Neurocomputing 72 (2009) 1414–1418

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Improving a statistical language model through non-linear prediction

Andriy Mnih *, Zhang Yuecheng, Geoffrey Hinton

*University of Toronto, Department of Computer Science, Toronto, Ontario, Canada*

ARTICLE INFO

Available online 13 January 2009

*Keywords:*
Statistical language modelling
Distributed representations
Neural networks

ABSTRACT

We show how to improve a state-of-the-art neural network language model that converts the previous "context" words into feature vectors and combines these feature vectors linearly to predict the feature vector of the next word. Significant improvements in predictive accuracy are achieved by using a non-linear subnetwork to modulate the effects of the context words or to produce a non-linear correction term when predicting the feature vector. A log-bilinear language model that incorporates both of these improvements achieves a 26% reduction in perplexity over the best $n$-gram model on a fairly large dataset.



## 1. Introduction

The fundamental problem of statistical language modelling is capturing the distribution of words in a sentence. The task of modelling distributions over sequences of words is greatly simplified by making the Markov assumption which states that the distribution of the next word depends only on some fixed number of immediately preceding words. If we assume that only $k$ previous words have an effect on the distribution of the next word, the probability of a sequence of words $w_1,\ldots,w_N$ can be expressed as follows:

$$P(w_{1:N}) = P(w_{1:k-1})\prod_{i=1}^{N} P(w_i|w_{i-1:i-k}), \quad (1)$$

where $w_{1:i}$ denotes $w_1,\ldots,w_i$. As a result, the task of modelling the distribution of sequences of arbitrary length is reduced to modelling the distribution of the next word given $k$ preceding words, which are called the *context*. Though the Markov assumption is clearly false for natural languages, modelling $P(w_{k+1}|w_{1:k})$ can result in very effective statistical language models.

$n$-Gram models, which are the most popular statistical language models, are effectively (sparse) probability tables that store $P(w_n|w_{1:n-1})$. These models are estimated by counting $n$-tuples in the training data and normalizing the counts appropriately. Smoothing these naive estimates is essential for achieving good performance, since the number of parameters in these models is exponential in the context size. A number of effective smoothing methods have been developed specifically for $n$-gram models [4]. However, using larger context sizes (e.g. 5 and larger) does not usually lead to better models due to overfitting even when the best smoothing methods are used [7]. This problem is a consequence of the way $n$-gram models parameterize $P(w_n|w_{1:n-1})$: there is a separate free parameter for each context/next word pair. Since there are no a priori smoothness constraints on the parameters, there is no natural way to use information about similar contexts when estimating $P(w_n|w_{1:n-1})$ for a particular context. Another consequence of this parameterization is that $n$-gram models tend to require large amounts of memory, since each non-zero entry in the table for $P(w_n|w_{1:n-1})$ has to be stored explicitly.

By representing words using real-valued feature vectors and modelling $P(w_n|w_{1:n-1})$ as a smooth function of the context word feature vectors, neural network language models avoid the drawbacks of $n$-gram models. Since words that are used in similar ways are mapped to similar feature vectors by such models, similar words are guaranteed to have similar probabilities of occurring in similar contexts. Memory requirements of such language models are linear in the context size, which is a great improvement over the exponential memory requirements of $n$-gram models. A number of neural network language models have been proposed [10,6,3], the best known of which is the neural probabilistic language model (NPLM) introduced in [1].

Our starting point in this paper is the log-bilinear (LBL) model from [9]. Unlike other statistical language models based on distributed representations, the LBL model has been shown to outperform the best $n$-gram models on a large dataset without resorting to model averaging. The goal of this paper is to improve the LBL model by removing some of its obvious limitations without adding too many free parameters. Some of our preliminary results have been published in [12].

