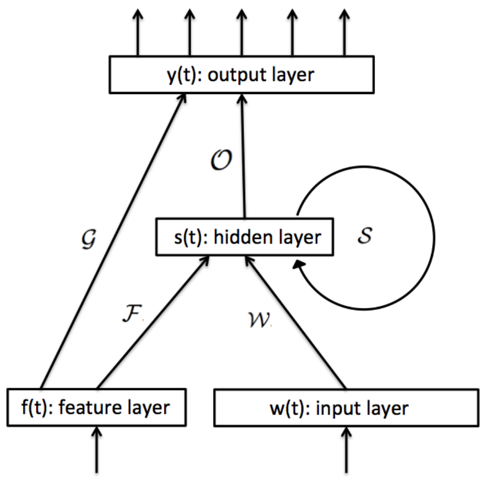

Figure 2: *The structure of context-dependent recurrent neural network LMs.*

adaptation considers two text corpora: the first is the adaptation corpus A, which is "in-domain" or updated with respect to the target recognition task, but possibly too small to train a robust standalone LM; the other is a large background corpus B, which may not be sufficiently related to the target task or may perhaps be outdated. In traditional N-gram models, the two corpora are interpolated; various methods are used to estimate their interpolation weights [35, 36, 37, 38]. However, for NN-based LMs, there is still little focus on the problem of adaptation. Unlike previous works [24, 39], which proposed adding an additional adaptation layer between the projection and input layers in feed-forward NNLMs, we propose directly fine-tuning the network weights using the adaptation corpus A on an RNN-based background LM. model structure as simple as possible. Moreover, instead of considering only the user's personal corpus A and background corpus B, we also take into account the corpora of his/her friends, C. The following three steps describe the training and adaptation of the personalized RNNLMs:

1. Train a general-domain RNNLM using the background corpus B, which is split into a training set and a validation set such that the training set likelihood is maximized and the validation set is used to control the training epochs. An initial set of model parameters $\mathcal{W}_0$, $\mathcal{F}_0$, $\mathcal{S}_0$, $\mathcal{G}_0$, and $\mathcal{O}_0$ are obtained here.

2. Given a target user's corpus A, split it into training set T_a and validation set V_a. Copy one background RNNLM and use BPTT to fine-tune parameters $\mathcal{W}_0$, $\mathcal{F}_0$, $\mathcal{S}_0$, $\mathcal{G}_0$, and $\mathcal{O}_0$ by maximizing the likelihood of the training set T_a while controlling the epochs using the validation set V_a. Fine-tuning yields the adapted model parameters $\mathcal{W}'$, $\mathcal{F}'$, $\mathcal{S}'$, $\mathcal{G}'$, and $\mathcal{O}'$.

3. Given corpora C, we directly treat C as a complete training set again maximizing its likelihood but still controlling the epochs by V_a. This yields $\mathcal{W}''$, $\mathcal{F}''$, $\mathcal{S}''$, $\mathcal{G}''$, and $\mathcal{O}''$.

There are several benefits of this three-step RNNLM personalization: First, due to the common computational overhead found in RNNLMs, the background RNNLM forms a solid foundation for future model fine-tuning where it learns some general domain histories beforehand. Secondly, although the amount of training data is small, it is believed that the distributed representation of RNNLMs can amplify the training efficiency since one training sentence can inform the model about a combinatorial number of other sentences in the continuous space. Unlike our previous approach [18], which computes similarities as interpolation weights while incorporating the corpora of multiple friends, the use of personal validation set V_a in Step 3 makes the approach completely data-driven. We are no longer worried about how to select and weigh those friends, but only about how to maximize the training data we can see, while at the same time, use the personal validation set V_a to point the adaptation in the right direction. Lastly, considering that the recurrent connection and the additional feature layer have already made the RNNLM structure more complex, we keep the model as simple as possible without adding any additional layers.

3.2. User-Oriented Features

As shown previously in Fig. 2, an additional feature layer $f(t)$ was added to the context-dependent RNNLM [28] in order to provide complementary information for the input word vector. In a previous study [29], adding complementary features such as part-of-speech tags, word lemmas, and topics, reduced both the model perplexity and word error rate. Besides the features mentioned above, the use of language is usually influenced by the demographic characteristics of the individual, his/her preferences, interests, or even the role he/she plays in a community. For example, male users may mention superhero movies more often than female users. People who often work together may share the same jargon. These kinds of user-oriented features are hard to obtain in traditional text corpora, but are relatively easy to glean from social networks.

