Structured Output Layer Neural Network Language Models for Speech Recognition

Hai-Son Le, Ilya Oparin, *Member, IEEE*, Alexandre Allauzen, Jean-Luc Gauvain, *Member, IEEE*, and François Yvon

***Abstract*—This paper extends a novel neural network language model (NNLM) which relies on word clustering to structure the output vocabulary: Structured OUtput Layer (SOUL) NNLM. This model is able to handle arbitrarily-sized vocabularies, hence dispensing with the need for shortlists that are commonly used in NNLMs. Several softmax layers replace the standard output layer in this model. The output structure depends on the word clustering which is based on the continuous word representation determined by the NNLM. Mandarin and Arabic data are used to evaluate the SOUL NNLM accuracy via speech-to-text experiments. Well tuned speech-to-text systems (with error rates around 10%) serve as the baselines. The SOUL model achieves consistent improvements over a classical shortlist NNLM both in terms of perplexity and recognition accuracy for these two languages that are quite different in terms of their internal structure and recognition vocabulary size. An enhanced training scheme is proposed that allows more data to be used at each training iteration of the neural network.**

***Index Terms*—Automatic speech recognition, neural network language model, speech-to-text.**

I. Introduction

HAVING been used for several decades, n-gram models still form the basis of modern language modeling for speech-to-text (STT) transcription. Despite numerous attempts there are very few approaches that have been shown to systematically and significantly improve over well-estimated n-gram language model (LM) baselines. Neural network language models (NNLMs) are among these approaches and have been adopted in some state-of-the-art STT systems [1]–[3].

Neural network LMs were introduced in [4], [5] as a means to improve discrete models. Standard n-gram back-off LMs rely on a discrete representation of the vocabulary, where each word is associated with a discrete index. In contrast, NNLMs are based on the idea of a continuous word representation, where each word is associated with a real-valued feature vector. In this continuous space, distributionally similar words are neighbors. Thus n-gram distributions are expressed as a smooth function of the word representation, and can take into the account underlying similarities between words. A neural network jointly estimates both the word representations in a continuous space and the associated probabilities.

An important specificity of NNLMs is the capability to take into account longer n-gram contexts. Previous experiments at LIMSI with standard n-gram LMs and large setups indicated that the gain obtained by increasing the n-gram order from 4 to 5 is almost negligible despite a drastic increase in model size. Handling such models is thus rather impractical and can hardly be done without pruning. For NNLMs, increasing the word context length at the input layer results in at most a linear growth in complexity [1] and does not lead to any prohibitive computational or memory overhead.

The major bottleneck with NNLMs is the computation of posterior probabilities in the output layer, which must contain one unit for each word in the vocabulary. As the softmax function is used to obtain the posterior probabilities, the summation over the entire vocabulary is required for each word in the training or test data. This makes the handling of large vocabularies too complex since it would require prohibitive computation time. As a practical workaround, NNLMs usually estimate probabilities only for the fraction of the vocabulary consisting of the most frequent words which is called a *shortlist*. Probabilities of all n-grams finishing with an out-of-shortlist (OOS) word are estimated with a conventional n-gram LM. Such a restriction limits the potential of NNLMs.

This article extends the *Structured OUtput Layer* (SOUL) model introduced in [6], [7], describing training and clustering issues in detail. The SOUL approach combines the benefits of neural networks and class-based LMs by structuring the vocabulary by means of a clustering tree automatically induced from the continuous word representation. In contrast to standard NNLMs, SOUL NNLM makes it feasible to estimate n-gram probabilities for large vocabularies. As a result, all vocabulary words, and not just the words in a shortlist, can benefit from the improved prediction capabilities of the NNLM.

Since estimating word probabilities for all vocabulary entries can be regarded as one of the major distinctive features of the SOUL NNLM, it is natural to evaluate and compare their performance to more traditional NNLMs for different vocabulary sizes.

Manuscript received February 17, 2012; revised May 25, 2012 and July 24, 2012; accepted August 03, 2012. Date of publication August 28, 2012; date of current version nulldate. This work was supported in part under the research program Quaero, funded by OSEO, the French State agency for innovation and in part under the GALE program of the Defense Advanced Research Projects Agency, Contract HR0011-06-C-0022. Any opinions, findings or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gokhan Tur.

H.-S. Le, A. Allauzen, and F. Yvon are with the Université Paris-Sud and LIMSI-CNRS, Orsay, France ***[AUTHOR: Please provide postal codes for each address.]*** (e-mail: lehaison@limsi.fr; allauzen@limsi.fr; yvon@limsi.fr).

I. Oparin and J.-L. Gauvain are with LIMSI-CNRS, Orsay, France (e-mail: oparin@limsi.fr; gauvain@limsi.fr).

Digital Object Identifier 10.1109/TASL.2012.2215599

Speech recognition experiments were carried out to assess the performance of the SOUL NNLMs for the Mandarin Chinese and Arabic (Modern Standard Arabic) languages using data and systems developed for the GALE program. The baseline STT systems [8], [9] have vocabularies of different sizes: 56 k words for Mandarin (including all characters) and 300 k MADA-decomposed entries for Arabic. These systems make use of well-tuned 4-gram LMs trained on corpora containing several billion words of text (without any pruning or cut-offs) interpolated with standard shortlist NNLMs.

It was shown that the training of full vocabulary NNLMs is computationally feasible with the SOUL architecture, and that this leads to improved STT accuracy for large tasks. In this paper the clustering issues and the schemes for training the class model embedded within the SOUL NNLM are described in detail. Additional experiments dealing with the investigation of the influence of different parameters on SOUL NNLM performance are also reported. Examples of classes formed by words that are close according to the SOUL NNLM projection space are presented.

The remainder of this paper is organized as follows. Related work on hierarchical neural networks is summarized in Section II, followed by a description of the architecture of SOUL NNLMs in Section III. Section IV describes the experimental setup and the baseline STT systems, with the experimental results given in Section V. Finally, Section VI concludes the paper along with providing a discussion of the main findings.

II. Related Work

A number of techniques are conventionally used to make the training of NNLMs computationally feasible on the very large corpora used to develop state-of-the-art STT systems.

As the time needed to train a NNLM on all available data is usually prohibitive, a resampling technique was used to partially circumvents this bottleneck [1]. Limited amounts of data are selected at each neural network training iteration by sampling the original corpus. This resampling is usually biased towards in-domain data. However, resampling alone is not sufficient to train NNLMs with large recognition vocabularies as the output layer is also prohibitively large. A proposed solution to this problem was to restrict the output vocabulary to a shortlist of several thousand most frequent words [5].

Recently, some new methods for optimizing NNLM training have been proposed. For example, a significant speed-up in training without performance degradation can be obtained by using a small hidden layer in combination with direct connections between the input and the output layers. Such a model with only 40 neurons in the hidden layer was reported to be much faster to train and to provide the same results as a model with 320 neurons in the hidden layer but without direct connections [10]. Another recent study [11] showed the impact of the initialization of the NNLM parameters and proposed three new training schemes.

A crucial issue that arises with the use of shortlists is that the NNLMs estimate the probabilities only of a limited number of words. Therefore the probability distribution must be normalized with a standard back-off LM in order to cover words that are not in the shortlist. The formula for calculating the normalized probabilities with shortlist NNLMs is:

$$P(w_t|h_t) = \begin{cases} \hat{P}_N(w_t|h_t) \cdot \alpha(h_t) & \text{if } w_t \in \text{shortlist} \\ \hat{P}_B(w_t|h_t) & \text{otherwise} \end{cases} \quad (1)$$

where w_t is the word to be predicted and h_t its $n-1$ word history, $\hat{P}_N$ is the probability of an in-shortlist word calculated with the NNLM, $\hat{P}_B$ is the probability assigned by the standard backoff n-gram LM. The scaling factor $\alpha(h_t)$ depends on the history h_t and was defined in [5] as:

$$\alpha(h_t) = \sum_{w \in shortlist} \hat{P}_B(w|h_t), \quad (2)$$

As both $\hat{P}_N$ and $\hat{P}_B$ are normalized to sum to one, the "combined" conditional distribution P is also normalized.

To handle large output vocabularies, [12] and [13] proposed to use a hierarchical structure for the output layer following previous research to speed-up systems using maximum entropy models. It should be noted that the idea to use classes to factorize the output layer of a neural network can be traced to the classical work of Brown *et al.* on distributed word representation [14].

A conditional maximum entropy model in its general form is expressed as

$$P(w_t|h_t) = \frac{\exp\left(\sum_j \lambda_j f_j(w_t, h_t)\right)}{\sum_{w \in V} \exp\left(\sum_j \lambda_j f_j(w, h_t)\right)} \quad (3)$$

where f_j are the features, λ_j are the feature weights. The normalization term in the denominator requires a summation over all in-vocabulary words $w \in V$. This poses exactly the same computational problems as the softmax in the output layer of neural networks. In [15] it was proposed to first cluster words into classes and then compute the conditional probabilities:

$$P(w_t|h_t) = P\left(c(w_t)|h_t\right) \cdot P\left(w_t|h_t, c(w_t)\right), \quad (4)$$

where $c(w)$ denotes the class assigned to a word w. Two models are trained separately: one that predicts the class probability given the history; and one predicting a word given its class and its history. With this kind of model, a significant speed-up can be obtained since the required summations for both models are drastically reduced: for the first model, the summation ranges over the number of different classes, whereas for the second model it involves only the words in the given class.

The idea of clustering the vocabulary words was introduced for NNLMs in [12], where a binary hierarchical clustering was derived from the WordNet semantic hierarchy. In that work, the output vocabulary was first clustered and represented by a binary tree. Each internal node of the tree held a word cluster which was further divided into two sub-clusters and so on. The leaves correspond to the words at the end of this representation of the vocabulary. The neural network aims to estimate probabilities for the paths in this binary tree given the history, rather than for the word itself, as reflected by the following equation:

$$P(w_t|h_t) = \prod_{j=1}^{m} P\left(b_j(w_t)|b_1(w_t)\dots b_{j-1}(w_t), h_t\right) \quad (5)$$

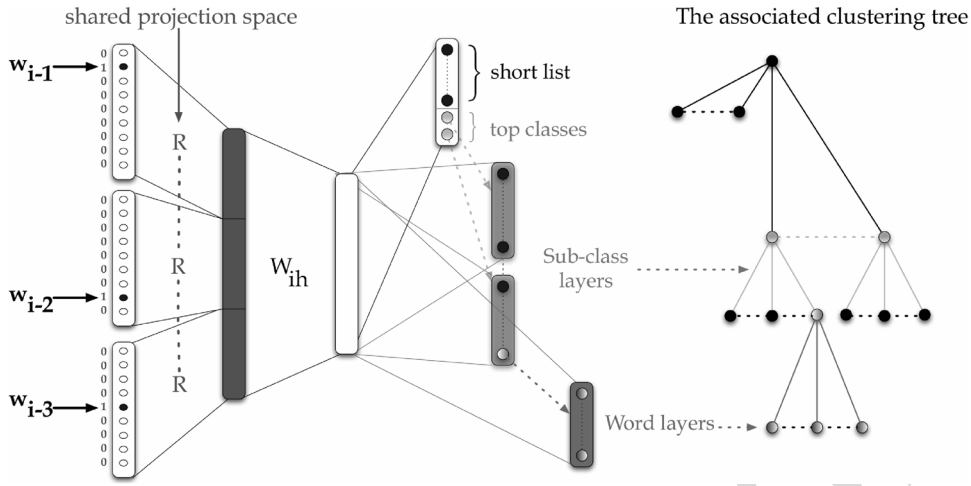

Fig. 1. The architecture of the structured output layer neural network language model.

where m is the depth of the tree and b stands for 1 or 0 in the path for a given word. This approach resulted in a two order of magnitude speed-up on a small data set (1 million words of the Brown corpus) with a vocabulary of 10 k words. However, a significant increase in perplexity was observed relative to a standard NNLM. One of the possible explanations is the widely observed difficulty of integrating linguistic information in statistical LMs and in the NNLM described above the clustering tree is constructed using the semantic information from the WordNet. Alternative solutions that do not rely on external linguistic sources were proposed in [13], [16] for neural network and log-bilinear models. These methods are also appealing, as the WordNet or similar resources covering large output vocabularies are not necessarily available for different languages. Using automatic clustering to factorize the output layer resulted in significant speed-ups in training and perplexities similar to those obtained without clustering.

It can be seen that much of the previous work concentrated on reducing the computational complexity, with less emphasis on improving the accuracy of speech recognition or machine translation systems. In the binary hierarchical log-bilinear approach the resulting binary clustering tree is deep (i.e. has many levels), and thus faces a data fragmentation problem. This problem is well known for decision trees, especially for language modeling tasks [17]. Moreover, if a word is assigned to a wrong class at some level in a decision tree, this error affects all the internal nodes (or clusters) leading to this word. This is typically the case for rare words that represent a large fraction of the vocabulary. Thus an error in one word may have a significant impact on the whole system. By relaxing the constraint of the binary structure, it is expected that this shortcoming will be overcome as explained in Section III.

