International Journal of Neural Systems, Vol. 0, No. 0 (2005) 1–15

EFFICIENT EMBEDDED DECODING OF NEURAL NETWORKS LANGUAGE MODELS IN A MACHINE TRANSLATION SYSTEM

FRANCISCO ZAMORA-MARTINEZ
R&D Department, das-Nano S.L., Polígono Industrial Talluntxe II, Tajonar 31192, Spain
E-mail: pakozm@gmail.com

MARIA JOSE CASTRO-BLEDA
Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, València, Spain
E-mail: mcastro@dsic.upv.es

Neural Network Language Models are a successful approach to Natural Language Processing tasks, such as Machine Translation. We introduce in this work a Statistical Machine Translation system which fully integrates Neural Network Language Models in the decoding stage, breaking the traditional approach based on n-best list rescoring. The neural net models (both language models and translation models) are fully coupled in the decoding stage, allowing to more strongly influence the translation quality. Computational issues were solved by using a novel idea based on memorization and smoothing of the softmax constants to avoid their computation, which introduces a trade-off between language model quality and computational cost. These ideas were studied in a machine translation task with different combinations of neural networks used both as translation models and as target language models, comparing phrase-based and N-gram-based systems, showing that the integrated approach seems more promising for N-gram-based systems, even with non-full-quality Neural Network Language Models.

Keywords: Neural networks; Language modeling; Machine translation; Statistical machine translation; Embedded decoding.

1. Introduction

Neural Network Language Models (NNLMs) in machine translation, and their incorporation during decoding, is a highly active research area,[1–3] even more after recent deep learning breakthrough.[4,5] Indeed, language modeling is one of the most important parts in an Statistical Machine Translation (SMT) system. Formally, the goal of an SMT system is the translation of a sentence $\mathbf{f} = f_1 f_2 \dots f_{|\mathbf{f}|}$, for a given source language and source vocabulary $f_i \in \Sigma$, to an equivalent sentence $\hat{\mathbf{e}} = e_1 e_2 \dots e_{|\mathbf{e}|}$, for a certain target language and target vocabulary $e_i \in \Gamma$. Under the maximum entropy approach, the most likely sentence is searched by computing its probability with a log-linear combination of several models:[6,7]

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \prod_{m=1}^{M} h_m(\mathbf{f}, \mathbf{e})^{\lambda_m}, \qquad (1)$$

where M is the number of features, $h_m(\mathbf{f}, \mathbf{e})$ is a feature score function used for the translation of $\mathbf{f}$ into $\mathbf{e}$, and λ_m are the weights of the log-linear combination. Optimization of weights λ_m is performed during a tuning stage.

Two SMT approaches were followed in this work: phrase-based SMT and N-gram-based SMT. Both approaches use a similar modelization of the problem, differentiated in how the translation units are extracted from parallel corpora. Language Models (LMs) play a critical role in both approaches, more pronounced for the N-gram-based approach because the translation model is also an LM.

Continuous space representation of the lexicon proposes a better smoothing of unseen patterns, and it has been successfully applied in neural networks approaches to language modeling.[8–11] NNLMs[1,12] are the most popular approach in state-of-the-art SMT systems, presented in the form of feedforward neural networks. Nevertheless, the high computational cost of using NNLMs in decoding, particularly in SMT, limits their use and the traditional approach is to apply these NNLMs at a decoupled step using n-best list rescoring.[1,13,14]

Several practical solutions have been proposed to alleviate the computational problems associated with the softmax computation, such as adopting hierarchical versions of the softmax,[15–19] avoiding the normalization of the softmax,[3,20] speeding up training,[21–24] or using recurrent neural networks.[25,26]

In this same line, we have solved speed issues of NNLMs by memorizarion and smoothing of softmax constants for normalization. This paper describes this technique to speed up NNLMs in order to fulfill a coupled integration into an SMT decoder. Also, during decoding both the scores for the N-grams and the hidden state for contexts are cached to avoid potentially expensive forward propagation from the input to the hidden layer. Moreover, comparisons between phrase-based and N-gram-based approaches, and integrated NNLM decoding vs. n-best list rescoring, are presented. Different combinations of NNLMs used as translation models and as target LMs are also tested. A deep discussion about the trade-off between quality and speed is also stated. Experiments were accomplished in a machine translation task based on the News-Commentary 2010 corpora extracted from the WMT'10 evaluation campaign.