3.2.1. Demographic information

Consider demographic information that can separate p users into K clusters. We compute the word distribution $P(w|C_k)$ over those K clusters by

$$P(w|C_k) = \frac{\sum_{p\in C_k} n(w,p)}{\sum_k \sum_{p\in C_k} n(w,p)} \quad (1)$$

where $n(w,p)$ is the number of occurrences of word w in the corpus of p users. The resulting word distribution $P(w|C_k)$ over clusters is then fed into the feature layer $f(t)$, as shown in Fig. 2. We simply compute and cascade several demographic feature vectors in the same way to obtain richer information.

3.2.2. Latent user group

Often, the use of language is not directly related to the demographic characteristics of a person. Instead, his/her interests, habits, idols, community, friends, etc. play an even more important role [40, 41]. However, modeling all these phenomena is difficult not only because too many aspects make the feature dimension too large to efficiently train the RNNLMs, but also because they are elusive, making explicit modeling impractical. In order to model these elusive aspects implicitly, we use the method called latent factor model method [4, 42]. First, we construct a user-word matrix M from the user corpora that we have collected. For simplicity, we consider only unigrams. Then we conduct singular value decomposition (SVD) to identify the latent factors in the matrix. Retaining only the top-K eigenvalues, we project both users and words onto a latent space of dimension K, where each dimension represents one latent factor that the user or word possesses. These latent factor features of words are then fed into the feature layer. Note that although our method is similar to latent semantic analysis [42], there is one fundamental difference. LSA factorizes a document-word matrix to identify latent topics in the documents, whereas our method factorizes the user-word matrix to identify any possible correlations among them.

4. Experiments

4.1. Experimental Setup

4.1.1. Corpus & LMs

Our experiments are conducted on a crawled Facebook dataset. A total of 42 users logged in and authorized this project to collect their messages and basic information for the purpose of academic research. These 42 users were treated as our target users, and we attempted to build a personalized LM for each of them. For each target user, 3/5 of his/her corpus was taken as the training set, 1/5 as the validation set, and the remaining 1/5 as testing data to compute perplexity from. Furthermore, with their consent, the observable public data of these 42 target users were also made available to our system. Through this process, besides the personal corpora for the 42 target users, we also had an entire set of publicly observable data. These were, primarily, the data of those individuals linked to the 42 users on the network. Thus, we had 93,000 anonymous personal corpora, and we collected 3.3 million sentences. After preprocessing and filtering, the number of sentences used in the work was approximately 2.4 million. The number of sentences for each user ranged from 1 to 8,566 with a mean of 25.7, comprising 10.6 words (Chinese, English, or mixed) per sentence in average. On the network, each target user had an average of 238 friends. For the background LM, 250 million sentences were collected from another popular social networking site called Plurk. There were both Chinese and English words in the Plurk data with a mixing rate of 9:1. The modified Kneser-Ney algorithm [43] was used for the N-gram LM smoothing. The most frequent 18,000 English words and 46,000 Chinese words appearing in the corpus were selected to form the lexicon. The SRILM [44] toolkit was used for the N-gram LM training and adaptation. For RNNLMs, we utilized the RNNLM toolkit [45] for implementation. We used three different feature sets in our experiment: POS tags (POS), demographic features (DE) reflecting gender (male, female, unknown) and language settings (Traditional Chinese, Simplified Chinese, English, unknown), and latent user groups (LUG) with 100 latent dimensions. We reported the results of using these features independently as well as a combination of feature sets.

4.1.2. N-best rescoring

For the n-best rescoring experiments, we used lattices produced using the HTK toolkit [46] to generate 1,000-best lists for rescoring. The LMs we used to generate the n-best lists were trigram models, adapted using the personal corpora as well as friend corpora with Kneser-Ney smoothing (KN3). The Mandarin triphone AMs used for first-pass decoding were trained on the ASTMIC corpus with 37 Chinese phones, whereas the English triphone AMs were trained on the Sinica Taiwan English corpus with 35 English phones. Both training sets included hundreds of speakers. The AMs were also adapted by the unsupervised MLLR speaker adaptation. We report the results both with and without speaker adaptation on two different test sets: Set *(I)* and Set *(II)*. The users in Set *(I)* were a subset of the users in Set *(II)*, in which the 42 target users with richer data were selected. Table 1 is a comparison of the two test sets. The word error rates (WER) reported here were averaged over all sentences. The LM and AM decode weights of 8.0 and 0.5, respectively, were set empirically.