## 2. The LBL language model

The LBL language model learns to convert each word into a vector of real-valued features in such a way that the feature

* Corresponding author.
*E-mail address:* amnih@cs.toronto.edu (A. Mnih).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.12.025

vectors of the $n-1$ context words are good at predicting the feature vector for the next word. The prediction is done by using $n-1$ different, learned weight matrices to linearly transform and combine the contextual feature vectors to produce a predicted feature vector $\hat{r}$:

$$\hat{r} = \sum_{i=1}^{n-1} C_i r_{w_i}, \tag{2}$$

where $r_w$ is the feature vector for word $w$. The predicted feature vector is then compared to the feature vectors of all the words in the dictionary using the dot product, resulting in a set of similarity scores which are then exponentiated and normalized to obtain the predicted distribution for the next word:

$$P(w_n = w|w_{1:n-1}) = \frac{\exp(\hat{r}^{\mathrm{T}} r_w + b_w)}{\sum_j \exp(\hat{r}^{\mathrm{T}} r_j + b_j)}, \tag{3}$$

where the sum is over all the words in the dictionary. Here $b_w$ is the bias for word $w$ that captures the a priori probability of observing word $w$. All model parameters including the matrix of word feature vectors $R$ and the context weight matrices are learned by maximizing the penalized log-likelihood using steepest ascent.

## 3. Extending the LBL model

A major limitation of the LBL model is that the interactions between the context words are not taken into account when predicting the next word. For example, the occurrence of a full stop in the context does not affect the contribution of the context words that come before it to the predicted feature vector for the next word. Intuitively, the effect of words that precede the full stop should be downplayed when predicting words that come after the full stop, since they belong to a different sentence.

For example, in the 5-word[1] sequence, *"the newspaper. For example"*, due to the full stop, *"the"* and *"newspaper"* are much less useful than *"For"* and *"example"* for predicting the next word (which is usually a comma). Hence, an effective model should reduce the effects of *"the"* and *"newspaper"* and amplify the effect of *"For"* and *"example"*.

### 3.1. The gated LBL model

Perhaps the simplest way to deal with this limitation to introduce a way to modulate the contribution of the context words to the predicted representation for the next word. We achieve this by augmenting the LBL model with a gating subnetwork with one hidden layer which produces weighting coefficients, one per context word. Each coefficient represents the relative importance of the corresponding context word for predicting the next word. The gating subnetwork takes the context word feature vectors as inputs and combines the hidden layer activities, which represent the learned higher-order context features, to compute a weighting coefficient for each context word. The predictive effect of each context word is weighted by its coefficient before being incorporated into the predicted representation for the next word.

Let $r$ be the vector obtained by concatenating the feature vectors for the context words in the order they appear in the context. The activity $h_j$ of the hidden unit $j$ is then given by a logistic function of its total input:

$$h_j = \frac{1}{1 + \exp(-\sum_k A_{jk} r_k - a_j)}, \tag{4}$$

where $A$ is the matrix of weights between the context feature vectors and the hidden layer, and $a_j$ is the bias of the hidden unit $j$. The activity $s_i$ of the gating unit $i$ is a logistic function of the total input it receives from the hidden layer, scaled by a factor of 2 to make the gating unit output 1 when its total input is 0:

$$s_i = \frac{2}{1 + \exp\left(-\sum_j B_{ij} h_j - b_i\right)}, \tag{5}$$

where $B$ is the matrix of weights between the hidden layer and the gating units, and $b_i$ is the bias of the gating unit $i$. The gating unit activities are used to weight the context word feature vectors when predicting the feature vector for the next word as follows:

$$\hat{r} = \sum_{i=1}^{n-1} C_i s_i r_{w_i}. \tag{6}$$

As with the LBL model, the predicted distribution $P(w_n|w_{1:n-1})$ is obtained by plugging in the predicted representation into Eq. (3).

### 3.2. The LBL model with a general non-linear subnetwork

While the LBL model with a gating subnetwork is capable of capturing some interactions between the context words, the effect these interactions can have on the predicted feature vector is limited to upweighting and downweighting the contributions from the individual context words. To allow the interactions between context words to have more general effects on the predicted representation we added a general non-linear subnetwork to the LBL model. This subnetwork is a neural network with one hidden layer, that outputs a vector that is added to the predicted representation of the LBL model. Thus, this approach makes the predicted feature vector a fairly general non-linear function of the context feature vectors:

$$\hat{r} = \sum_{i=1}^{n-1} C_i r_{w_i} + Sh. \tag{7}$$

Here $h$ is the vector of hidden unit activities computed using Eq. (4) and $S$ is the matrix of weights from the hidden units to the predicted feature vector for the next word.