The paper is organized as follows. A brief introduction to NNLMs is described in Section 2. After posing the problem of high cost of using NNLMs in decoding, Section 3 reviews related solutions found in the literature for this problem, and also presents our contribution for fast testing. Section 4 is devoted to explain the proposed embedded decoding SMT system. Section 5 reviews both Phrase-based and N-gram-based SMT systems and Section 6 describes the experimentation and the analysis of the obtained results for a machine translation task. Finally, conclusions are drawn at Section 7, outlining our major contributions, and proposing future works in order to combine our approach with others.

2. Feedforward Neural Network Language Models

N-gram LMs are useful to estimate an approximation of the a priori probability of a given sentence $\mathbf{w}$ to be correct, using the assumption that word w_i only depends of the history composed by the $(N-1)$ previous words w_{i-N+1}^{i-1}:

$$p(\mathbf{w}) \approx \prod_{i=1}^{|\mathbf{w}|} p(w_i | w_{i-N+1}^{i-1}) \,. \tag{2}$$

NNLMs improve N-gram LMs conditional probabilities by using an automatic smoothing procedure for unseen patterns.[1,9,10] The word symbols are projected into a continuous space where similar words achieve similar projection, and probability is computed over this projections. The projection matrix is shared among all input positions (see Figure 1). Each output neuron estimates the conditional probability $p(w_i | w_{i-N+1}^{i-1})$, for a given word w_i of the vocabulary Ω. The input layer is composed of the sequence w_{i-N+1}^{i-1}, where each word is locally codified as a "1-to-$|\Omega|$" vector. Every input word is projected onto a much smaller set of projection neurons which learns the distributed codification of the words. The weights of all projection layers are shared, and the NNLM is able to learn both the distributed representation of words onto a continuous space and the conditional probability estimates of Eq. (2). After training, the projection layer can be removed from the network since it is much more efficient to replace it by a precomputed table of size $|\Omega|$ which stores the distributed encoding of each word.

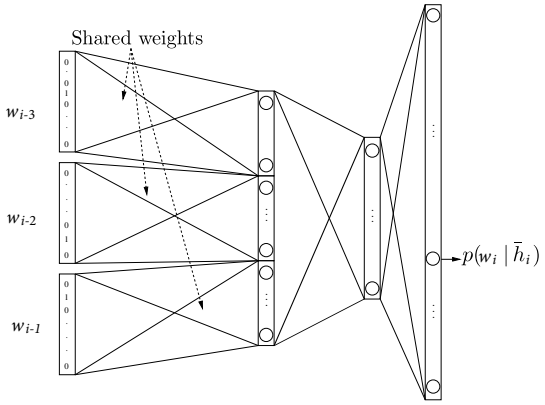


Figure 1. Instance of a 4-gram NNLM during training when computing $p(w_i|\bar{h}_i)$ with history $\bar{h}_i = w_{i-3}w_{i-2}w_{i-1}$. The input layer is composed by the three previous words, locally codified as "1-to-$|\Omega|$" vectors. The

projection layer learns the distributed codification of the words.

Big computational issues arise when the task vocabulary Ω is large, due to the NNLM output size, which needs one output neuron i for each $w \in \Omega$ to estimate $p(w|h)$, the probability of the word w given its history h, using the softmax activation function as follows:

$$o_i = \frac{\exp(a_i)}{\sum_{j=1}^{|\Omega|} \exp(a_j)} \tag{3}$$

a_i being the activations of neuron i.