Table 1: Summary of two test sets for n-best rescoring.

	# of user	speakers	# of utt.	record env.
Set (I)	12	Two	840	Clean
Set (II)	42	Multiple	948	Various

Table 2: Perplexity results. Both RNNLMs and Kneser-Ney 3grams (KN3) trained with background corpus only (*B*), further adapted with personal corpus (*B+S*), and even further adapted with friend corpora (*B+S+F*) are shown. Several combination of hidden layer sizes (100H, 200H, 500H) and feature settings are also reported. Features used are defined in Sec. 4.1.1.

Model		Perplexity
(a)	KN3, B	343.57
	KN3, B+S	299.32
	KN3, B+S+F	233.20
(b)	RNN 100H, B	309.50
	RNN 200H, B	289.14
	RNN 500H, B	267.38
(c)	RNN 500H, B+S	234.55
	RNN 500H, B+S+F	209.56
(d)	RNN 500H, B+S+F, POS	196.50
	RNN 500H, B+S+F, DE	199.23
	RNN 500H, B+S+F, LUG	195.59
	RNN 500H, B+S+F, ALL	192.83
(e)	RNN 500H, B+S+F, ALL + KN3	155.40

4.2. Experimental Results

4.2.1. Perplexity

Table 2 shows the preliminary results of the perplexity experiments using our proposed approach. Both RNNLMs and Kneser-Ney 3grams (KN3) trained with background corpus only (*B*), further adapted with personal corpus (*B+S*), and even further adapted with friend corpora (*B+S+F*) are shown. Several combination of hidden layer sizes (100H, 200H, 500H) and feature settings are also reported. It is not surprising that the adaptation works well both on RNNLMs and N-grams ($B+S+F < B+S < B$ in (a)(c)) considering the mismatch between the background corpus and each individual. RNNLMs with three different hidden layer sizes and different feature settings are also reported. Generally speaking, RNNLMs with larger hidden layers perform better than smaller ones (500H < 200H < 100H in (b)) due to their ability to represent richer histories. Adding additional features resulted in better perplexity performance ((d) < (c)). Although different features seem to provide the model with different capabilities of predicting the next word, but they doesn't vary too much in perplexity either using only one set of feature solely or combination of them. This may due to the fact that models with large hidden layers can learn to compensate for those information contained in different set of features by their stronger context. Lastly, as many previous studies have shown, RNNLMs, in general, outperform N-grams in perplexity, but perform even better when combined together ((e) < (a)(d)). For our best model (RNN 500H, *B+S+F*, ALL + KN3), we reduced the perplexity by 54.7% compared to background N-grams (KN3, B), around 41.8% compared to its simplest RNNLM version (RNN 500H, *B*).

4.2.2. Rescoring

Table 3 reports the selected n-best rescoring results using our proposed methods on two different test sets, Set (I) and Set (II), with and without AM adaptation, respectively. We decoded our first-pass recognition results using three different trigram LMs. They were models trained on the background corpus only (KN3 B), adapted by personal corpus (KN3 B+S), and further adapted by friend corpora (KN3 B+S+F). We find that, as expected, the

Table 3: The selected n-best rescoring results conducted on two different test sets, Set (I) and Set (II), with/without adapted AM (MLLR/AM) respectively.

WER(%)		Set (I)		Set (II)	
		AM	MLLR	AM	MLLR
First-pass	KN3 B	40.98	38.75	49.19	43.80
	KN3 B+S	40.31	38.08	48.63	43.39
	KN3 B+S+F	33.45	30.13	46.81	41.95
RNN 500H	RNN None	32.59	29.80	44.87	40.10
	RNN POS	32.51	29.50	44.84	**40.06**
	RNN DE	32.65	29.95	44.94	40.18
	RNN LUG	32.95	29.36	**44.78**	40.17
	RNN POS+LUG	32.46	29.58	44.86	40.17
RNN 500H+KN3	RNN None	32.49	29.37	45.31	40.57
	RNN POS	32.53	**29.26**	45.23	40.38
	RNN DE	**32.37**	29.91	45.47	40.40
	RNN LUG	32.51	**29.26**	45.35	40.53
	RNN POS+LUG	32.75	29.75	45.32	40.52