Factorization of the output layer was also used for recurrent NNLMs. Although a simpler approach based on the distribution of unigram probabilities that does not reveal any relationships between the words was used, this work illustrates the growing interest in improving neural network LMs for STT [18]. Another active direction of research dealing with different clustering methods is connected with recently proposed Model M and its enhancements [19], [20].

III. Structured Output Layer Neural Network Language Model

In this section the class-based neural network language model, namely Structured OUtput Layer (SOUL) NNLM, is described in detail.

In the SOUL NNLM [6], [7] the output vocabulary is structured by a clustering tree, where each word belongs to only one class and ends up in a single leaf of the tree. If w_i denotes the i^{th} word in a sentence, the sequence $c_{1:D}(w_i) = c_1, \ldots, c_D$ encodes the path for word w_i in the clustering tree, and D is the depth of the tree. $c_d(w_i)$ is a class or sub-class assigned to w_i, and $c_D(w_i)$ is the leaf associated with w_i (the word itself). Then the n-gram probability of w_i given its history h can be estimated as follows:

$$P(w_i|h) = P\left(c_1(w_i)|h\right) \prod_{d=2}^{D} P\left(c_d(w_i)|h, c_{1:d-1}\right) \qquad (6)$$

The probability of a word is conditioned not only on its context, but also on a non-binary string of indices encoding a path in the clustering tree. Each word belongs to only one class and there is a softmax function at each level of the hierarchical representation.

Fig. 1 represents the architecture of the SOUL NNLM. Both SOUL and standard NNLMs share the same architecture up to the hidden layer. The output structure of the SOUL model is made of three kinds of output layers. The first output layer estimates the distribution over the shortlist words and the top (most general) classes for the OOS words. Words in the shortlist are regarded as a special case since each represents its own class without sub-clustering (the tree depth D is equal to 1 in this case). The top classes serve as the roots of trees to handle the OOS words and include multiple sub-class layers each with a softmax function, forming the second kind layers. Finally, the word layers estimate the word probabilities for the OOS words. To summarize, the SOUL model estimate the n-gram probability of a word w_i given its context h as defined by the (6) as follows: the upper layer handles the distribution over the top classes $P(c_1(w_i)|h)$; then subsequent sub-class layers compute the sub-class probabilities $P(c_d(w_i)|h, c_{1:d-1})$ for $1 < d < D$;

finally the OOS word probabilities $P(c_D(w_i)|h, c_{1:D-1})$ are given by the word layers.

The SOUL model can handle any arbitrary tree structure of the vocabulary. A slightly different architecture could be considered, where the first softmax layer consists of classes corresponding to the shortlist words plus an additional node. This additional node serves as a root of a tree that includes all sub-class layers needed to estimate the probabilities of the OOS words. Experiments with this architecture showed slightly worse results with a negligible complexity reduction.

A. Training Scheme

The training scheme addresses two closely related parts of the SOUL model: the parameters estimation and the word clustering that structures the output vocabulary. In [13] a rather sophisticated divisive clustering algorithm is presented which is based on a mixture of two Gaussians applied to the continuous word space. In this work, we propose a more straightforward method based on the relationship between the two word spaces defined in a standard NNLM [11], i.e. the context and the prediction spaces. The training procedure is summarized as follows:

Step 1) **pre-training**: Train an NNLM with a shortlist as output, using the one vector initialization method [11]. This step is only required to provide a preliminary estimate of the projection space defined by the matrix R.

Step 2) **dimension reduction of the projection space**: Apply the standard Principal Component Analysis (PCA) on the matrix R.

Step 3) **word clustering**: Perform a divisive top-down word clustering based on the K-means algorithm, using the continuous representation induced by the previous step for words that are not in the shortlist.

Step 4) **full training**: Train a full-vocabulary NNLM with the tree structure as output.

The goal of step 1 is to learn the word features for clustering. In the one word initialization method introduced in [11], all the words in the context vocabulary are initially projected onto the same (random) point in the context space. The intuition is that it will be easier to build meaningful neighborhoods, especially for rare types, if all words are initially considered similar and only diverge if there is sufficient evidence in the training data to suggest that they should. With this kind of initialization, only a few iterations are required as the learning of word features converges quickly.

As words input to a NNLM are represented in 1-of-K coding with 1 corresponding to a word's vocabulary index and all other elements set to zero, each line in the projection matrix R corresponds to a continuous representation of a particular word. The dimensionality of this representation is reduced with the PCA at step 2. The size of the context space after the dimensionality reduction is equal to 10 for the experiments described here. Such low dimensional context space makes the subsequent K-means clustering at the step 3 computationally cheap.

The clustering at step 3 divides a word class (or sub-class) only if the number of words in this class is above an empirically determined threshold W. Each class containing more than W words is divided in $|\sqrt{W}+1|$ sub-classes. The value of this threshold allows to tune the depth of the clustering tree: small values result in a deep clustering structure while a very large value generates a flat tree. This threshold depends on the vocabulary size and can be optimized. However, experimental results with different vocabulary sizes show that its value does not have much influence on the overall performance of the STT systems and three levels of hierarchy seem sufficient. The clustering deals with rare words and thus there is little point in making small classes and having a deep tree structure. In these experiments, the K-means algorithm starts with 4 k top classes. Then each sub class is recursively divided. However, in practice, the one vector initialization used at step 1 implies that very rare words are represented by very similar or identical feature vectors. In that particular case, the rare words are then naturally grouped in the same class by the first K-means step, and a recursive subdivision is therefore not well suited. In the rest of this paper, the class for rare words is randomly divided.

Finally, standard back-propagation training [21] is performed for the full vocabulary SOUL NNLM at step 4 using the parameters obtained at step 1 for initialization.

In order to evaluate the influence of the type of clustering in the SOUL architecture, the clustering proposed above was compared with two other techniques. The first is the widely known Brown clustering algorithm based on the maximization of the mutual information between adjacent classes [14]. This algorithm has served as a baseline in the community and it has been hard to improve upon, even with much more sophisticated methods. The second method has been implemented in the BUT RNNLM recurrent network toolkit [18], and assigns words to classes according to their unigram probabilities.

Comparative results for the three clustering methods, different shortlist sizes, number of classes and depth of the clustering tree are given and discussed in Section V.

B. Enhanced Training Scheme

NNLM training maximizes the log-likelihood of the training data. This optimization is performed by stochastic back-propagation [21]. A previously trained standard NNLM is used to initialize the shared parameters (i.e matrices R and W_{ih} in Fig. 1). The other parameters are initialized in a usual way, as described in the previous section. Overall, the training time for a SOUL model is only about 30% longer than for standard 8 k shortlist NNLMs and slightly shorter the time to train a 12 k shortlist NNLM.

When dealing with large vocabularies (e.g. the Arabic one in this study), the number of parameters related to the class-part of the model increases significantly. As a result, the resampling technique that is used for neural network training at each epoch may be insufficient to obtain robust parameter estimates.

As described in Section III, the output layer of the SOUL NNLM is comprised of two parts: the first layer which directly models the probabilities of the most frequent in-shortlist words and top classes, and the remaining sub-class layers that deal with the classes of OOS words as illustrated in Fig. 1.

The parameters related to the first output layer are updated for all training examples since it covers the shortlist words and the most general classes for the less frequent words. The n-grams

ending with in-shortlist words are used to update the parameters only of this first layer, leaving sub-class layers intact. The parameters of sub-class layers are updated with the n-grams ending in an OOS word and only those class layers leading to the leaf with this particular word are activated and the corresponding parameters are updated. A shortlist usually covers a large part of training examples so that the updates of the parameters related to sub-class layers are less frequent. Moreover, when such an update occurs, it is done for a subset of the parameters corresponding to a particular path in the clustering tree of the class layers. At the same time the number of parameters corresponding to OOS words is much larger. As a result, the two parts of the SOUL output layer are not trained equally well. Therefore a modified SOUL training scheme based on the separate training of the OOS part at the output layer is proposed. This scheme adds an additional step to the training procedure described in Section III-A.

This additional step 3′ is similar to 1, but is carried out only for OOS words. At this step the n-grams ending with shortlist words are skipped and the parameters associated with shortlist words are temporarily fixed, reducing the size of the first output layer to the number of main classes of infrequent words only (4 k as opposed to 12 k in the SOUL setup). As explained above, the softmax in the first output layer is triggered for each training example, which, in turn, requires summation over all nodes in this layer. Thus, any reduction in the size of the first output layer results in a significant speed-up. Since the number of OOS occurrences represent a relatively small proportion of the training data (for our experiments on Chinese and Arabic data these represent 5–10% of all word occurrences), the number of training examples can be easily increased by the factor of 10. As NNLMs are trained (with resampling) on less data as compared with the baseline n-gram LMs, there are always additional data available for the step 3′. The parameters obtained at step 3′ are used to initialize the parameters associated with the OOS words at the step 4.

IV. Experimental Setup

The performance of the SOUL NNLM is compared to standard shortlist NNLMs on the GALE Arabic and Mandarin Chinese tasks.

The general parameters of NNLMs are presented in Table I. Several NNLMs differing in the size of the projection layer and the hidden layer are trained for each task, and then interpolated together, along with the standard N-gram backoff LM. Interpolation weights are tuned to minimize perplexity on the development data. The same setups are used for both the SOUL NNLMs and the shortlist NNLMs. Thus the difference in results can be attributed to the use of the whole vocabulary at the output (which, in turn, uses the class information).

Each NNLM is trained on about 25–30 M words at each iteration after resampling of the training data. The enhanced SOUL NNLM training scheme (see Section III-A) is investigated on Arabic, for which shortlists of similar sizes provide lower data coverage than for Mandarin thus leaving more space to improvement with the proposed scheme. Up to 300 M words are used during step 3′ to train the class output layers that deal with OOS words. As all the n-grams ending with an in-shortlist word are skipped at step 3′, it makes about 30 M n-grams that are used to update the parameters.

TABLE I
Parameters of the Mandarin Chinese and Arabic NNLMs. For Each Language, NNLMs With Different Parameters are Subsequently Interpolated Together

	Mandarin				Arabic		
	NN1	NN2	NN3	NN4	NN1	NN2	NN3
initial learning rate	5×10^{-3}				5×10^{-3}		
learning rate decay	5×10^{-8}				5×10^{-8}		
weight decay	3×10^{-5}				3×10^{-5}		
projection layer	500	220	250	300	200	300	400
hidden layer	200	430	450	500	500	400	300

To assess the impact of the increase in the shortlist size, NNLMs with 8 k and 12 k shortlists are trained on the Mandarin data. As slightly better results are obtained with a larger one, the Arabic shortlist NNLMs use a shortlist of 12 k MADA-decomposed units.

In order to study possible improvements from using longer span NNLMs, the increase in context length from 3 (that corresponds to 4-grams) to 5 is investigated. For the shortlist NNLMs the same 4-gram back-off LM is used for OOS words.

The performance of the different Mandarin models was assessed on the GALE dev09_M and eval09_M data sets containing broadcast news and broadcast conversations. A subset of dev09_M called dev09s_M was also defined, comprised of about a third of dev09_M data. The dev09_M, dev09s_M and eval09_M sets contain respectively 97459, 31529 and 79246 segmented words. Three sets are used to evaluate the performance of the different Arabic models, namely dev09s_A, eval10ns_A and dev10c_A. These sets contain of 23576, 45629 and 52181 MADA-decomposed units respectively.

A. Mandarin STT System

Mandarin Chinese is a language with a low morpheme-per-word ratio. Most words in a running text are composed of a single morpheme (character), and, as there are no markers neither for inflection nor for parts of speech, it has a fixed word order. Words in written Chinese are not separated by white spaces. A natural solution is either to use character-based LMs or to perform word segmentation as a pre-processing step. The former was shown to be inferior to the latter [22], so the longest-match segmentation approach is taken in this work. As is the convention, the character error rate is used to evaluate recognition performance for Chinese.

The recognition vocabulary contains 56 k entries including both multicharacter words (about 50 k) and individual Chinese characters (6 k entries). There are no out-of-vocabulary (OOV) words for Chinese. The acoustic models and the decoding process of the Mandarin STT system are described in detail in [8]. The Mandarin LM is trained on 3.2 billion word tokens after segmentation. Individual 4-gram LMs are first built for each of 48 sub-corpora without any cut-offs and pruning. These models are smoothed according to the unmodified interpolated Kneser-Ney scheme and are subsequently linearly interpolated to form the baseline 4-gram LM, with the weights tuned on the dev09_M data. This resulting model includes 2.2 billion unique n-grams.

TABLE II
PERPLEXITY AND CER (%) FOR DIFFERENT MANDARIN LMS

model	ppl dev09_M s/a	ppl dev09_M int	CER dev09s_M	CER eval09_M
Kneser-Ney 4g	211		9.8%	8.9%
+ shortlist 8k NN 4g	229	187	9.5%	8.6%
+ shortlist 12k NN 4g	227	185	9.4%	8.6%
+ SOUL NN 4g	221	180	9.3%	8.5%
+ shortlist 8k NN 6g	214	177	9.4%	8.5%
+ shortlist 12k NN 6g	207	172	9.3%	8.5%
+ SOUL NN 6g	**192**	**162**	**9.1%**	**8.3%**

TABLE III
PERPLEXITY FOR DIFFERENT ARABIC LMS

LM type	dev09s_A s/a	dev09s_A int	eval10ns_A s/a	eval10ns_A int	dev10c_A s/a	dev10c_A int
Kneser-Ney 4g	312		239		256	
shortlist 12k NN 4g	324	276	247	213	256	224
SOUL NN 4g	293	256	225	200	231	208
SOUL+ NN 4g	277	250	214	195	221	204
shortlist 12k NN 6g	302	263	228	202	236	210
SOUL NN 6g	255	231	196	180	200	186
SOUL+ NN 6g	**245**	**227**	**189**	**177**	**194**	**183**

B. Arabic STT System

Arabic is a highly inflective and morphologically rich language characterized by a large number of word forms for a given lemma. This usually results in vocabularies that are several times larger than the ones used for Chinese or English.