The shortlist approach[1, 27–29] is widely used to solve this problem, training the NNLM over a restricted vocabulary $\Omega' \subset \Omega$ composed by the most frequent words in the training corpora. In our approach to NNLMs, Out-Of-Shortlist (OOS) word probabilities are computed adding a new special output neuron which estimates the joint probability of all OOS words, $p(OOS|w_{i-N+1}^{i-1})$. The value of this OOS neuron is smoothed using a unigram computed over all OOS words.

Similar alternatives are found in the literature. For instance, Ref.[1] uses a standard statistical N-gram (not only a unigram) to estimate the OOS word smoothing. Ref.[28] proposes to approximate the value of $p(OOS|w_{i-N+1}^{i-1})$ to 0, obtaining indistinguishable results compared to the previous approach, mostly due to the linear combination of NNLMs with a standard N-gram LM. In Ref.[30] the authors propose to consider equally the computation of OOS class probability by using the NNLM and withby using a standard N-gram. Therefore, the computation of OOS words probability is reduced to take $p_{NG}(w_i|w_{i-N+1}^{i-1})$ being p_{NG} the probability computed using the standard N-gram.

Finally, note that every NNLM probability computation is combined with a standard N-gram probability by introducing both models at the log-linear combination,[31] ensuring full task vocabulary coverage.

3. Fast Computation of Neural Network Language Models

A totally coupled integration of NNLMs into the decoder requires an efficient computation of NNLMs. The solutions found in the literature to circumvent the problem of high cost of NNLMs in decoding are reviewed. Secondlly, our ideas for fast evaluation of NNLMs are presented.

3.1. *Related work*

Several improvements have been reported since the shortlist approach for the vocabulary size issue. Among others, researchers from the Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI) have obtained significant improvements using Structured Output Layer Neural Network Language Models,[18, 19] which avoid the shortlist problem by allowing the use of the whole task vocabulary at NNLMs output. Basically, it consists of using a softmax layer to determine a word class and other softmax layers to determine the probability of a word given its class. A recent application of these new models in an N-gram-based SMT system with n-best list rescoring achieved large improvements.[13] Other solutions adopting hierarchical versions of the softmax to alleviate the complexity problem can be found in Refs.[15–17]

More recently, Devlin et al[3] present a novel formulation for a neural network joint model which augments the NNLM with a source context window which can be integrated into the SMT decoder. Their approach is based on self-normalization, that encourages the normalization constant to be close to 1 through a regularizer. At decoding time, the unnormalized scores from the softmax layer are used. Similar ideas applied to a speech recognition system are found in Ref.[20]

More promising ideas to speed up training will arise from the use of Noise Contrastive Estimation (NCE).[21–24] For instance, in Ref.,[14] n-best list rescoring and integration of target language models are compared. However, in that work, NNLM output probabilities are not normalized, what could be very task depending and could be harmful.

Finally, using recurrent neural networks for language modeling[25] is also a hot topic. A particularly relevant work, which applies recurrent neural networks to the N-gram formalism is presented in Ref.[26]

Different alternatives to speed up test or training of NNLMs can be combined to obtained further speedups. For instance, we propose a way to integrate the language model into the decoder using normalized probabilities values following an idea which is orthogonal to the NCE approach, so it is possible to combine both ideas in a system.

3.2. *Our approach to fast evaluation of NNLMs*

The number of LM look-ups at decoding stage (from thousands to millions depending on sentence length) force to use fast LMs. For this reason, NNLMs are used traditionally at a decoupled stage, rescoring lattices or n-best lists. NNLM bottleneck is located at its output activation function (typically softmax), which needs the computation of a normalization factor over the sum of all output neurons, even when only a few are needed.[1] Following Ref.[32] it is possible to speed up the NNLM by precomputation and memorization of the most frequent softmax normalization constants in a table, making possible NNLMs integration at decoding stage. Two solutions are given when a softmax normalization constant is not found in the table:

- On-the-fly Fast NNLM approach: compute the needed constant and store it in the table for future uses. This solution is equivalent in performance to using a standard NNLM.
- Smoothed Fast NNLM approach: use a model of lower order to compute the probability. Recursively, this solution uses the first model which has precomputed the constant in its table. The simplest model is a bigram NNLM because all softmax normalization constants could be stored at a table of size of the vocabulary. This approach is not equivalent to a standard NNLM, and lower quality results should be expected.