adapted models performed better (KN3 B+S+F < KN3 B+S < B). We used the best adapted N-gram models (KN3 B+S+F) to generate lattices, and 1000-best lists were extracted from the lattices, which were then used in the n-best rescoring experiments. The RNNLMs used for rescoring were all configured to 500 hidden layers while adding different types of features. The interpolation weight between RNNLM and KN3 were empirically set to 0.75. Generally, all rescoring results were better than the first-pass results (RNN KN3, RNN < First-pass), which indicates that the personalized RNNLMs perform better than personalized N-grams. However, as to the combination of RNNLMs with N-grams, Set (I) and Set (II) yielded different conclusions. We obtained better results on Set (I) when we combined them together, but got worse results for the combination on Set (II). This may be due to RNNLMs can better handle noisy conditions where shorter context seems to be not reliable anymore, by its capability of modeling longer context. We also found that more features does not necessarily yield better results (RNN POS+LUG) given that the back-off information provided by auxiliary features maybe overlap or interfere with each other when the feature layer is oversize. In our experiments, generally speaking, a single use of POS or LUG features yielded more improvements compared to other settings.

5. Conclusion

In this paper, we proposed a new framework for RNNLM personalization in which the social network data was crawled from a crowdsourcing platform as adaptation resources. A three-step adaptation mechanism considering the feasibility and complexity was also proposed. Some user-oriented features were also incorporated into the model for better word prediction capabilities. Experiments showed that by using personalized RNNLMs while adding user-oriented features, the perplexity can be reduced up to 54.7% compared to background N-grams and 41.8% compared to original RNNLM. Moreover, n-best rescoring experiments also yielded 2.9% to 4.5% relative word error rate reductions on different experimental settings.

6. References

[1] G. Zweig and C. Shuang yu, “Personalizing model m for voice-search,” in *Proc. on InterSpeech*, 2011.

[2] M. Speretta and S. Gauch, “Personalized search based on user search histories,” in *Proc. on Web Intelligence*, 2005.

[3] Y. H. Cho, J. K. Kim, and S. H. Kim, “A personalized recommender system based on web usage mining and decision tree induction,” *Expert Systems with Applications*, 2002.

[4] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, 2009.

[5] F. Walter, S. Battiston, and F. Schweitzer, “A model of a trust-based recommendation system on a social network,” *Autonomous Agents and Multi-Agent Systems*, 2008.

[6] M.-H. Park, J.-H. Hong, and S.-B. Cho, “Location-based recommendation system using bayesian users preference model in mobile devices,” in *Ubiquitous Intelligence and Computing*, 2007.

[7] C. J. Leggetter and P. C. Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models,” *Computer Speech and Language*, 1995.

[8] P. C. Woodland, “Speaker adaptation for continuous density hmms: A review,” in *Proc. on ITRW on Adaptation Methods for Speech Recognition*, 2001.

[9] l. G. Jean and C.-H. Lee, “Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains,” *IEEE Transactions on Speech and Audio Processing*, 1994.

[10] J. R. Bellegarda, “Statistical language model adaptation: review and perspectives,” *Speech Communication*, 2004.

[11] A. Heidel and L.-S. Lee, “Robust topic inference for latent semantic language model adaptation,” in *Proc. on ASRU*, 2007.

[12] H. Bo-June and J. Glass, “Style and topic language model adaptation using hmm-lda,” in *Proc. on EMNLP*, 2006.

[13] D. Hakkani-Tur, G. Tur, and L. Heck, “Research challenges and opportunities in mobile applications,” *Signal Processing Magazine, IEEE*, 2011.

[14] X. Sun and A. May, “The role of spatial contextual factors in mobile personalization at large sports events,” *Personal and Ubiquitous Computing*, 2009.

[15] S. Arbanowski, P. Ballon, K. David, O. Droegehorn, H. Eertink, W. Kellerer, H. van Kranenburg, K. Raatikainen, and R. Popescu-Zeletin, “I-centric communications: personalization, ambient awareness, and adaptability for future mobile services,” *Communications Magazine*, 2004.

[16] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, “Class-based n-gram models of natural language,” *Computational Linguistics*, 1992.

[17] A. Stolcke, “Entropy-based pruning of backoff language models,” in *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, 2000.