The baseline Arabic STT system used in the speech recognition experiments reported here gives state-of-the-art performance on the GALE tasks [9]. The Arabic vocabulary contains 300 k entries. The MADA[1] (Morphological Analysis and Disambiguation for Arabic) tool is used to decompose the words into their morphological constituents, increasing lexical coverage and improving recognition performance [23], [24]. The Arabic LM training data contains about 1.7 billion words before decomposition, resulting in a total of about 2 billion morphs. 4-gram LMs are trained for 32 different MADA-decomposed text subsets using the unmodified interpolated Kneser-Ney scheme, without pruning or cut-offs. These LMs are further interpolated to form the final 4-gram LM. The resulting model includes 1.2 billion unique n-grams.

V. EXPERIMENTAL RESULTS

In this section, experimental results are provided for Arabic and Mandarin, both in terms of perplexity and recognition error rates. The internal structure of these two languages is quite different which is reflected by the vocabulary sizes of their STT systems. The SOUL word clustering approach is also compared to the classical Brown clusters and the more straightforward unigram algorithm. The representation of words in the projection space is explored by finding the neighbors for some selected words.

A. Perplexity Results

Tables II and III summarize the results in terms of perplexity for Mandarin and Arabic respectively. The results are provided both for stand-alone NNLMs (columns *s/a*) and after interpolation with the baseline 4-gram LMs (columns *int*).

The rows marked with *shortlist* correspond to shortlist NNLMs (with 8 k or 12 k shortlists). The context size for the NNLMs is indicated as 4 g or 6 g. The models marked as *SOUL* are based on the general SOUL architecture, while $SOUL+$ corresponds to the SOUL NNLM that uses the enhanced training scheme for the parameters dealing with OOS words, as described in Section III-B.

As can be seen from Table II, increasing the shortlist size by 50% from 8 k to 12 k words brings only a small improvements in perplexity. The SOUL model that predicts probabilities for all words in the vocabulary and uses word clustering for infrequent words systematically outperforms the 12 k shortlist NNLM for both languages even though it is not more computationally demanding.

Using a longer context (6 g vs. 4 g) reduces the perplexity both for the shortlist and the SOUL NNLMs. These models directly provide lower perplexities than the Kneser-Ney LM, trained on much more data. All interpolations of different orders and types of NNLMs with the Kneser-Ney LMs were found to significantly reduce the perplexity.

These results show that the SOUL NNLM consistently outperforms the shortlist counterparts of the same order on all the test sets. For the 4-gram stand-alone NNLMs, the relative improvement obtained with the SOUL NNLM over the shortlist NNLM is 3% for Mandarin and 9–10% for Arabic (13–15% for SOUL+). In the longer-context 6-gram case, the gains with the SOUL NNLM are somewhat larger, 7% for Mandarin and 14–16% for Arabic (17–19% for the SOUL+). The same tendency holds for the NNLMs interpolated with the Kneser-Ney LMs. For the 4-gram interpolated models the improvement with the SOUL NNLM is 3% for Mandarin and 6–7% for Arabic (8–9% for the SOUL+). For 6-gram interpolated models the relative gains are 8% for Mandarin and 11–12% for Arabic (12–14% for the SOUL+).

The difference in perplexity reduction between the enhanced SOUL NNLM training and the standard SOUL NNLM is smaller after interpolation with the Kneser-Ney LMs. This suggests that the advantage of using 10 times more data to train the part of SOUL NNLMs that deals with rare words should not bring much benefit in state-of-the-art STT systems where NNLMs are interpolated with n-gram LMs.

B. Speech Recognition Results

Tables II and IV summarize the results of speech recognition experiments in terms of character error rate (CER) for Mandarin and word error rate (WER) for Arabic. The lattices generated with the baseline 4-gram Kneser-Ney LMs are rescored with the different models types. The lattice rescoring is performed in a usual way, by extracting n-grams from the lattice and estimating their probabilities with NNLMs. The recognition results with NNLMs are reported after interpolation with the baseline

[1]http://www1.ccls.columbia.edu/~cadim/MADA.html.

TABLE IV
WER (%) WITH DIFFERENT ARABIC LMS

LM type	dev09s_A	eval10ns_A	dev10c_A
Kneser-Ney 4g	14.8	9.6	14.5
+ shortlist 12k NN 4g	14.4	9.1	14.2
+ SOUL NN 4g	14.3	9.0	14.0
+ SOUL+ NN 4g	14.1	9.1	14.0
+ shortlist 12k NN 6g	14.3	9.1	14.2
+ SOUL NN 6g	**14.0**	**8.9**	14.0
+ SOUL+ NN 6g	**14.0**	**8.9**	**13.9**

TABLE V
NNLM WEIGHTS FOR INTERPOLATION WITH THE BASELINE n-GRAM LMS

NNLM model type	NNLM interpolation weight Mandarin	Arabic
shortlist 12k NN 4g	0.53	0.50
SOUL NN 4g	0.60	0.68
SOUL+ NN 4g	-	0.72
shortlist 12k NN 6g	0.60	0.55
SOUL NN 6g	0.69	0.74
SOUL+ NN 6g	-	0.75

4-gram Kneser-Ney LMs, the same as used to generate 4-gram lattices.

To give the idea of lower bounds of possible WER reductions, e.g. on the Mandarin dev09_M set, the oracle CER is about 4%. It should be noted that the exact calculation of oracle WERs is not straightforward on GALE setups due to their specificities (reference transcription is divided into "snipets", normalization, segmentation of Mandarin data into words, MADA decomposition for Arabic). Thus, oracle calculation is approximative and may exhibit differences as compared to the way one-best hypothesis scores are calculated by the NIST *sclite* tool.

The interpolation weights for different NNLMs with the baseline Kneser-Ney LMs are presented in Table V. It can be seen in this table that higher interpolation weights are obtained for the SOUL NNLMs than the shortlist NNLMs.

The results in Table II for Mandarin and Table IV for Arabic show that the improvements in perplexity attained with the SOUL NNLMs compared with the shortlist NNLMs carry over to speech recognition. The best recognition performance is obtained with the 6-gram context NNLMs.

As compared to the Kneser-Ney LMs, the SOUL NNLM, when interpolated with the latter, improves the WER by up to 0.7% absolute for Mandarin and 0.8% for Arabic. The SOUL model brings an absolute error reduction of about 0.2–0.3% more than the shortlist NNLM.

It should be noted that the gains from using 6-gram NNLMs on Arabic are smaller than might be expected since for computational reasons the lattices had to be pruned before rescoring with the 6-gram model. The effect of pruning is most notable on the dev10c_A set which contained some large lattices that were subject to severe pruning. However, since the 6-gram shortlist NNLMs showed no improvement with pruned lattices over 4-gram NNLMs, the 6-gram SOUL NNLMs still improve the results.

C. NNLM Configurations

In order to investigate the impact of different NNLM parameters, such as the size of the shortlist, the number of top classes in the first output layer and the depth of the clustering tree, a number of additional experiments were carried out on the Chinese setup. In these experiments, only one NNLM (as opposed to three or four in the experiments described in the previous sections, see Table I) with 300 nodes in the projection layer for each history word and 500 nodes in the hidden layer is trained for each configuration.

TABLE VI
PERPLEXITY FOR LMS WITH DIFFERENT SHORTLIST SIZES ON THE MANDARIN DEV09_M SET

model shortlist	top classes	depth	ppl s/a	int	training time (days)
Kneser-Ney 4g					
-	-	-	211		-
shortlist NNLMs 6g					
8k	-	-	222	178	2.4
12k	-	-	222	175	3.5
25k	-	-	223	171	7.1
SOUL NNLMs 6g					
8k	4k	3	220	169	3.1
12k	4k	3	219	168	5.6
25k	4k	3	219	168	7.0
flat full-vocabulary NNLMs 6g					
all (56k)	0	1	226	171	14.7

Results with NNLMs with different sizes of the shortlist are presented in Table VI. *Shortlist* column corresponds to different sizes of the shortlist part, *top classes* reports the number of top classes of the first SOUL output layer, *depth* is the depth of the SOUL clustering tree, *s/a* stands for stand-alone NNLMs and *int* for NNLMs interpolated with the baseline n-gram model.

One conclusion from Table VI is that the flat full-vocabulary NNLM, while being computationally very expensive, does similarly or worse than the ones that make use of a shortlist. The SOUL NNLMs benefit from clustering of rare words by means of a clustering tree (shortlist NNLMs back off to normalized Kneser-Ney estimates in this case) at the output, as seen from the comparison with the full-vocabulary flat model. The SOUL NNLMs also deliver top results with a relatively small shortlist (i.e. 8 k). This may be important in order to save computation time and resources, as it is not necessary to train SOUL NNLMs with large shortlists. For example, running similar experiments with shortlists equal or close to the vocabulary size is hardly feasible on larger vocabulary setups (56 k Mandarin vocabulary size can be considered as very moderate), as e.g. Arabic with 300 k vocabulary entries, because of the prohibitive training costs.

A natural question is then whether stand-alone NNLMs outperform n-gram models. This question is difficult to answer on the GALE setups, since it is infeasible to train NNLMs on the same amounts of data as the Kneser-Ney LM baseline; as pointed out in Sections II and IV, resampling has to be used for NNLMs. A fair comparison is however possible on smaller setups. The reader may find a discussion for which type of events NNLMs do better that n-gram LMs and vice versa in [25].

Table VII reports perplexity for Mandarin 6-gram SOUL NNLMs with different numbers of top classes in the first output layer, SOUL NNLMs with clustering trees of different depths and a full-vocabulary SOUL NNLM (no shortlist is

TABLE VII
PERPLEXITY FOR 6-GRAM SOUL NNLMS WITH DIFFERENT NUMBER OF TOP CLASSES AND CLUSTERING TREE DEPTHS ON THE MANDARIN DEV09_M SET

model			ppl		training time
shortlist	top classes	depth	s/a	int	(days)
SOUL NNLMs with different number of top classes					
8k	128	3	221	169	2.4
8k	256	3	218	168	3.5
8k	1k	3	220	169	3.6
8k	2k	3	220	169	2.7
SOUL NNLMs with different clustering tree depth					
8k	4k	3	220	169	3.1
8k	4k	2	220	169	3.0
all	0	1	226	171	14.7
SOUL NNLM without shortlist part					
0	4k	2	242	176	2.2
0	4k	3	240	176	2.4

used, all words are clustered). It can be seen that the number of top-level classes does not have much influence on the final perplexity. The same holds for the depth of the clustering tree. On one hand, training with a flat tree is slow and it yields higher perplexity. On the other hand, there is little difference between the clustering tree of depth two and three. This can be explained by the fact that clustering mostly concern rare words; there is thus little point to perform a deep and fine-grained clustering. However, deeper clustering trees are expected to provide training speed-ups for larger vocabulary tasks as it results in more numerous but smaller softmax layers (see discussion in Section II). A similar experiment on Arabic showed that a three-level tree indeed provides gains in training time as opposed to the two-level one (5.8 days vs. 6.2 days).

The benefit of using the shortlist part in the SOUL architecture as described in Section III-B was also verified. The SOUL NNLM representing the whole vocabulary as a clustering tree was trained. It is reported in the last row of Table VII, showing that in the SOUL architecture, frequent words should be treated separately.

D. Clustering Within SOUL NNLMs

According to the SOUL architecture, the 8 k most frequent words do not undergo clustering as they form classes on their own. Only the remaining words are clustered into 4 k classes on the top level. As described in Section III-A, the training steps (1 to 3) of the SOUL model are used to derive the word clustering. The depth of clustering hierarchy equals to 3.

With the Brown (as in [14]) and the unigram clustering (see Section III-A), models can be directly trained with step 4 within the SOUL architecture. This scenario is referred as *single-step* in Table VIII, where the results obtained with different word clustering schemes are given. The original SOUL clustering procedure with all the training steps described in Section III-A is referred as *SOUL*. In this *SOUL* scenario neural networks based on the *Brown* and the *Unigram* clustering also benefit from the information obtained during steps 1 (e.g. lookup tables) and 3′ (using more data to estimate probabilities of OOS words). The original clustering method based on word similarity in continuous space in the neural network is referred as NN. It should be noted that both Brown and unigram approaches provide clustering tree structures, just as the original NN method.