The smoothed Fast NNLM approach achieves more useful speedup, but it introduces a trade-off between quality and speed, because quality could be increased with the increment of precomputed softmax normalization constants, but decreasing the speedup of the system due to a major proportion of LM look-ups which will be computed using more complex models. The study of this trade-off in a medium-sized vocabulary SMT task is one of the contributions of this paper. For instance, a million of constants is required for the evaluation task proposed in this paper, which leads to an upper bound of 20MB of memory: 1 000 000 constants represented in a linear table, using a trie as the one proposed below, considerably reduce this number. Nevertheless, the memory requirements grow linearly with the number of constants, but the number of constants grow exponentially with the N order.

4. Embedded Decoding of Neural Network Language Models

The totally coupled integration of NNLMs into the decoder requires to take care with some peculiarities which will be explained in this section. A generalization of NNLMs to fully connected finite state automaton is described, and some remarks about caching are stated.

4.1. *Neural Network Language Models as a Full Finite State Automaton*

The integration into the decoding needs some LM hypotheses generation procedure in order to store the LM hypotheses in the active states of the decoder, and to update the hypotheses with the expansion of new incoming words. For standard N-grams, it is possible to use an automaton representation of the N-gram, using the automaton states as history (previous $N-1$ words) identifiers.[33]

An NNLM could compute the probability of full N-gram space (Ω^N N-grams of order N). Due to its huge size, it is impossible to expand the underlying automaton (take into account that the conversion of an NNLM into an automata has to deal explicitly with back-off to avoid an exponential increase of its size[34,35]), but on-demand approach is feasible. Like for standard N-grams, each NNLM history is identified by a state number at the full automaton. The states are enumerated on-demand using a trie data structure which keeps word sequences of length $N-1$, associating each trie node with an automaton state. The trie allows to look for a state number given its sequence of history words, and to retrieve the words sequence given the state number. Precomputed softmax normalization constants are stored as persistent paths in the trie, and decoder LM look-ups are stored as dynamic paths which will be erased with every sentence decoding end. In order to use the smoothed Fast NNLM approach, the same trie may store paths for NNLMs of different order.

4.2. *Caching LM look-ups during decoding*

Another important issue to increase performance of NNLM integration is to introduce a two-level cache of LM look-ups.

The first level stores complete N-gram probabilities: the key of the cache is a pair of ⟨automaton

state, next word⟩, and the stored value is the probability p(next word|automaton state). It is a cache with large number of entries where each entry needs 8 bytes for the key and 4 bytes for the value.

The second level stores NNLM hidden layer activations for a given N-gram history: the key is an automaton state, and the value is an array of size H, being H the number of hidden neurons. It is a cache with short number of entries because the size of each entry is 4 bytes for the key and $4H$ for the value.

5. Phrase-based and *N*-gram-based SMT Systems

The most popular statistical approach to translation is the so-called phrase-based SMT[36] (in most cases using Moses implementation[37]). However, N-gram-based SMT systems[38–41] have shown their capability to obtain state-of-the-art results, but their acceptance by the machine translation community is low.[13,41–43]

Phrase-based translation models allow lexical blocks with more than one word on either the source-language or target-language side, where the lengths may differ, eliminating the restrictions of word-based translation. The lexical blocks are found using statistical methods from corpora.[36]