### 3.3. The LBL model with gating and non-linear prediction

While the non-linear subnetwork should be able to capture very general interactions between context words, it might still be beneficial to incorporate the non-linear subnetwork and the gating subnetwork into a single model. In such a model the gating network should handle the gating interactions allowing the non-linear network to concentrate on more complex interactions. The predicted representation for the next word in such a model is computed as follows:

$$\hat{r} = \sum_{i=1}^{n-1} C_i s_i r_{w_i} + Sh. \tag{8}$$

We will call the resulting model as the GLBLN model because it is a combination of the gated LBL (GLBL) model and the LBL model with a non-linear subnetwork (LBLN) (Figs. 1 and 2).

### 3.4. Relationship to the NPLM

The three language models proposed in this paper are extensions of the LBL model introduced in [9]. One of the main

[1] We treat punctuation marks as special words.

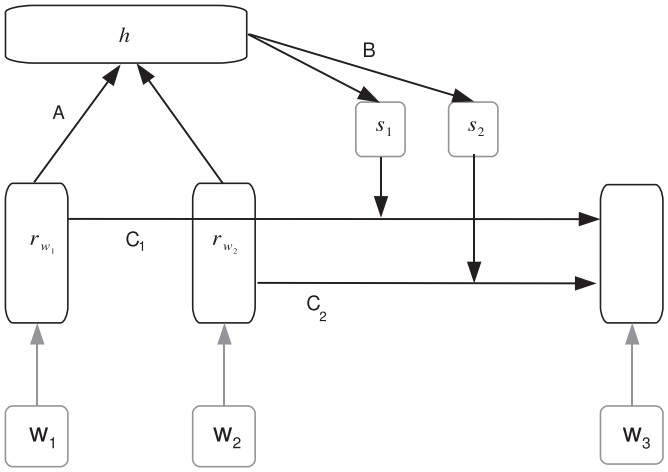


**Fig. 1.** The log-bilinear model with a context of size 2 and a gating subnetwork (GLBL).

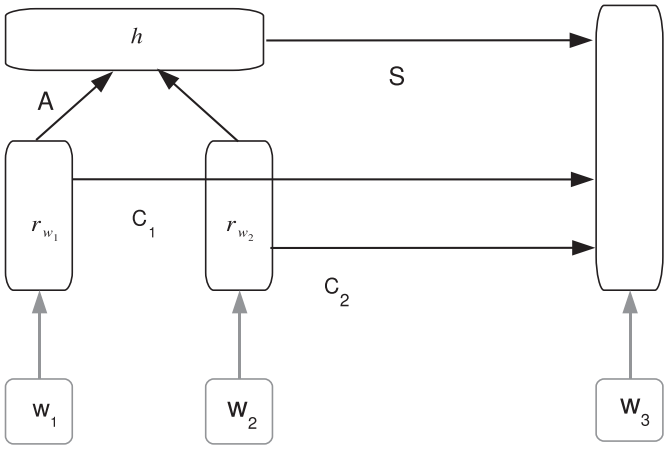


**Fig. 2.** The log-bilinear model with a context of size 2 and a general non-linear subnetwork (LBLN).

features that distinguish the LBL model from the NPLM [2] is that it does not have a non-linear hidden layer. Since all three of our extensions of the LBL model involve a non-linear hidden layer, we will explain here what makes them different from the NPLM. The NPLM takes the feature vectors of the context words as inputs, passes them through a non-linear hidden layer, and then computes the activations in the softmax output layer. The LBL model can be viewed as the NPLM with a linear hidden layer that has as many hidden units as word feature vector dimensions, and the weights from the hidden layer to the output layer set to the word feature vector matrix $R$. In this version of the NPLM the activities of the hidden units correspond to the predicted feature vector for next word as computed by the LBL model. Our extensions in this paper involve introducing a non-linear hidden layer that takes the context word feature vectors as inputs and somehow affects the predicted representation for the next word. These extensions then correspond to introducing a hidden layer that modifies the inputs to the original (linear) hidden layer of the LBL-seen-as-NPLM.