TABLE VIII
STAND-ALONE SOUL NNLM PERPLEXITY WITH DIFFERENT CLUSTERINGS ON THE MANDARIN DEV09_M SET

Clustering type	4-gram		6-gram	
	single-step	SOUL	single-step	SOUL
Unigram	259	248	229	222
Brown	253	245	225	218
NN	-	245	-	220

TABLE IX
EXAMPLES OF CLOSE MADA-DECOMPOSED ARABIC WORDS ACCORDING TO THE SOUL NNLM PROJECTION SPACE

headword	close words
Aldwlp	AlHwmp - bnwknA - mWsstnA - jAmctkm - mSArfhA - bldytnA - nqAbth - vwrtnA - cAlqDyp - cAlclAqAt - mSArfnA - dwltkm - Aldwylp - wzArthA - mHAkmnA
country	district - our banks - our institutions - your universities - its banks(f) - our town - syndicate - our war - about affair - about relations - our banks - your country - small country - its ministry(f) - our tribunals
jAC	yjyC - EtY - yEty - wAtY - mAjAC - mAwrd - yEtyAn - jACA - tEtY - jAChA - jACtA - Astwqfny - AstcjlwA - ycksh - wkAn - yje - ytbdY
arrived	will arrive - come - he will come - and come - he didn't come - destination - they will come - they arrived - she will come - smb/smth(m) came/happened to her - smb/smth(f) came/happened to her - I stopped (because of smth.) - they hurried up - he met - he was - he will arrive - it/he will start
Intrnt	mwdm - blwtwv - HAswbyp - myJAhyrtz - lAbtwb - AlwAb - kwmywnykyXnz - HwAsb - brmjyp - Abswn - nAfyJytr - mdmjA - AllAbtwb - HAswby - mdmj - AlHAswbyp - byksl
Internet	modem - bluetooth - computer - megahertz - laptop - web - communications - computers - programming - Epson - navigator - compact(f) - laptop - my - computer - compact(m) - computer - pixel

Several conclusions can be drawn from the comparative results in Table VIII. First, models with the original NN clustering slightly outperform the Brown and unigram clustering if the latter are performed in the classic *single-step* way. However, the perplexity results in single-step and SOUL columns should not be directly compared as all the NNLMs in the latter scenario benefit from pre-training of neural network parameters in steps 1 and 3′, as it was mentioned above. Results for the *SOUL* scenario lead to the conclusion that taking advantage of the SOUL training approach brings additional improvements for the NNLMs based on the Brown and unigram clustering methods. At the same time, there is no significant difference in perplexity between the three methods in the SOUL scenario. This implies that the way words are assigned to classes is not very important when the complete SOUL NNLM training is performed. Finally, an additional benefit of the original NN clustering is that it is obtained as a by-product during model training, and thus at no extra cost, whereas Brown clustering is computationally expensive.

The SOUL clustering scheme is based on the similarity between words in continuous space. This similarity can be analyzed by finding the nearest neighbors of words according to the Euclidean distance in the projection space. Table IX includes some words with close concepts or sharing similar functions, that are close in the SOUL projection space. Headwords are the words for which "similar" words are determined (labeled *close words*). The closest words are presented first with

more distant words near the end of the lists. The Arabic words are MADA-decomposed units represented with a slightly modified MADA notation used at LIMSI along with their possible translations. The indices (m) and (f) stand for the grammatical markers for gender (masculine/feminine), information encoded in Arabic words. Many cases were observed where words with semantic or grammatical similarities have similar representations in the projection space. Thus, neural networks seem to reveal some of the similarities that exist between words.

VI. Discussion and Conclusions

The Structured OUtput Layer Neural Network approach to language modeling was presented in detail in this paper. This approach combines neural network and class-based language models, with the goal of improving STT system performance for large-scale tasks. The SOUL architecture allows training of neural network LMs with full vocabularies, in contrast to standard NNLMs that require shortlists for computational reasons. The performance of the SOUL NNLMs was evaluated on Mandarin Chinese and Arabic data from recent GALE development and test sets. Significant improvements in speech recognition were obtained over challenging baselines using shortlist NNLMs interpolated with conventional 4-gram LMs.

Longer-context NNLMs were shown to improve the results without drastic increase in computational costs and model size. The ability of feed-forward NNLMs to improve system performance with increasing of context is in line with results obtained with recurrent networks that implicitly take into account the full history to predict a given word [26].

There is a major difference in the recognition vocabulary size for the Arabic and Mandarin languages. The Mandarin vocabulary contains 56 k words and covers essentially all lexical items, where as the Arabic vocabulary contains 300 k MADA-decomposed entries. Perplexity gains with the SOUL over shortlist NNLMs are larger for Arabic than for Mandarin. Comparing the SOUL NNLM and shortlist NNLMs, the reduction of the speech recognition error rate is less than the perplexity reduction. This can be explained by a relatively high data coverage with shortlists for both languages. The training data coverage of the 12 k shortlists for Mandarin and Arabic are 95% and 90% respectively. Such statistics show that similar size shortlists do relatively well in terms of data coverage even for models with very different sized vocabularies. Thus, as confirmed by the experimental results, the improvements from using full-vocabulary SOUL NNLMs, while being consistent, are not proportional to the vocabulary size.

Another reason that similar gains in speech recognition are observed for both languages could be that the amounts of data are insufficient to robustly estimate parameters for very infrequent words. In order to address this issue, the enhanced SOUL NNLM training scheme was proposed. This method carries out separate training of different parts of the structured output layer. An order of magnitude more data are used to train the OOS part of the SOUL NNLM without any prohibitive increase in computational cost and training time. Although it was observed that the enhanced SOUL NNLM is advantageous when used on its own, the enhanced training scheme does not seem to have much influence on the overall performance after interpolation with standard n-gram LMs.

Investigation of SOUL NNLM configurations led to several conclusions about the peculiarities of the SOUL architecture. First, frequent words should be treated separately though the size of the shortlist can be kept small (e.g. 8 k words). Second, the number of top-level classes and the depth of the clustering tree do not have much influence on perplexity. The use of clustering tree itself is important since it provides faster training and better perplexities as compared to flat NNLMs.

Consistent perplexity and speech recognition improvements over both conventional n-gram baselines and shortlist NNLMs on the GALE Mandarin and Arabic STT tasks make the SOUL NNLM an alternative to the shortlist approach to neural network language modeling. The application of the SOUL NNLM is not confined to speech recognition but can be used for other language technology tasks. The SOUL NNLM has been recently reported to bring improvements in the statistical machine translation framework [27].

Acknowledgment

The authors are grateful to the anonymous reviewers for their insightful and valuable comments. The authors also thank their colleagues who provided the baseline LIMSI STT systems, and in particular Abdel Messaoudi and Mohamed Faouzi Benzeghiba, who provided much useful advice.

References

[1] H. Schwenk, "Continuous space language models," *Comput., Speech Lang.*, vol. 21, no. 3, pp. 492–518, 2007.
[2] H.-K. Kuo, L. Mangu, A. Emami, and I. Zitouni, "Morphological and syntactic features for Arabic speech recognition," in *Proc. ICASSP'10*, 2010, pp. 5190–5193.
[3] J. Park, X. Liu, M. Gales, and P. Woodland, "Improved neural network based language modeling and adaptation," in *Proc. Interspeech'10*, 2010, pp. 1041–1044.
[4] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *NIPS*, vol. 13, pp. 933–938, 2001.
[5] H. Schwenk and J.-L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Proc. ICASSP'02*, 2002, pp. 765–768.
[6] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in *Proc. ICASSP'11*, 2011, pp. 5524–5527.
[7] H.-S. Le, I. Oparin, A. Messaoudi, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Large vocabulary SOUL neural network language models," in *Proc. Interspeech'11*, 2011, pp. 1469–1472.
[8] L. Lamel, J.-L. Gauvain, V. Bac Le, I. Oparin, and S. Meng, "Improved models for Mandarin speech-to-text transcription," in *Proc. ICASSP'11*, 2011, pp. 4660–4663.
[9] L. Lamel, A. Messaoudi, and J.-L. Gauvain, "Automatic speech-to-text transcription in Arabic," *ACM TALIP, Spec. Iss. Arabic Natural Lang. Process.*, vol. 8, no. 4, pp. 1–18, 2009.
[10] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký, "Strategies for training large scale neural network language model," in *Proc. ASRU'11*, 2011, pp. 196–201.
[11] H. Le, A. Allauzen, G. Wisniewski, and F. Yvon, "Training continuous space language models: Some practical issues," in *Proc. EMNLP'10*, 2010, pp. 778–788.
[12] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. AISTATS'05*, 2005, pp. 246–252.
[13] A. Mnih and G. Hinton, "A scalable hierarchical distributed language model," *NIPS*, vol. 21, pp. 1081–1088, 2008.

[14] P. Brown, P. de Souza, R. Mercer, V. Della Pietra, and J. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, 1992.
[15] J. Goodman, "Classes for fast maximum entropy training," in *Proc. ICASSP'01*, 2001, pp. 561–564.
[16] A. Emami, "A Neural Syntactic Language Model," Ph.D. dissertation, Johns Hopkins Univ., Baltimore, MD, 2006.
[17] P. Xu and F. Jelinek, "Random forests and the data sparseness problem in language modeling," *Comput. Speech Lang.*, vol. 21, no. 1, pp. 105–152, 2007.
[18] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Černocký, "Recurrent neural network language modeling toolkit," in *Proc. ASRU'11 Demo Session*, 2011.
[19] S. Chen and S. Chu, "Enhanced word classing for Model M," in *Proc. Interspeech'10*, 2010, pp. 1037–1040.
[20] A. Emami and S. Chen, "Multi-class Model M," in *Proc. ICASSP'11*, 2011, pp. 5516–5519.
[21] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *JMLR*, vol. 3, pp. 1137–1155, 2003.
[22] J. Luo, L. Lamel, and J.-L. Gauvain, "Modeling characters versus words for Mandarin speech recognition," in *Proc. ICASSP'09*, 2009, pp. 4325–4328.
[23] L. Lamel, A. Messaoudi, and J.-L. Gauvain, "Investigating morphological decomposition for transcription of Arabic broadcast news and broadcast conversation data," in *Proc. Interspeech'08*, 2008, pp. 1429–1432.
[24] F. Diehl, M. Gales, M. Tomalin, and P. Woodland, "Morphological analysis and decomposition for Arabic speech-to-text systems," in *Proc. Interspeech'09*, 2009, pp. 2675–2678.
[25] I. Oparin, M. Sundermeyer, H. Ney, and J.-L. Gauvain, "Performance analysis of neural networks in combination with n-gram language models," in *Proc. ICASSP'12*, 2012, pp. 5005–5008.
[26] T. Mikolov, M. Karafiat, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech'10*, 2010, pp. 1045–1048.
[27] A. Allauzen, G. Adda, J. Crego, H. Bonneau-Maynard, H.-S. Le, A. Max, G. Wisniewski, F. Yvon, A. Lardilleux, A. Sokolov, and T. Lavergne, "Limsi @ WMT'11," in *Proc. VI Workshop Statist. Mach. Translat. WMT'11*, 2011, pp. 309–315.

Hai-Son Le is currently a Ph.D. student at the University Paris-Sud 11, working in LIMSI/CNRS as a member of Spoken Language Processing Group (TLP). His current research interests are Statistical Language Modeling, Statistical Machine Translation and Automatic Speech Recognition.

Ilya Oparin (M'11) is currently a researcher at the Spoken Language Processing group at LIMSI/CNRS, France. He holds a diploma cum laude (2003) in Theoretical and Applied Linguistics from the St.Petersburg State University in Russia and received a Ph.D. (2009) in Computer Science and Engineering from the University of West Bohemia, Czech Republic, having done research work in collaboration with Speech@FIT group at BUT in Brno. His research interests include speech recognition and statistical machine translation.

Alexandre Allauzen is Associate Professor in Computer Science at the University Paris-Sud and member of the Spoken Language Processing group of the LIMSI/CNRS. He received a Ph.D. (2003) in Computer Science from the University Paris-Sud in collaboration with INA, the French National Institute in charge of audiovisual archives. His main research interests include statistical language modeling, machine translation and automatic speech recognition.

Jean-Luc Gauvain (M'98) is a senior researcher at the CNRS, where he is head of the Spoken Language Processing Group at LIMSI. He received a doctorate in Electronics from the University of Paris-Sud 11 in 1982, and joined the CNRS as a permanent researcher in 1983. His primary research centers on large vocabulary continuous speech recognition and audio indexing. His research interests also include conversational interfaces, speaker identification, language identification, and speech translation. He has participated in many speech related projects both at the French National and European levels and has led the LIMSI participation in DARPA/NIST organized evaluations since 1992, most recently for the transcription of broadcast news data and of conversational speech. He has over 250 publications and received the 1996 IEEE SPS Best Paper Award in Speech Processing and the 2004 ISCA Best Paper Award for a paper in the Speech Communication Journal. He was co-editor-in-chief of the Speech Communication from 2007 to 2009.

François Yvon is Professor in Computer Science at the University Paris Sud and member of the Spoken Language Processing group of the LIMSI/CNRS. He was previously (1996–2007) associate professor in the Department of Computer Science at Telecom ParisTech. Françcois Yvon holds a Ph.D. in Computer Science and engineering degrees from the Telecom ParisTech and an engineering degree from the École Polytechnique. His main research interests include analogy-based and statistical language learning, speech recognition and synthesis and statistical machine translation.