N-gram-based SMT is based on finite state machine translation framework. The N-gram LMs are trained over a bilingual corpora aligned and segmented in a unique way, generating a sequence of bilingual units called *tuples*. The LM trained over these tuples is known as a bilingual translation model, and computes an approximation to the joint probability $p(\mathbf{e}, \mathbf{f})$ at sentence level. N-gram-based approach, by its nature, allows a tighter integration of NNLMs for both the target language model and the translation models, motivating the study presented in this work. Before the tuple generation, normally the bilingual corpora is monotonized, reordering source language sentences to follow the order of words at its corresponding target language sentence. N-gram-based decoders need to produce reordering hypotheses, which could be constrained to reduce the search space. The reordering search constraints are taken into account generalizing the Eq. (1) as follows:

$$(\hat{\mathbf{T}}, \hat{\varphi}) = \arg\max_{(\mathbf{T},\varphi)} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{T}, \varphi), \qquad (4)$$

where $\mathbf{T} = T_1 T_2 \dots T_{|\mathbf{T}|}$ is a sequence of target tuples $T_i = (x_i, y_i) \in \Delta$, with $x_i \in \Sigma^+$ (one or more source words), and $y_i \in \Gamma^\star$ (zero or more target words). The vocabulary Δ is the bilingual vocabulary of the N-gram translation model. The function $\varphi : \{1, 2, \dots, |\mathbf{f}|\} \to \{1, 2, \dots, |\mathbf{T}|\}$ associates a tuple index with each source word position of the sentence $\mathbf{f}$. The model restricts the order of source words inside a tuple to be always in an increasing order. The target sentence $\hat{\mathbf{e}}$ is extracted from the sequence $\hat{\mathbf{T}}$ taking into account the target part y_i of each tuple.

5.1. *Combination of LMs for translation*

Both SMT approaches to translation, phrase and N-gram based approaches, use a combination of models as shown in Eq. (1): the bilingual N-gram translation model, an N-gram target language model, phrase translation probabilities (direct and inverse), lexicon direct and inverse translation models, a weak distortion model, a lexicalized reordering model, word penalty, and tuple penalty. In our SMT system, both language models and translation models are NNLMs.

5.1.1. *The bilingual N-gram translation model*

The N-gram translation model[41] computes the approximation of $p(\mathbf{e}, \mathbf{f}) \approx p(\mathbf{T})$ over the composition of the source and target sentences into a sequence of bilingual tuples, given the sequence of tuples. This model is a Stochastic Finite State Transducer trained from the bilingual tuple segmented corpus, following the GIATI (Grammar Inference and Alignments for Transducer Inference) technique:[40]

$$\begin{aligned} h_{TrM}(\mathbf{T}, \varphi) = p(\mathbf{T}) &\approx p(T_1 T_2 \dots T_{|\mathbf{T}|}) \\ &\approx \prod_{i=1}^{|\mathbf{T}|} p(T_i | T_{i-N+1} \dots T_{i-1}) \end{aligned} \qquad (5)$$

where N is the order of the N-gram translation model. In our SMT system, this translation model is an NNLM, which will be called Neural Network Translation Model (NNTrM).

5.1.2. *The N-gram target language model*

The N-gram target language model computes the approximation of $p(\mathbf{e})$ as:

$$h_{TaLM}(\mathbf{T}, \varphi) = p(\mathbf{e}) \approx \prod_{i=1}^{|\mathbf{e}|} p(e_i | e_{i-N+1} \dots e_{i-1}) \quad (6)$$

where N is the order of the N-gram target language model. Again, an NNLM is used as the target language model in our translation system, Neural Network Target Language Model (NNTaLM).

5.1.3. *Other models*

Besides the translation and the target LMs, we added to the SMT system the following models:

- Lexicon direct and inverse translation models: Both are based on IBM-1 models.[44] Let $q(y_j|x_i)$ the direct statistical dictionary probability for the alignment of source word x_i with target word y_j:

$$h_{f2e}(\mathbf{T}, \varphi) = \prod_{i=1}^{|\mathbf{T}|} h'_{f2e}(T_i) \quad (7)$$

$$h'_{f2e}(x, y) = \frac{1}{(|x|+1)^{|y|}} \prod_{j=1}^{|y|} \sum_{i=0}^{|x|} q(y_j|x_i) \quad (8)$$

  being h_{e2f} and h'_{e2f} computed in a complementary way.
- Weak distortion model: It computes a penalization of the reordering of the output hypothesis penalizing adjacent tuples which source part is not consecutive. The order of words inside a tuple is not penalized.
- Lexicalized reordering model: Six different models were trained, based on the Moses lexicalized reordering model,[37] and they were computed over the current and previous tuple, and the φ function.
- Word and tuple bonus models: These two models penalize the number of inserted words and inserted tuples generated by the system.