## 4. Experimental results

We evaluated our models using the Associated Press News (APNews) dataset. The dataset has been preprocessed by replacing all proper nouns, numbers, and rare words with special symbols, reducing the number of unique words to 17 964. For a more detailed description of the preprocessing procedure see [2]. The dataset was split into a training set of 14 million words, a validation set of 1 million words, and a test set of 1 million words. This version of the dataset has already been used for evaluating the NPLM [2] and the LBL model [9], which allows direct comparison to the results reported in those papers.

All models were evaluated based on their perplexity on the test set. The perplexity of a sequence under a model $P(w_n|w_{1:n-1})$ is given by

$$\mathscr{P} = \exp\left(-\frac{1}{N}\sum_{w_{1:n}} \log P(w_n|w_{1:n-1})\right), \tag{9}$$

where the sum is over all subsequences of length $n$ in the sequence.

We performed two sets of experiments, with the difference between the sets being the context size used by the models. In the first set, the context size was 5, while in the second set it was 10. In both cases we used the LBL model from [9] with the corresponding context size and 100-dimensional word feature vectors as a baseline for comparison as well as the starting point from which we trained our extended models. That is, in all of our experiments we initialized the feature vector matrix $R$ and the context weight matrices $\{C_i\}$ by copying the corresponding matrices from the LBL model. The word feature vector matrix was kept fixed during learning in order to speed up convergence. Better results should be achievable by allowing the word feature vectors to be updated along with other parameters.

The weights in the gating subnetworks of the GLBL models were initialized to small random values, while the biases in these subnetworks were initialized to zero. This initialization procedure ensured that initially the GLBL models weighted all context positions almost equally by setting $s_i$ to values close to 1 and hence produced predictions very close to those of the initializing LBL model. The weights and biases in the non-linear subnetworks of the LBLN and GLBLN models and in the gating subnetworks of the GLBLN models were initialized in the same manner. Thus all of our models were initialized to nearly match the predictions of the LBL model they were based on to reduce the training time and ensure that the initial model state was sensible.

Models were trained by maximizing the penalized log-likelihood of the training data using steepest ascent. Parameters were updated after accumulating gradients from 1000 training cases. We used the following learning rates: $10^{-3}$ for the context weights ($C$), $10^{-1}$ for the weights to the hidden layer ($A$), and $10^{-5}$ for the weights from the hidden layer ($B$ and $S$). A momentum of 0.5 was used for all parameter updates. An $L2$ weight penalty of $10^{-5}$ was applied to all weights, but no penalty was applied to biases. Each model was trained until its performance on a subset of the validation set stopped improving.

Table 1 reports the test set perplexities for the models based on the LBL model with a context of size 5. We report four perplexity scores as the baselines for comparison. The first three are the scores for the back-off $n$-grams with modified Kneser–Ney discounting, fitted using the SRILM toolkit [11]. The best-performing $n$-gram model is the 5-gram, which shows that the $n$-gram models do not benefit from using large contexts on this dataset. The final baseline score is for the LBL model which was reported in [9] to achieve state-of-the-art performance, outperforming the 5-gram model by over 5%. The results show that both extensions of the LBL model result in a significant reduction in perplexity. Adding a gating subnetwork with 500 hidden units to the LBL model reduces its perplexity by 11%. Using fewer hidden units in the gating subnetwork leads to smaller reductions, but even with 100 hidden the perplexity of the GLBL model is more than 8% lower than that of the LBL model. Adding the