Structured Output Layer Neural Network Language Models for Speech Recognition

Hai-Son Le, Ilya Oparin, *Member, IEEE*, Alexandre Allauzen, Jean-Luc Gauvain, *Member, IEEE*, and François Yvon

***Abstract*—This paper extends a novel neural network language model (NNLM) which relies on word clustering to structure the output vocabulary: Structured OUtput Layer (SOUL) NNLM. This model is able to handle arbitrarily-sized vocabularies, hence dispensing with the need for shortlists that are commonly used in NNLMs. Several softmax layers replace the standard output layer in this model. The output structure depends on the word clustering which is based on the continuous word representation determined by the NNLM. Mandarin and Arabic data are used to evaluate the SOUL NNLM accuracy via speech-to-text experiments. Well tuned speech-to-text systems (with error rates around 10%) serve as the baselines. The SOUL model achieves consistent improvements over a classical shortlist NNLM both in terms of perplexity and recognition accuracy for these two languages that are quite different in terms of their internal structure and recognition vocabulary size. An enhanced training scheme is proposed that allows more data to be used at each training iteration of the neural network.**

***Index Terms*—Automatic speech recognition, neural network language model, speech-to-text.**

I. Introduction

HAVING been used for several decades, n-gram models still form the basis of modern language modeling for speech-to-text (STT) transcription. Despite numerous attempts there are very few approaches that have been shown to systematically and significantly improve over well-estimated n-gram language model (LM) baselines. Neural network language models (NNLMs) are among these approaches and have been adopted in some state-of-the-art STT systems [1]–[3].

Neural network LMs were introduced in [4], [5] as a means to improve discrete models. Standard n-gram back-off LMs rely on a discrete representation of the vocabulary, where each word is associated with a discrete index. In contrast, NNLMs are based on the idea of a continuous word representation, where each word is associated with a real-valued feature vector. In this continuous space, distributionally similar words are neighbors. Thus n-gram distributions are expressed as a smooth function of the word representation, and can take into the account underlying similarities between words. A neural network jointly estimates both the word representations in a continuous space and the associated probabilities.

An important specificity of NNLMs is the capability to take into account longer n-gram contexts. Previous experiments at LIMSI with standard n-gram LMs and large setups indicated that the gain obtained by increasing the n-gram order from 4 to 5 is almost negligible despite a drastic increase in model size. Handling such models is thus rather impractical and can hardly be done without pruning. For NNLMs, increasing the word context length at the input layer results in at most a linear growth in complexity [1] and does not lead to any prohibitive computational or memory overhead.

The major bottleneck with NNLMs is the computation of posterior probabilities in the output layer, which must contain one unit for each word in the vocabulary. As the softmax function is used to obtain the posterior probabilities, the summation over the entire vocabulary is required for each word in the training or test data. This makes the handling of large vocabularies too complex since it would require prohibitive computation time. As a practical workaround, NNLMs usually estimate probabilities only for the fraction of the vocabulary consisting of the most frequent words which is called a *shortlist*. Probabilities of all n-grams finishing with an out-of-shortlist (OOS) word are estimated with a conventional n-gram LM. Such a restriction limits the potential of NNLMs.

This article extends the *Structured OUtput Layer* (SOUL) model introduced in [6], [7], describing training and clustering issues in detail. The SOUL approach combines the benefits of neural networks and class-based LMs by structuring the vocabulary by means of a clustering tree automatically induced from the continuous word representation. In contrast to standard NNLMs, SOUL NNLM makes it feasible to estimate n-gram probabilities for large vocabularies. As a result, all vocabulary words, and not just the words in a shortlist, can benefit from the improved prediction capabilities of the NNLM.

Since estimating word probabilities for all vocabulary entries can be regarded as one of the major distinctive features of the SOUL NNLM, it is natural to evaluate and compare their performance to more traditional NNLMs for different vocabulary sizes.

Manuscript received February 17, 2012; revised May 25, 2012 and July 24, 2012; accepted August 03, 2012. Date of publication August 28, 2012; date of current version nulldate. This work was supported in part under the research program Quaero, funded by OSEO, the French State agency for innovation and in part under the GALE program of the Defense Advanced Research Projects Agency, Contract HR0011-06-C-0022. Any opinions, findings or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gokhan Tur.

H.-S. Le, A. Allauzen, and F. Yvon are with the Université Paris-Sud and LIMSI-CNRS, Orsay, France ***[AUTHOR: Please provide postal codes for each address.]*** (e-mail: lehaison@limsi.fr; allauzen@limsi.fr; yvon@limsi.fr).

I. Oparin and J.-L. Gauvain are with LIMSI-CNRS, Orsay, France (e-mail: oparin@limsi.fr; gauvain@limsi.fr).

Digital Object Identifier 10.1109/TASL.2012.2215599

Speech recognition experiments were carried out to assess the performance of the SOUL NNLMs for the Mandarin Chinese and Arabic (Modern Standard Arabic) languages using data and systems developed for the GALE program. The baseline STT systems [8], [9] have vocabularies of different sizes: 56 k words for Mandarin (including all characters) and 300 k MADA-decomposed entries for Arabic. These systems make use of well-tuned 4-gram LMs trained on corpora containing several billion words of text (without any pruning or cut-offs) interpolated with standard shortlist NNLMs.

It was shown that the training of full vocabulary NNLMs is computationally feasible with the SOUL architecture, and that this leads to improved STT accuracy for large tasks. In this paper the clustering issues and the schemes for training the class model embedded within the SOUL NNLM are described in detail. Additional experiments dealing with the investigation of the influence of different parameters on SOUL NNLM performance are also reported. Examples of classes formed by words that are close according to the SOUL NNLM projection space are presented.

The remainder of this paper is organized as follows. Related work on hierarchical neural networks is summarized in Section II, followed by a description of the architecture of SOUL NNLMs in Section III. Section IV describes the experimental setup and the baseline STT systems, with the experimental results given in Section V. Finally, Section VI concludes the paper along with providing a discussion of the main findings.

II. Related Work

A number of techniques are conventionally used to make the training of NNLMs computationally feasible on the very large corpora used to develop state-of-the-art STT systems.

As the time needed to train a NNLM on all available data is usually prohibitive, a resampling technique was used to partially circumvents this bottleneck [1]. Limited amounts of data are selected at each neural network training iteration by sampling the original corpus. This resampling is usually biased towards in-domain data. However, resampling alone is not sufficient to train NNLMs with large recognition vocabularies as the output layer is also prohibitively large. A proposed solution to this problem was to restrict the output vocabulary to a shortlist of several thousand most frequent words [5].

Recently, some new methods for optimizing NNLM training have been proposed. For example, a significant speed-up in training without performance degradation can be obtained by using a small hidden layer in combination with direct connections between the input and the output layers. Such a model with only 40 neurons in the hidden layer was reported to be much faster to train and to provide the same results as a model with 320 neurons in the hidden layer but without direct connections [10]. Another recent study [11] showed the impact of the initialization of the NNLM parameters and proposed three new training schemes.

A crucial issue that arises with the use of shortlists is that the NNLMs estimate the probabilities only of a limited number of words. Therefore the probability distribution must be normalized with a standard back-off LM in order to cover words that are not in the shortlist. The formula for calculating the normalized probabilities with shortlist NNLMs is:

$$P(w_t|h_t) = \begin{cases} \hat{P}_N(w_t|h_t) \cdot \alpha(h_t) & \text{if } w_t \in \text{shortlist} \\ \hat{P}_B(w_t|h_t) & \text{otherwise} \end{cases} \quad (1)$$

where w_t is the word to be predicted and h_t its $n-1$ word history, $\hat{P}_N$ is the probability of an in-shortlist word calculated with the NNLM, $\hat{P}_B$ is the probability assigned by the standard backoff n-gram LM. The scaling factor $\alpha(h_t)$ depends on the history h_t and was defined in [5] as:

$$\alpha(h_t) = \sum_{w \in shortlist} \hat{P}_B(w|h_t), \quad (2)$$

As both $\hat{P}_N$ and $\hat{P}_B$ are normalized to sum to one, the "combined" conditional distribution P is also normalized.

To handle large output vocabularies, [12] and [13] proposed to use a hierarchical structure for the output layer following previous research to speed-up systems using maximum entropy models. It should be noted that the idea to use classes to factorize the output layer of a neural network can be traced to the classical work of Brown *et al.* on distributed word representation [14].

A conditional maximum entropy model in its general form is expressed as

$$P(w_t|h_t) = \frac{\exp\left(\sum_j \lambda_j f_j(w_t, h_t)\right)}{\sum_{w \in V} \exp\left(\sum_j \lambda_j f_j(w, h_t)\right)} \quad (3)$$

where f_j are the features, λ_j are the feature weights. The normalization term in the denominator requires a summation over all in-vocabulary words $w \in V$. This poses exactly the same computational problems as the softmax in the output layer of neural networks. In [15] it was proposed to first cluster words into classes and then compute the conditional probabilities:

$$P(w_t|h_t) = P\left(c(w_t)|h_t\right) \cdot P\left(w_t|h_t, c(w_t)\right), \quad (4)$$

where $c(w)$ denotes the class assigned to a word w. Two models are trained separately: one that predicts the class probability given the history; and one predicting a word given its class and its history. With this kind of model, a significant speed-up can be obtained since the required summations for both models are drastically reduced: for the first model, the summation ranges over the number of different classes, whereas for the second model it involves only the words in the given class.

The idea of clustering the vocabulary words was introduced for NNLMs in [12], where a binary hierarchical clustering was derived from the WordNet semantic hierarchy. In that work, the output vocabulary was first clustered and represented by a binary tree. Each internal node of the tree held a word cluster which was further divided into two sub-clusters and so on. The leaves correspond to the words at the end of this representation of the vocabulary. The neural network aims to estimate probabilities for the paths in this binary tree given the history, rather than for the word itself, as reflected by the following equation:

$$P(w_t|h_t) = \prod_{j=1}^{m} P\left(b_j(w_t)|b_1(w_t)\dots b_{j-1}(w_t), h_t\right) \quad (5)$$

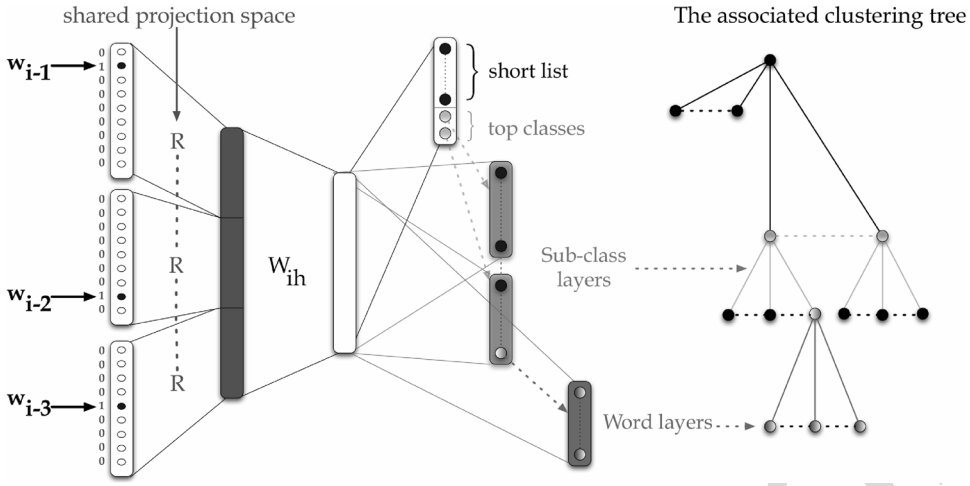

Fig. 1. The architecture of the structured output layer neural network language model.

where m is the depth of the tree and b stands for 1 or 0 in the path for a given word. This approach resulted in a two order of magnitude speed-up on a small data set (1 million words of the Brown corpus) with a vocabulary of 10 k words. However, a significant increase in perplexity was observed relative to a standard NNLM. One of the possible explanations is the widely observed difficulty of integrating linguistic information in statistical LMs and in the NNLM described above the clustering tree is constructed using the semantic information from the WordNet. Alternative solutions that do not rely on external linguistic sources were proposed in [13], [16] for neural network and log-bilinear models. These methods are also appealing, as the WordNet or similar resources covering large output vocabularies are not necessarily available for different languages. Using automatic clustering to factorize the output layer resulted in significant speed-ups in training and perplexities similar to those obtained without clustering.

It can be seen that much of the previous work concentrated on reducing the computational complexity, with less emphasis on improving the accuracy of speech recognition or machine translation systems. In the binary hierarchical log-bilinear approach the resulting binary clustering tree is deep (i.e. has many levels), and thus faces a data fragmentation problem. This problem is well known for decision trees, especially for language modeling tasks [17]. Moreover, if a word is assigned to a wrong class at some level in a decision tree, this error affects all the internal nodes (or clusters) leading to this word. This is typically the case for rare words that represent a large fraction of the vocabulary. Thus an error in one word may have a significant impact on the whole system. By relaxing the constraint of the binary structure, it is expected that this shortcoming will be overcome as explained in Section III.