[18] T.-H. Wen, H.-Y. Lee, T.-Y. Chen, and L.-S. Lee, “Personalized language modeling by crowd sourcing with social network data for voice access of cloud applications,” in *Proc. on IEEE SLT workshop*, 2012.

[19] J. Wu and S. Khudanpur, “Combining nonlocal, syntactic and n-gram dependencies in language modeling,” in *Proc. on EuroSpeech*, 1999.

[20] S. Khudanpur and J. Wu, “Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling,” *Computer Speech and Language*, 2000.

[21] H. S. Le, A. Allauzen, and Y. Fran, “Measuring the influence of long range dependencies with neural network language models,” in *Proc. on NAACL-HLT Workshop*, 2012.

[22] I. Oparin, M. Sundermeyer, H. Ney, and J. Gauvain, “Performance analysis of neural networks in combination with n-gram language models,” in *Proc. on ICASSP*, 2012.

[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *Journal of Machine Learning Research*, 2003.

[24] J. Park, X. Liu, M. J. F. Gales, and P. C. Woodland, “Improved neural network based language modeling and adaptation,” in *Proc. on InterSpeech*, 2010.

[25] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, “Structured Output Layer neural network language model,” in *Proc. on ICASSP*, 2011.

[26] T. Mikolov, M. Karafit, L. Burget, J. Cernock, and S. Khudanpur, “Recurrent neural network based language model,” in *Proc. on InterSpeech*, 2010.

[27] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. on ICASSP*, 2011.

[28] T. Mikolov and G. Zweig, “Context dependent recurrent neural network language model,” in *Proc. on IEEE SLT workshop*, 2012.

[29] Y. Shi, P. Wiggers, and C. M. Jonker, “Towards recurrent neural networks language models with linguistic and contextual features,” in *Proc. on InterSpeech*, 2012.

[30] A. Doan, R. Ramakrishnan, and A. Y. Halevy, “Crowdsourcing systems on the world-wide web,” *Communications of the ACM*, 2011.

[31] Munro and Robert, “Crowdsourcing and language studies: the new generation of linguistic data,” in *Proc. on NAACL*, 2010.

[32] K. Berberich, S. Bedathur, O. Alonso, and G. Weikum, “A language modeling approach for temporal information needs,” in *Advances in Information Retrieval*, 2010.

[33] J. Liu, S. Cyphers, P. Pasupat, I. McGraw, and J. Glass, “A conversational movie search system based on conditional random field,” in *Proc. on InterSpeech*, 2012.

[34] I. McGraw, S. Cyphers, P. Pasupat, J. Liu, and J. Glass, “Automating crowd-supervised learning for spoken language systems,” in *Proc. on InterSpeech*, 2012.

[35] R. Iyer and M. Ostendorf, “Modeling long distance dependence in language: topic mixtures versus dynamic cache models,” *IEEE Transactions on Speech and Audio Processing*, 1999.

[36] A. Heidel, H.-A. Chang, and L.-S. Lee, “Language model adaptation using latent dirichlet allocation and an efficient topic inference algorithm,” in *Proc. on InterSpeech*, 2007.

[37] M. Federico, “Efficient language model adaptation through mdi estimation,” in *Proc. on EuroSpeech*, 1999.

[38] C. Chelba and F. Jelinek, “Structured language modeling,” *Computer Speech and Language*, 2000.

[39] X. Liu, M. J. F. Gales, and P. C. Woodland, “Improving lvcsr system combination using neural network language model cross adaptation,” in *Proc. on InterSpeech*, 2011.

[40] J. Paolillo, “The virtual speech community: Social network and language variation on irc,” *Journal of Computer-Mediated Communication*, 1999.

[41] D. Rosen and M. Corbit, “Social network analysis in virtual environments,” in *Proc. on ACM Hypertext*, 2009.

[42] T. K. Landauer, P. W. Foltz, and D. Laham, “An Introduction to Latent Semantic Analysis,” *Discourse Processes*, 1998.

[43] F. James, “Modified kneser-ney smoothing of n-gram models modified kneser-ney smoothing of n-gram models,” Tech. Rep., 2000.

[44] A. Stolcke, “Srilm - an extensible language modeling toolkit,” in *Proc. on Spoken Language Processing*, 2002.

[45] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, “Rnnlm - recurrent neural network language modeling toolkit,” in *Proc. on ASRU*, 2011.

[46] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*. Cambridge University Press, 2006.