**Table 1**
Test set perplexity scores for the models trained on the APNews dataset.

| Model type | Number of hidden units | Test set perplexity | Perplexity reduction | Mixture test perplexity |
|---|---|---|---|---|
| KN 3-gram | – | 129.8 | −10.9% | – |
| KN 5-gram | – | 123.2 | − 5.3% | – |
| KN 9-gram | – | 124.6 | −6.5% | – |
| NPLM | – | – | – | 109 |
| LBL | – | 117.0 | 0% | 97.3 |
| GLBL | 100 | 107.1 | 8.5% | 93.3 |
| GLBL | 200 | 105.4 | 9.9% | 92.6 |
| GLBL | 300 | 105.4 | 9.9% | 92.2 |
| GLBL | 500 | 104.1 | 11.0% | 91.9 |
| LBLN | 100 | 105.4 | 9.9% | 92.2 |
| LBLN | 200 | 102.6 | 12.3% | 91.0 |
| LBLN | 300 | 101.2 | 13.5% | 90.3 |
| LBLN | 500 | 99.0 | 15.3% | 89.2 |
| GLBLN | 100 | 102.8 | 12.1% | 91.4 |
| GLBLN | 200 | 100.5 | 14.1% | 90.3 |
| GLBLN | 300 | 98.5 | 15.8% | 89.6 |
| GLBLN | 500 | 96.8 | 17.2% | 88.5 |

The reduction in perplexity for each model relative to the log-bilinear model is also shown. The mixture test perplexity was computed by averaging the predictions of the model with those of the Kneser–Ney 5-gram model. The score for the NPLM is from [2]. All network models in the comparison have a context of size 5.

**Table 2**
Test set perplexity scores for the models trained on the APNews dataset.

| Model type | Number of hidden units | Test set perplexity | Perplexity reduction | Mixture test perplexity |
|---|---|---|---|---|
| KN 3-gram | – | 129.8 | −20.4% | – |
| KN 5-gram | – | 123.2 | −14.3% | – |
| KN 9-gram | – | 124.6 | −15.6% | – |
| LBL | – | 107.8 | 0% | 92.1 |
| GLBL | 100 | 100.3 | 7.0% | 88.7 |
| GLBL | 200 | 99.4 | 7.8% | 88.1 |
| GLBL | 300 | 99.1 | 8.1% | 87.6 |
| GLBL | 500 | 98.5 | 8.6% | 87.3 |
| LBLN | 100 | 99.2 | 8.0% | 87.9 |
| LBLN | 200 | 97.2 | 9.8% | 86.8 |
| LBLN | 300 | 95.9 | 11.0% | 86.0 |
| LBLN | 500 | 94.1 | 12.7% | 85.1 |
| GLBLN | 100 | 96.4 | 10.6% | 86.9 |
| GLBLN | 200 | 93.8 | 13.0% | 85.6 |
| GLBLN | 300 | 92.6 | 14.1% | 84.7 |
| GLBLN | 500 | 91.1 | 15.5% | 83.9 |

The mixture test perplexity was computed by averaging the predictions of the model with those of the Kneser–Ney 5-gram model. All network models in the comparison have a context of size 10.

non-linear subnetwork to the LBL model works even better than adding a gating subnetwork. The reduction in perplexity for the LBLN models range from 9.9% for the model with 100 hidden units to 15.3% for the model with 500 hidden units. The GLBLN models that have both the gating subnetwork and the non-linear subnetwork achieved the lowest perplexity values, outperforming the LBLN models by a small margin. The GLBLN model with 500 hidden units achieved a test set perplexity of 96.8, which is the best result among the models with a context of size 5, a 17.2% reduction in perplexity relative to the LBL model. Table 1 also includes the score for the NPLM of Bengio et al. [2]. Since that paper includes the score for the mixture of the NPLM and a 5-gram on the APNews dataset but not the score for the NPLM on its own, we computed the corresponding mixture scores for our models as well. The scores show that when mixed with a 5-gram our models outperform the corresponding NPLM mixture, sometimes by as much as 18%. More impressively, all of our extended LBL models, even when not mixed with a 5-gram, outperform the NPLM/5-gram mixture.