Factorization of the output layer was also used for recurrent NNLMs. Although a simpler approach based on the distribution of unigram probabilities that does not reveal any relationships between the words was used, this work illustrates the growing interest in improving neural network LMs for STT [18]. Another active direction of research dealing with different clustering methods is connected with recently proposed Model M and its enhancements [19], [20].

III. Structured Output Layer Neural Network Language Model

In this section the class-based neural network language model, namely Structured OUtput Layer (SOUL) NNLM, is described in detail.

In the SOUL NNLM [6], [7] the output vocabulary is structured by a clustering tree, where each word belongs to only one class and ends up in a single leaf of the tree. If w_i denotes the i^{th} word in a sentence, the sequence $c_{1:D}(w_i) = c_1, \ldots, c_D$ encodes the path for word w_i in the clustering tree, and D is the depth of the tree. $c_d(w_i)$ is a class or sub-class assigned to w_i, and $c_D(w_i)$ is the leaf associated with w_i (the word itself). Then the n-gram probability of w_i given its history h can be estimated as follows:

$$P(w_i|h) = P\left(c_1(w_i)|h\right) \prod_{d=2}^{D} P\left(c_d(w_i)|h, c_{1:d-1}\right) \qquad (6)$$

The probability of a word is conditioned not only on its context, but also on a non-binary string of indices encoding a path in the clustering tree. Each word belongs to only one class and there is a softmax function at each level of the hierarchical representation.

Fig. 1 represents the architecture of the SOUL NNLM. Both SOUL and standard NNLMs share the same architecture up to the hidden layer. The output structure of the SOUL model is made of three kinds of output layers. The first output layer estimates the distribution over the shortlist words and the top (most general) classes for the OOS words. Words in the shortlist are regarded as a special case since each represents its own class without sub-clustering (the tree depth D is equal to 1 in this case). The top classes serve as the roots of trees to handle the OOS words and include multiple sub-class layers each with a softmax function, forming the second kind layers. Finally, the word layers estimate the word probabilities for the OOS words. To summarize, the SOUL model estimate the n-gram probability of a word w_i given its context h as defined by the (6) as follows: the upper layer handles the distribution over the top classes $P(c_1(w_i)|h)$; then subsequent sub-class layers compute the sub-class probabilities $P(c_d(w_i)|h, c_{1:d-1})$ for $1 < d < D$;

finally the OOS word probabilities $P(c_D(w_i)|h, c_{1:D-1})$ are given by the word layers.

The SOUL model can handle any arbitrary tree structure of the vocabulary. A slightly different architecture could be considered, where the first softmax layer consists of classes corresponding to the shortlist words plus an additional node. This additional node serves as a root of a tree that includes all sub-class layers needed to estimate the probabilities of the OOS words. Experiments with this architecture showed slightly worse results with a negligible complexity reduction.

A. Training Scheme

The training scheme addresses two closely related parts of the SOUL model: the parameters estimation and the word clustering that structures the output vocabulary. In [13] a rather sophisticated divisive clustering algorithm is presented which is based on a mixture of two Gaussians applied to the continuous word space. In this work, we propose a more straightforward method based on the relationship between the two word spaces defined in a standard NNLM [11], i.e. the context and the prediction spaces. The training procedure is summarized as follows:

Step 1) **pre-training**: Train an NNLM with a shortlist as output, using the one vector initialization method [11]. This step is only required to provide a preliminary estimate of the projection space defined by the matrix R.

Step 2) **dimension reduction of the projection space**: Apply the standard Principal Component Analysis (PCA) on the matrix R.

Step 3) **word clustering**: Perform a divisive top-down word clustering based on the K-means algorithm, using the continuous representation induced by the previous step for words that are not in the shortlist.

Step 4) **full training**: Train a full-vocabulary NNLM with the tree structure as output.

The goal of step 1 is to learn the word features for clustering. In the one word initialization method introduced in [11], all the words in the context vocabulary are initially projected onto the same (random) point in the context space. The intuition is that it will be easier to build meaningful neighborhoods, especially for rare types, if all words are initially considered similar and only diverge if there is sufficient evidence in the training data to suggest that they should. With this kind of initialization, only a few iterations are required as the learning of word features converges quickly.

As words input to a NNLM are represented in 1-of-K coding with 1 corresponding to a word's vocabulary index and all other elements set to zero, each line in the projection matrix R corresponds to a continuous representation of a particular word. The dimensionality of this representation is reduced with the PCA at step 2. The size of the context space after the dimensionality reduction is equal to 10 for the experiments described here. Such low dimensional context space makes the subsequent K-means clustering at the step 3 computationally cheap.

The clustering at step 3 divides a word class (or sub-class) only if the number of words in this class is above an empirically determined threshold W. Each class containing more than W words is divided in $|\sqrt{W}+1|$ sub-classes. The value of this threshold allows to tune the depth of the clustering tree: small values result in a deep clustering structure while a very large value generates a flat tree. This threshold depends on the vocabulary size and can be optimized. However, experimental results with different vocabulary sizes show that its value does not have much influence on the overall performance of the STT systems and three levels of hierarchy seem sufficient. The clustering deals with rare words and thus there is little point in making small classes and having a deep tree structure. In these experiments, the K-means algorithm starts with 4 k top classes. Then each sub class is recursively divided. However, in practice, the one vector initialization used at step 1 implies that very rare words are represented by very similar or identical feature vectors. In that particular case, the rare words are then naturally grouped in the same class by the first K-means step, and a recursive subdivision is therefore not well suited. In the rest of this paper, the class for rare words is randomly divided.

Finally, standard back-propagation training [21] is performed for the full vocabulary SOUL NNLM at step 4 using the parameters obtained at step 1 for initialization.

In order to evaluate the influence of the type of clustering in the SOUL architecture, the clustering proposed above was compared with two other techniques. The first is the widely known Brown clustering algorithm based on the maximization of the mutual information between adjacent classes [14]. This algorithm has served as a baseline in the community and it has been hard to improve upon, even with much more sophisticated methods. The second method has been implemented in the BUT RNNLM recurrent network toolkit [18], and assigns words to classes according to their unigram probabilities.

Comparative results for the three clustering methods, different shortlist sizes, number of classes and depth of the clustering tree are given and discussed in Section V.

B. Enhanced Training Scheme

NNLM training maximizes the log-likelihood of the training data. This optimization is performed by stochastic back-propagation [21]. A previously trained standard NNLM is used to initialize the shared parameters (i.e matrices R and W_{ih} in Fig. 1). The other parameters are initialized in a usual way, as described in the previous section. Overall, the training time for a SOUL model is only about 30% longer than for standard 8 k shortlist NNLMs and slightly shorter the time to train a 12 k shortlist NNLM.

When dealing with large vocabularies (e.g. the Arabic one in this study), the number of parameters related to the class-part of the model increases significantly. As a result, the resampling technique that is used for neural network training at each epoch may be insufficient to obtain robust parameter estimates.

As described in Section III, the output layer of the SOUL NNLM is comprised of two parts: the first layer which directly models the probabilities of the most frequent in-shortlist words and top classes, and the remaining sub-class layers that deal with the classes of OOS words as illustrated in Fig. 1.

The parameters related to the first output layer are updated for all training examples since it covers the shortlist words and the most general classes for the less frequent words. The n-grams

ending with in-shortlist words are used to update the parameters only of this first layer, leaving sub-class layers intact. The parameters of sub-class layers are updated with the n-grams ending in an OOS word and only those class layers leading to the leaf with this particular word are activated and the corresponding parameters are updated. A shortlist usually covers a large part of training examples so that the updates of the parameters related to sub-class layers are less frequent. Moreover, when such an update occurs, it is done for a subset of the parameters corresponding to a particular path in the clustering tree of the class layers. At the same time the number of parameters corresponding to OOS words is much larger. As a result, the two parts of the SOUL output layer are not trained equally well. Therefore a modified SOUL training scheme based on the separate training of the OOS part at the output layer is proposed. This scheme adds an additional step to the training procedure described in Section III-A.

This additional step 3′ is similar to 1, but is carried out only for OOS words. At this step the n-grams ending with shortlist words are skipped and the parameters associated with shortlist words are temporarily fixed, reducing the size of the first output layer to the number of main classes of infrequent words only (4 k as opposed to 12 k in the SOUL setup). As explained above, the softmax in the first output layer is triggered for each training example, which, in turn, requires summation over all nodes in this layer. Thus, any reduction in the size of the first output layer results in a significant speed-up. Since the number of OOS occurrences represent a relatively small proportion of the training data (for our experiments on Chinese and Arabic data these represent 5–10% of all word occurrences), the number of training examples can be easily increased by the factor of 10. As NNLMs are trained (with resampling) on less data as compared with the baseline n-gram LMs, there are always additional data available for the step 3′. The parameters obtained at step 3′ are used to initialize the parameters associated with the OOS words at the step 4.

IV. Experimental Setup

The performance of the SOUL NNLM is compared to standard shortlist NNLMs on the GALE Arabic and Mandarin Chinese tasks.

The general parameters of NNLMs are presented in Table I. Several NNLMs differing in the size of the projection layer and the hidden layer are trained for each task, and then interpolated together, along with the standard N-gram backoff LM. Interpolation weights are tuned to minimize perplexity on the development data. The same setups are used for both the SOUL NNLMs and the shortlist NNLMs. Thus the difference in results can be attributed to the use of the whole vocabulary at the output (which, in turn, uses the class information).

Each NNLM is trained on about 25–30 M words at each iteration after resampling of the training data. The enhanced SOUL NNLM training scheme (see Section III-A) is investigated on Arabic, for which shortlists of similar sizes provide lower data coverage than for Mandarin thus leaving more space to improvement with the proposed scheme. Up to 300 M words are used during step 3′ to train the class output layers that deal with OOS words. As all the n-grams ending with an in-shortlist word are skipped at step 3′, it makes about 30 M n-grams that are used to update the parameters.

TABLE I
Parameters of the Mandarin Chinese and Arabic NNLMs. For Each Language, NNLMs With Different Parameters are Subsequently Interpolated Together

	Mandarin				Arabic		
	NN1	NN2	NN3	NN4	NN1	NN2	NN3
initial learning rate	5×10^{-3}				5×10^{-3}		
learning rate decay	5×10^{-8}				5×10^{-8}		
weight decay	3×10^{-5}				3×10^{-5}		
projection layer	500	220	250	300	200	300	400
hidden layer	200	430	450	500	500	400	300

To assess the impact of the increase in the shortlist size, NNLMs with 8 k and 12 k shortlists are trained on the Mandarin data. As slightly better results are obtained with a larger one, the Arabic shortlist NNLMs use a shortlist of 12 k MADA-decomposed units.

In order to study possible improvements from using longer span NNLMs, the increase in context length from 3 (that corresponds to 4-grams) to 5 is investigated. For the shortlist NNLMs the same 4-gram back-off LM is used for OOS words.

The performance of the different Mandarin models was assessed on the GALE dev09_M and eval09_M data sets containing broadcast news and broadcast conversations. A subset of dev09_M called dev09s_M was also defined, comprised of about a third of dev09_M data. The dev09_M, dev09s_M and eval09_M sets contain respectively 97459, 31529 and 79246 segmented words. Three sets are used to evaluate the performance of the different Arabic models, namely dev09s_A, eval10ns_A and dev10c_A. These sets contain of 23576, 45629 and 52181 MADA-decomposed units respectively.

A. Mandarin STT System

Mandarin Chinese is a language with a low morpheme-per-word ratio. Most words in a running text are composed of a single morpheme (character), and, as there are no markers neither for inflection nor for parts of speech, it has a fixed word order. Words in written Chinese are not separated by white spaces. A natural solution is either to use character-based LMs or to perform word segmentation as a pre-processing step. The former was shown to be inferior to the latter [22], so the longest-match segmentation approach is taken in this work. As is the convention, the character error rate is used to evaluate recognition performance for Chinese.

The recognition vocabulary contains 56 k entries including both multicharacter words (about 50 k) and individual Chinese characters (6 k entries). There are no out-of-vocabulary (OOV) words for Chinese. The acoustic models and the decoding process of the Mandarin STT system are described in detail in [8]. The Mandarin LM is trained on 3.2 billion word tokens after segmentation. Individual 4-gram LMs are first built for each of 48 sub-corpora without any cut-offs and pruning. These models are smoothed according to the unmodified interpolated Kneser-Ney scheme and are subsequently linearly interpolated to form the baseline 4-gram LM, with the weights tuned on the dev09_M data. This resulting model includes 2.2 billion unique n-grams.

TABLE II
PERPLEXITY AND CER (%) FOR DIFFERENT MANDARIN LMS

model	ppl dev09_M s/a	ppl dev09_M int	CER dev09s_M	CER eval09_M
Kneser-Ney 4g	211		9.8%	8.9%
+ shortlist 8k NN 4g	229	187	9.5%	8.6%
+ shortlist 12k NN 4g	227	185	9.4%	8.6%
+ SOUL NN 4g	221	180	9.3%	8.5%
+ shortlist 8k NN 6g	214	177	9.4%	8.5%
+ shortlist 12k NN 6g	207	172	9.3%	8.5%
+ SOUL NN 6g	**192**	**162**	**9.1%**	**8.3%**

TABLE III
PERPLEXITY FOR DIFFERENT ARABIC LMS

LM type	dev09s_A s/a	dev09s_A int	eval10ns_A s/a	eval10ns_A int	dev10c_A s/a	dev10c_A int
Kneser-Ney 4g	312		239		256	
shortlist 12k NN 4g	324	276	247	213	256	224
SOUL NN 4g	293	256	225	200	231	208
SOUL+ NN 4g	277	250	214	195	221	204
shortlist 12k NN 6g	302	263	228	202	236	210
SOUL NN 6g	255	231	196	180	200	186
SOUL+ NN 6g	**245**	**227**	**189**	**177**	**194**	**183**

B. Arabic STT System

Arabic is a highly inflective and morphologically rich language characterized by a large number of word forms for a given lemma. This usually results in vocabularies that are several times larger than the ones used for Chinese or English.

The baseline Arabic STT system used in the speech recognition experiments reported here gives state-of-the-art performance on the GALE tasks [9]. The Arabic vocabulary contains 300 k entries. The MADA[1] (Morphological Analysis and Disambiguation for Arabic) tool is used to decompose the words into their morphological constituents, increasing lexical coverage and improving recognition performance [23], [24]. The Arabic LM training data contains about 1.7 billion words before decomposition, resulting in a total of about 2 billion morphs. 4-gram LMs are trained for 32 different MADA-decomposed text subsets using the unmodified interpolated Kneser-Ney scheme, without pruning or cut-offs. These LMs are further interpolated to form the final 4-gram LM. The resulting model includes 1.2 billion unique n-grams.

V. EXPERIMENTAL RESULTS

In this section, experimental results are provided for Arabic and Mandarin, both in terms of perplexity and recognition error rates. The internal structure of these two languages is quite different which is reflected by the vocabulary sizes of their STT systems. The SOUL word clustering approach is also compared to the classical Brown clusters and the more straightforward unigram algorithm. The representation of words in the projection space is explored by finding the neighbors for some selected words.

A. Perplexity Results

Tables II and III summarize the results in terms of perplexity for Mandarin and Arabic respectively. The results are provided both for stand-alone NNLMs (columns *s/a*) and after interpolation with the baseline 4-gram LMs (columns *int*).

The rows marked with *shortlist* correspond to shortlist NNLMs (with 8 k or 12 k shortlists). The context size for the NNLMs is indicated as 4 g or 6 g. The models marked as *SOUL* are based on the general SOUL architecture, while $SOUL+$ corresponds to the SOUL NNLM that uses the enhanced training scheme for the parameters dealing with OOS words, as described in Section III-B.

As can be seen from Table II, increasing the shortlist size by 50% from 8 k to 12 k words brings only a small improvements in perplexity. The SOUL model that predicts probabilities for all words in the vocabulary and uses word clustering for infrequent words systematically outperforms the 12 k shortlist NNLM for both languages even though it is not more computationally demanding.

Using a longer context (6 g vs. 4 g) reduces the perplexity both for the shortlist and the SOUL NNLMs. These models directly provide lower perplexities than the Kneser-Ney LM, trained on much more data. All interpolations of different orders and types of NNLMs with the Kneser-Ney LMs were found to significantly reduce the perplexity.

These results show that the SOUL NNLM consistently outperforms the shortlist counterparts of the same order on all the test sets. For the 4-gram stand-alone NNLMs, the relative improvement obtained with the SOUL NNLM over the shortlist NNLM is 3% for Mandarin and 9–10% for Arabic (13–15% for SOUL+). In the longer-context 6-gram case, the gains with the SOUL NNLM are somewhat larger, 7% for Mandarin and 14–16% for Arabic (17–19% for the SOUL+). The same tendency holds for the NNLMs interpolated with the Kneser-Ney LMs. For the 4-gram interpolated models the improvement with the SOUL NNLM is 3% for Mandarin and 6–7% for Arabic (8–9% for the SOUL+). For 6-gram interpolated models the relative gains are 8% for Mandarin and 11–12% for Arabic (12–14% for the SOUL+).

The difference in perplexity reduction between the enhanced SOUL NNLM training and the standard SOUL NNLM is smaller after interpolation with the Kneser-Ney LMs. This suggests that the advantage of using 10 times more data to train the part of SOUL NNLMs that deals with rare words should not bring much benefit in state-of-the-art STT systems where NNLMs are interpolated with n-gram LMs.

B. Speech Recognition Results

Tables II and IV summarize the results of speech recognition experiments in terms of character error rate (CER) for Mandarin and word error rate (WER) for Arabic. The lattices generated with the baseline 4-gram Kneser-Ney LMs are rescored with the different models types. The lattice rescoring is performed in a usual way, by extracting n-grams from the lattice and estimating their probabilities with NNLMs. The recognition results with NNLMs are reported after interpolation with the baseline

[1]http://www1.ccls.columbia.edu/~cadim/MADA.html.

TABLE IV
WER (%) WITH DIFFERENT ARABIC LMS

LM type	dev09s_A	eval10ns_A	dev10c_A
Kneser-Ney 4g	14.8	9.6	14.5
+ shortlist 12k NN 4g	14.4	9.1	14.2
+ SOUL NN 4g	14.3	9.0	14.0
+ SOUL+ NN 4g	14.1	9.1	14.0
+ shortlist 12k NN 6g	14.3	9.1	14.2
+ SOUL NN 6g	**14.0**	**8.9**	14.0
+ SOUL+ NN 6g	**14.0**	**8.9**	**13.9**

TABLE V
NNLM WEIGHTS FOR INTERPOLATION WITH THE BASELINE N-GRAM LMS

NNLM model type	NNLM interpolation weight Mandarin	Arabic
shortlist 12k NN 4g	0.53	0.50
SOUL NN 4g	0.60	0.68
SOUL+ NN 4g	-	0.72
shortlist 12k NN 6g	0.60	0.55
SOUL NN 6g	0.69	0.74
SOUL+ NN 6g	-	0.75

4-gram Kneser-Ney LMs, the same as used to generate 4-gram lattices.

To give the idea of lower bounds of possible WER reductions, e.g. on the Mandarin dev09_M set, the oracle CER is about 4%. It should be noted that the exact calculation of oracle WERs is not straightforward on GALE setups due to their specificities (reference transcription is divided into "snipets", normalization, segmentation of Mandarin data into words, MADA decomposition for Arabic). Thus, oracle calculation is approximative and may exhibit differences as compared to the way one-best hypothesis scores are calculated by the NIST *sclite* tool.

The interpolation weights for different NNLMs with the baseline Kneser-Ney LMs are presented in Table V. It can be seen in this table that higher interpolation weights are obtained for the SOUL NNLMs than the shortlist NNLMs.

The results in Table II for Mandarin and Table IV for Arabic show that the improvements in perplexity attained with the SOUL NNLMs compared with the shortlist NNLMs carry over to speech recognition. The best recognition performance is obtained with the 6-gram context NNLMs.

As compared to the Kneser-Ney LMs, the SOUL NNLM, when interpolated with the latter, improves the WER by up to 0.7% absolute for Mandarin and 0.8% for Arabic. The SOUL model brings an absolute error reduction of about 0.2–0.3% more than the shortlist NNLM.

It should be noted that the gains from using 6-gram NNLMs on Arabic are smaller than might be expected since for computational reasons the lattices had to be pruned before rescoring with the 6-gram model. The effect of pruning is most notable on the dev10c_A set which contained some large lattices that were subject to severe pruning. However, since the 6-gram shortlist NNLMs showed no improvement with pruned lattices over 4-gram NNLMs, the 6-gram SOUL NNLMs still improve the results.

C. NNLM Configurations

In order to investigate the impact of different NNLM parameters, such as the size of the shortlist, the number of top classes in the first output layer and the depth of the clustering tree, a number of additional experiments were carried out on the Chinese setup. In these experiments, only one NNLM (as opposed to three or four in the experiments described in the previous sections, see Table I) with 300 nodes in the projection layer for each history word and 500 nodes in the hidden layer is trained for each configuration.

TABLE VI
PERPLEXITY FOR LMS WITH DIFFERENT SHORTLIST SIZES ON THE MANDARIN DEV09_M SET

model shortlist	top classes	depth	ppl s/a	int	training time (days)
Kneser-Ney 4g					
-	-	-	211		-
shortlist NNLMs 6g					
8k	-	-	222	178	2.4
12k	-	-	222	175	3.5
25k	-	-	223	171	7.1
SOUL NNLMs 6g					
8k	4k	3	220	169	3.1
12k	4k	3	219	168	5.6
25k	4k	3	219	168	7.0
flat full-vocabulary NNLMs 6g					
all (56k)	0	1	226	171	14.7

Results with NNLMs with different sizes of the shortlist are presented in Table VI. *Shortlist* column corresponds to different sizes of the shortlist part, *top classes* reports the number of top classes of the first SOUL output layer, *depth* is the depth of the SOUL clustering tree, *s/a* stands for stand-alone NNLMs and *int* for NNLMs interpolated with the baseline n-gram model.

One conclusion from Table VI is that the flat full-vocabulary NNLM, while being computationally very expensive, does similarly or worse than the ones that make use of a shortlist. The SOUL NNLMs benefit from clustering of rare words by means of a clustering tree (shortlist NNLMs back off to normalized Kneser-Ney estimates in this case) at the output, as seen from the comparison with the full-vocabulary flat model. The SOUL NNLMs also deliver top results with a relatively small shortlist (i.e. 8 k). This may be important in order to save computation time and resources, as it is not necessary to train SOUL NNLMs with large shortlists. For example, running similar experiments with shortlists equal or close to the vocabulary size is hardly feasible on larger vocabulary setups (56 k Mandarin vocabulary size can be considered as very moderate), as e.g. Arabic with 300 k vocabulary entries, because of the prohibitive training costs.

A natural question is then whether stand-alone NNLMs outperform n-gram models. This question is difficult to answer on the GALE setups, since it is infeasible to train NNLMs on the same amounts of data as the Kneser-Ney LM baseline; as pointed out in Sections II and IV, resampling has to be used for NNLMs. A fair comparison is however possible on smaller setups. The reader may find a discussion for which type of events NNLMs do better that n-gram LMs and vice versa in [25].

Table VII reports perplexity for Mandarin 6-gram SOUL NNLMs with different numbers of top classes in the first output layer, SOUL NNLMs with clustering trees of different depths and a full-vocabulary SOUL NNLM (no shortlist is

TABLE VII
PERPLEXITY FOR 6-GRAM SOUL NNLMS WITH DIFFERENT NUMBER OF TOP CLASSES AND CLUSTERING TREE DEPTHS ON THE MANDARIN DEV09_M SET

model			ppl		training time
shortlist	top classes	depth	s/a	int	(days)
SOUL NNLMs with different number of top classes					
8k	128	3	221	169	2.4
8k	256	3	218	168	3.5
8k	1k	3	220	169	3.6
8k	2k	3	220	169	2.7
SOUL NNLMs with different clustering tree depth					
8k	4k	3	220	169	3.1
8k	4k	2	220	169	3.0
all	0	1	226	171	14.7
SOUL NNLM without shortlist part					
0	4k	2	242	176	2.2
0	4k	3	240	176	2.4

used, all words are clustered). It can be seen that the number of top-level classes does not have much influence on the final perplexity. The same holds for the depth of the clustering tree. On one hand, training with a flat tree is slow and it yields higher perplexity. On the other hand, there is little difference between the clustering tree of depth two and three. This can be explained by the fact that clustering mostly concern rare words; there is thus little point to perform a deep and fine-grained clustering. However, deeper clustering trees are expected to provide training speed-ups for larger vocabulary tasks as it results in more numerous but smaller softmax layers (see discussion in Section II). A similar experiment on Arabic showed that a three-level tree indeed provides gains in training time as opposed to the two-level one (5.8 days vs. 6.2 days).

The benefit of using the shortlist part in the SOUL architecture as described in Section III-B was also verified. The SOUL NNLM representing the whole vocabulary as a clustering tree was trained. It is reported in the last row of Table VII, showing that in the SOUL architecture, frequent words should be treated separately.

D. Clustering Within SOUL NNLMs

According to the SOUL architecture, the 8 k most frequent words do not undergo clustering as they form classes on their own. Only the remaining words are clustered into 4 k classes on the top level. As described in Section III-A, the training steps (1 to 3) of the SOUL model are used to derive the word clustering. The depth of clustering hierarchy equals to 3.

With the Brown (as in [14]) and the unigram clustering (see Section III-A), models can be directly trained with step 4 within the SOUL architecture. This scenario is referred as *single-step* in Table VIII, where the results obtained with different word clustering schemes are given. The original SOUL clustering procedure with all the training steps described in Section III-A is referred as *SOUL*. In this *SOUL* scenario neural networks based on the *Brown* and the *Unigram* clustering also benefit from the information obtained during steps 1 (e.g. lookup tables) and 3′ (using more data to estimate probabilities of OOS words). The original clustering method based on word similarity in continuous space in the neural network is referred as NN. It should be noted that both Brown and unigram approaches provide clustering tree structures, just as the original NN method.