The results for the models based on the LBL model with a context of size 10 are presented in Table 2. The pattern of perplexity reductions relative to the LBL model here is very similar to that of models with a context of size 5. Though the reductions are slightly smaller here, all models in this table outperform their context size 5 counterparts by 5–7 perplexity points. The best performing model among all models in our comparison is the GLBLN model with a context of size 10 and 500 hidden units which outperforms the LBL model with a context of size 10 by 15.5% and the 5-gram model by 26.0%. A mixture of the GLBLN model with the 5-gram outperforms the 5-gram by 31.9%.[2]

All three extended LBL model types benefit from an increased number of hidden units, though the improvement in performance as the number of hidden units increases is more pronounced in the LBLN and GLBLN models. The reduction in perplexity achieved by using a gating subnetwork is substantial, which shows that the gating subnetwork is capable of capturing important interactions between the context words. The fact that the non-linear subnetwork reduces model perplexity more than the gating subnetwork indicates that some important interactions cannot be captured by the gating subnetwork alone. On the other hand, the modest reduction achieved by using both the gating subnetwork and the non-linear subnetwork in a single model shows that in spite of its generality the non-linear subnetwork does not entirely supersede the gating network.

The preliminary results of our experiments with models based on the LBL model with a context of size 5 were reported in [12]. The experiments reported in the current paper have been performed using somewhat different learning parameter settings and weight initializations than those used for the preliminary experiments. These changes resulted in large performance improvements for the LBLN and GLBLN models. The most important change was the reduction of the weight cost parameter[3] down to $10^{-5}$. The other change was learning the weights to the hidden units in the LBLN models from scratch instead of copying them from the trained GLBL models, as was done in the preliminary experiments. While this resulted in slow initial progress as reported in [12], the rate of learning accelerated after the initial slow phase and the resulting models performed considerably better than the ones learned using the old initialization.

[2] To allow a fair comparison to the NPLM result, where the neural network model and the $n$-gram model were given equal weights in the mixture, we also used equal mixing weights when computing mixture perplexity for our model. Giving more weight to our network models would lower the perplexity of the resulting mixtures.

[3] Though [12] states that the weight cost was set to $10^{-5}$, the actual value used was $10^{-2}$, which led to significant underfitting for the LBLN and GLBLN models.

## 5. Discussion

We have shown that adding a gating subnetwork or a non-linear subnetwork to a LBL model results in a significant performance improvement. While the non-linear subnetwork is more effective at reducing perplexity than the gating subnetwork, the best results are achieved by using both subnetworks in the same model. Adding the two subnetworks to the same LBL model with a context of size 5 resulted in a 15% reduction in perplexity relative to the underlying LBL model. The resulting model outperformed the mixture of a NPLM and a Kneser–Ney 5-gram from [2]. When the two subnetworks were added to an LBL model with a context of size 10, the resulting model outperformed the best $n$-gram model by 26%. Mixing the predictions of our models with those from an $n$-gram model leads to a further performance boost.

In our experiments, the word feature vectors of the extended LBL models were initialized to the feature vectors of trained LBL models and were then kept fixed during training. Allowing these feature vectors to be updated during training is likely to lead to better performance. It would be interesting to see whether training extended LBL models without such an initialization would lead to better results. It might also be worth investigating whether allowing the gating and the non-linear subnetworks to have separate hidden layers would improve model performance.

In this paper, we viewed language modelling as an idealized sequence modelling task, where words were treated as arbitrary symbols the properties of which were to be learned from training sequences alone. In practice, extra information about words might be available making language modelling easier. One example of such information is the letter-level representation of words that can be used to produce better estimates of feature vectors for rare words than co-occurrence data alone would allow [8]. Augmenting input words with syntactic information, such as part-of-speech tags, might also lead to improved predictive accuracy as shown in [5]. In contrast to $n$-grams, the flexibility of neural language models such as the NPLM and the LBL model allows them to be easily extended to deal with such augmented word representations without increasing the number of model parameters dramatically. We believe that the superior performance of the LBL model and its non-linear extensions demonstrated in this paper make them excellent candidates for replacing the $n$-gram models and the NPLM as the language modelling subsystems of choice for real-world systems.

## Acknowledgments

The research was supported by NSERC and CFI. G.E.H. is a fellow of the Canadian Institute for Advanced Research.