TABLE VIII
STAND-ALONE SOUL NNLM PERPLEXITY WITH DIFFERENT CLUSTERINGS ON THE MANDARIN DEV09_M SET

Clustering type	4-gram		6-gram	
	single-step	SOUL	single-step	SOUL
Unigram	259	248	229	222
Brown	253	245	225	218
NN	-	245	-	220

TABLE IX
EXAMPLES OF CLOSE MADA-DECOMPOSED ARABIC WORDS ACCORDING TO THE SOUL NNLM PROJECTION SPACE

headword	close words
Aldwlp	AlHwmp - bnwknA - mWsstnA - jAmctkm - mSArfhA - bldytnA - nqAbth - vwrtnA - cAlqDyp - cAlclAqAt - mSArfnA - dwltkm - Aldwylp - wzArthA - mHAkmnA
country	district - our banks - our institutions - your universities - its banks(f) - our town - syndicate - our war - about affair - about relations - our banks - your country - small country - its ministry(f) - our tribunals
jAC	yjyC - EtY - yEty - wAtY - mAjAC - mAwrd - yEtyAn - jACA - tEtY - jAChA - jACtA - Astwqfny - AstcjlwA - ycksh - wkAn - yje - ytbdY
arrived	will arrive - come - he will come - and come - he didn't come - destination - they will come - they arrived - she will come - smb/smth(m) came/happened to her - smb/smth(f) came/happened to her - I stopped (because of smth.) - they hurried up - he met - he was - he will arrive - it/he will start
Intrnt	mwdm - blwtwv - HAswbyp - myJAhyrtz - lAbtwb - AlwAb - kwmywnykyXnz - HwAsb - brmjyp - Abswn - nAfyJytr - mdmjA - AllAbtwb - HAswby - mdmj - AlHAswbyp - byksl
Internet	modem - bluetooth - computer - megahertz - laptop - web - communications - computers - programming - Epson - navigator - compact(f) - laptop - my - computer - compact(m) - computer - pixel

Several conclusions can be drawn from the comparative results in Table VIII. First, models with the original NN clustering slightly outperform the Brown and unigram clustering if the latter are performed in the classic *single-step* way. However, the perplexity results in single-step and SOUL columns should not be directly compared as all the NNLMs in the latter scenario benefit from pre-training of neural network parameters in steps 1 and 3′, as it was mentioned above. Results for the *SOUL* scenario lead to the conclusion that taking advantage of the SOUL training approach brings additional improvements for the NNLMs based on the Brown and unigram clustering methods. At the same time, there is no significant difference in perplexity between the three methods in the SOUL scenario. This implies that the way words are assigned to classes is not very important when the complete SOUL NNLM training is performed. Finally, an additional benefit of the original NN clustering is that it is obtained as a by-product during model training, and thus at no extra cost, whereas Brown clustering is computationally expensive.

The SOUL clustering scheme is based on the similarity between words in continuous space. This similarity can be analyzed by finding the nearest neighbors of words according to the Euclidean distance in the projection space. Table IX includes some words with close concepts or sharing similar functions, that are close in the SOUL projection space. Headwords are the words for which "similar" words are determined (labeled *close words*). The closest words are presented first with

more distant words near the end of the lists. The Arabic words are MADA-decomposed units represented with a slightly modified MADA notation used at LIMSI along with their possible translations. The indices (m) and (f) stand for the grammatical markers for gender (masculine/feminine), information encoded in Arabic words. Many cases were observed where words with semantic or grammatical similarities have similar representations in the projection space. Thus, neural networks seem to reveal some of the similarities that exist between words.

VI. Discussion and Conclusions

The Structured OUtput Layer Neural Network approach to language modeling was presented in detail in this paper. This approach combines neural network and class-based language models, with the goal of improving STT system performance for large-scale tasks. The SOUL architecture allows training of neural network LMs with full vocabularies, in contrast to standard NNLMs that require shortlists for computational reasons. The performance of the SOUL NNLMs was evaluated on Mandarin Chinese and Arabic data from recent GALE development and test sets. Significant improvements in speech recognition were obtained over challenging baselines using shortlist NNLMs interpolated with conventional 4-gram LMs.

Longer-context NNLMs were shown to improve the results without drastic increase in computational costs and model size. The ability of feed-forward NNLMs to improve system performance with increasing of context is in line with results obtained with recurrent networks that implicitly take into account the full history to predict a given word [26].

There is a major difference in the recognition vocabulary size for the Arabic and Mandarin languages. The Mandarin vocabulary contains 56 k words and covers essentially all lexical items, where as the Arabic vocabulary contains 300 k MADA-decomposed entries. Perplexity gains with the SOUL over shortlist NNLMs are larger for Arabic than for Mandarin. Comparing the SOUL NNLM and shortlist NNLMs, the reduction of the speech recognition error rate is less than the perplexity reduction. This can be explained by a relatively high data coverage with shortlists for both languages. The training data coverage of the 12 k shortlists for Mandarin and Arabic are 95% and 90% respectively. Such statistics show that similar size shortlists do relatively well in terms of data coverage even for models with very different sized vocabularies. Thus, as confirmed by the experimental results, the improvements from using full-vocabulary SOUL NNLMs, while being consistent, are not proportional to the vocabulary size.

Another reason that similar gains in speech recognition are observed for both languages could be that the amounts of data are insufficient to robustly estimate parameters for very infrequent words. In order to address this issue, the enhanced SOUL NNLM training scheme was proposed. This method carries out separate training of different parts of the structured output layer. An order of magnitude more data are used to train the OOS part of the SOUL NNLM without any prohibitive increase in computational cost and training time. Although it was observed that the enhanced SOUL NNLM is advantageous when used on its own, the enhanced training scheme does not seem to have much influence on the overall performance after interpolation with standard n-gram LMs.

Investigation of SOUL NNLM configurations led to several conclusions about the peculiarities of the SOUL architecture. First, frequent words should be treated separately though the size of the shortlist can be kept small (e.g. 8 k words). Second, the number of top-level classes and the depth of the clustering tree do not have much influence on perplexity. The use of clustering tree itself is important since it provides faster training and better perplexities as compared to flat NNLMs.

Consistent perplexity and speech recognition improvements over both conventional n-gram baselines and shortlist NNLMs on the GALE Mandarin and Arabic STT tasks make the SOUL NNLM an alternative to the shortlist approach to neural network language modeling. The application of the SOUL NNLM is not confined to speech recognition but can be used for other language technology tasks. The SOUL NNLM has been recently reported to bring improvements in the statistical machine translation framework [27].

Acknowledgment

The authors are grateful to the anonymous reviewers for their insightful and valuable comments. The authors also thank their colleagues who provided the baseline LIMSI STT systems, and in particular Abdel Messaoudi and Mohamed Faouzi Benzeghiba, who provided much useful advice.

References

[1] H. Schwenk, "Continuous space language models," *Comput., Speech Lang.*, vol. 21, no. 3, pp. 492–518, 2007.
[2] H.-K. Kuo, L. Mangu, A. Emami, and I. Zitouni, "Morphological and syntactic features for Arabic speech recognition," in *Proc. ICASSP'10*, 2010, pp. 5190–5193.
[3] J. Park, X. Liu, M. Gales, and P. Woodland, "Improved neural network based language modeling and adaptation," in *Proc. Interspeech'10*, 2010, pp. 1041–1044.
[4] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *NIPS*, vol. 13, pp. 933–938, 2001.
[5] H. Schwenk and J.-L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Proc. ICASSP'02*, 2002, pp. 765–768.
[6] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in *Proc. ICASSP'11*, 2011, pp. 5524–5527.
[7] H.-S. Le, I. Oparin, A. Messaoudi, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Large vocabulary SOUL neural network language models," in *Proc. Interspeech'11*, 2011, pp. 1469–1472.
[8] L. Lamel, J.-L. Gauvain, V. Bac Le, I. Oparin, and S. Meng, "Improved models for Mandarin speech-to-text transcription," in *Proc. ICASSP'11*, 2011, pp. 4660–4663.
[9] L. Lamel, A. Messaoudi, and J.-L. Gauvain, "Automatic speech-to-text transcription in Arabic," *ACM TALIP, Spec. Iss. Arabic Natural Lang. Process.*, vol. 8, no. 4, pp. 1–18, 2009.
[10] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký, "Strategies for training large scale neural network language model," in *Proc. ASRU'11*, 2011, pp. 196–201.
[11] H. Le, A. Allauzen, G. Wisniewski, and F. Yvon, "Training continuous space language models: Some practical issues," in *Proc. EMNLP'10*, 2010, pp. 778–788.
[12] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. AISTATS'05*, 2005, pp. 246–252.
[13] A. Mnih and G. Hinton, "A scalable hierarchical distributed language model," *NIPS*, vol. 21, pp. 1081–1088, 2008.

[14] P. Brown, P. de Souza, R. Mercer, V. Della Pietra, and J. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, 1992.
[15] J. Goodman, "Classes for fast maximum entropy training," in *Proc. ICASSP'01*, 2001, pp. 561–564.
[16] A. Emami, "A Neural Syntactic Language Model," Ph.D. dissertation, Johns Hopkins Univ., Baltimore, MD, 2006.
[17] P. Xu and F. Jelinek, "Random forests and the data sparseness problem in language modeling," *Comput. Speech Lang.*, vol. 21, no. 1, pp. 105–152, 2007.
[18] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Černocký, "Recurrent neural network language modeling toolkit," in *Proc. ASRU'11 Demo Session*, 2011.
[19] S. Chen and S. Chu, "Enhanced word classing for Model M," in *Proc. Interspeech'10*, 2010, pp. 1037–1040.
[20] A. Emami and S. Chen, "Multi-class Model M," in *Proc. ICASSP'11*, 2011, pp. 5516–5519.
[21] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *JMLR*, vol. 3, pp. 1137–1155, 2003.
[22] J. Luo, L. Lamel, and J.-L. Gauvain, "Modeling characters versus words for Mandarin speech recognition," in *Proc. ICASSP'09*, 2009, pp. 4325–4328.
[23] L. Lamel, A. Messaoudi, and J.-L. Gauvain, "Investigating morphological decomposition for transcription of Arabic broadcast news and broadcast conversation data," in *Proc. Interspeech'08*, 2008, pp. 1429–1432.
[24] F. Diehl, M. Gales, M. Tomalin, and P. Woodland, "Morphological analysis and decomposition for Arabic speech-to-text systems," in *Proc. Interspeech'09*, 2009, pp. 2675–2678.
[25] I. Oparin, M. Sundermeyer, H. Ney, and J.-L. Gauvain, "Performance analysis of neural networks in combination with n-gram language models," in *Proc. ICASSP'12*, 2012, pp. 5005–5008.
[26] T. Mikolov, M. Karafiat, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech'10*, 2010, pp. 1045–1048.
[27] A. Allauzen, G. Adda, J. Crego, H. Bonneau-Maynard, H.-S. Le, A. Max, G. Wisniewski, F. Yvon, A. Lardilleux, A. Sokolov, and T. Lavergne, "Limsi @ WMT'11," in *Proc. VI Workshop Statist. Mach. Translat. WMT'11*, 2011, pp. 309–315.

Hai-Son Le is currently a Ph.D. student at the University Paris-Sud 11, working in LIMSI/CNRS as a member of Spoken Language Processing Group (TLP). His current research interests are Statistical Language Modeling, Statistical Machine Translation and Automatic Speech Recognition.

Ilya Oparin (M'11) is currently a researcher at the Spoken Language Processing group at LIMSI/CNRS, France. He holds a diploma cum laude (2003) in Theoretical and Applied Linguistics from the St.Petersburg State University in Russia and received a Ph.D. (2009) in Computer Science and Engineering from the University of West Bohemia, Czech Republic, having done research work in collaboration with Speech@FIT group at BUT in Brno. His research interests include speech recognition and statistical machine translation.

Alexandre Allauzen is Associate Professor in Computer Science at the University Paris-Sud and member of the Spoken Language Processing group of the LIMSI/CNRS. He received a Ph.D. (2003) in Computer Science from the University Paris-Sud in collaboration with INA, the French National Institute in charge of audiovisual archives. His main research interests include statistical language modeling, machine translation and automatic speech recognition.

Jean-Luc Gauvain (M'98) is a senior researcher at the CNRS, where he is head of the Spoken Language Processing Group at LIMSI. He received a doctorate in Electronics from the University of Paris-Sud 11 in 1982, and joined the CNRS as a permanent researcher in 1983. His primary research centers on large vocabulary continuous speech recognition and audio indexing. His research interests also include conversational interfaces, speaker identification, language identification, and speech translation. He has participated in many speech related projects both at the French National and European levels and has led the LIMSI participation in DARPA/NIST organized evaluations since 1992, most recently for the transcription of broadcast news data and of conversational speech. He has over 250 publications and received the 1996 IEEE SPS Best Paper Award in Speech Processing and the 2004 ISCA Best Paper Award for a paper in the Speech Communication Journal. He was co-editor-in-chief of the Speech Communication from 2007 to 2009.

François Yvon is Professor in Computer Science at the University Paris Sud and member of the Spoken Language Processing group of the LIMSI/CNRS. He was previously (1996–2007) associate professor in the Department of Computer Science at Telecom ParisTech. Françcois Yvon holds a Ph.D. in Computer Science and engineering degrees from the Telecom ParisTech and an engineering degree from the École Polytechnique. His main research interests include analogy-based and statistical language learning, speech recognition and synthesis and statistical machine translation.