## References

[1] Y. Bengio, R. Ducharme, P. Vincent, A neural probabilistic language model, in: NIPS, 2000, pp. 932–938.
[2] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, Journal of Machine Learning Research 3 (2003) 1137–1155.
[3] J. Blitzer, K. Weinberger, L. Saul, F. Pereira, Hierarchical distributed representations for statistical language modeling, in: Advances in Neural Information Processing Systems, vol. 18, MIT Press, Cambridge, MA, 2005.
[4] S.F. Chen, J. Goodman, An empirical study of smoothing techniques for language modeling, in: Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, San Francisco, 1996, pp. 310–318.
[5] A. Emami, F. Jelinek, A neural syntactic language model, Machine Learning 60 (1–3) (2005) 195–227.
[6] A. Emami, P. Xu, F. Jelinek, Using a connectionist model in a syntactical based language model, in: Proceedings of ICASSP, vol. 1, 2003, pp. 372–375.
[7] J. Goodman, A bit of progress in language modeling, Technical report, Microsoft Research, 2000.
[8] H. Larochelle, Y. Bengio, Distributed representation prediction for generalization to new words, Technical Report 1284, Département d'informatique et recherche opérationnelle, Université de Montréal, 2006.
[9] A. Mnih, G.E. Hinton, Three new graphical models for statistical language modelling. in: ICML, 2007, pp. 641–648.
[10] H. Schwenk, J.L. Gauvain, Connectionist language modeling for large vocabulary continuous speech recognition, in: Acoustics, Speech, and Signal Processing, 2002. Proceedings of the IEEE International Conference on, vol. 1, 2002, (ICASSP'02).
[11] A. Stolcke, SRILM—an extensible language modeling toolkit, in: Proceedings of the International Conference on Spoken Language Processing, vol. 2, 2002, pp. 901–904.
[12] Z. Yuecheng, A. Mnih, G. Hinton, Improving a statistical language model by modulating the effects of context words, in: ESANN, 2008.

**Andriy Mnih** received the B.Math. degree in Computer Science from the University of Waterloo in 2002 and the M.Sc. degree in Computer Science from the University of Toronto in 2004. He is currently a Ph.D. student in the Department of Computer Science at the University of Toronto. His research interests include statistical language modelling, collaborative filtering, and large-scale Bayesian inference methods.

**Zhang Yuecheng** received Honours Bachelor of Science in Computer Science and Economics from the University of Toronto in 2008. She received the Undergraduate Research Scholarship from NSERC and did research on Statistical Language Models under the guidance of Professor Geoffrey E. Hinton in summer 2007.

**Geoffrey Hinton** received his BA in Experimental Psychology from Cambridge in 1970 and his Ph.D. in Artificial Intelligence from Edinburgh in 1978. He did Postdoctoral work at Sussex University and the University of California San Diego and spent five years as a faculty member in the Computer Science Department at the Carnegie-Mellon University. He then became a fellow of the Canadian Institute for Advanced Research and moved to the Department of Computer Science at the University of Toronto. He holds a Canada Research Chair in Machine Learning and is the director of the program on "Neural Computation and Adaptive Perception" which is funded by the Canadian Institute for Advanced Research. Geoffrey Hinton is a fellow of the Royal Society, the Royal Society of Canada, and the Association for the Advancement of Artificial Intelligence. He is an honorary foreign member of the American Academy of Arts and Sciences, and a former president of the Cognitive Science Society. He received an honorary doctorate from the University of Edinburgh in 2001. He was awarded the first David E. Rumelhart prize (2001), the IJCAI award for research excellence (2005), the IEEE Neural Network Pioneer award (1998), and the ITAC/NSERC award for contributions to information technology (1992).

He investigates ways of using neural networks for learning, memory, perception and symbol processing and has over 200 publications in these areas. He was one of the researchers who introduced the back-propagation algorithm that has been widely used for practical applications. His other contributions to neural network research include Boltzmann machines, distributed representations, time-delay neural nets, mixtures of experts, Helmholtz machines, and products of experts. His current main interest is in unsupervised learning procedures for deep neural networks with rich sensory input.