5.2. *Decoders for SMT*

All experiments were conducted using our APRIL toolkit,[45,46] which implements the decoding of both phrase-based and N-gram-based SMT systems. The two approaches are similar and the decoding process is divided in three steps:[47] source sentence reordering hypotheses lattice generation, extension of previous lattice with tuples or phrases, and finally a Viterbi-based procedure which searches the best translation over a previous bilingual lattice. The major difference between the phrase-based and the N-gram-based decoder lies in the generation of translation options (see Figure 2 for an example):

- In phrase-based SMT, the source sentence lattice is extended with translation options spanning on source word positions in *increasing order* and which are *contiguous*. The Viterbi step searches over phrase lattice using the target LM and the reordering models.
- In N-gram-based SMT, the source sentence lattice is extended with tuples spanning on source word positions in *increasing order* but allowing *jumps*, so positions may not be contiguous. At the Viterbi step, besides the target LM and the reordering models, a bilingual N-gram translation model is used.

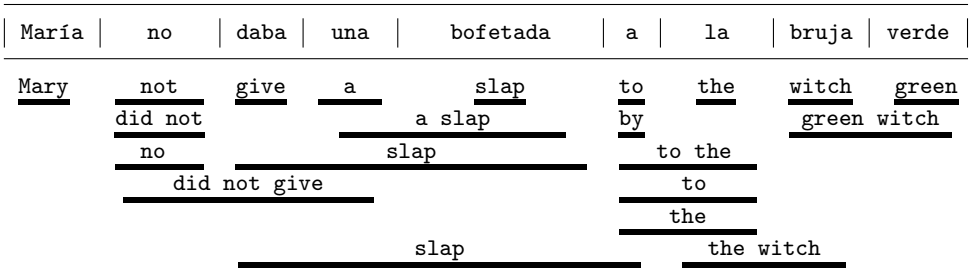
6. Experimentation in an SMT System

The translation experiments were performed with the medium-sized vocabulary News-Commentary 2010 Spanish-English task. Statistics extracted from the WMT'10 are shown in Table 1. All numbers are computed after cleaning, tokenization and lowercase preprocessing. The tokenization was done using the script `tokenizer.perl` from the WMT10. The English vocabulary was extracted from the 80.9K sentences of the News-Commentary 2010 corpus that comes from limiting sentence lengths to 40. The News2008 set was used as a development set; the News2009 set was used as an internal test set, for comparison purposes between systems. Finally, the News2010 set was used as a final test to measure the generalization ability of the full experimentation.

All translation systems were trained from Giza++[48] word alignments using the heuristic `grow-diag-final-and`.

N-gram baseline system combines 15 models in the log-linear search: two lexical translation scores (direct and inverse), two phrase conditional translation scores (direct and inverse), a 4-gram bilingual translation model and, six lexicalized reordering model (based on the Moses lexicalized reordering model[37]), one weak reordering model, an English 4-gram LM (trained on monolingual data), word penalty, and tuple penalty. N-gram LMs were trained with the SRI tookit,[49] and the NNLMs using our APRIL toolkit.[45,46]

(a) Translation options for a phrase-based decoder:



(b) In an N-gram-based SMT, all previous translation options were possible, and more:

daba bofetada	|||	slap
la verde	|||	green
...		...

Figure 2. (a) Translation options for a phrase-based decoder, and (b) two additional options for an N-gram-based decoder.

Table 1. NC-WMT'10 task. News2008 is used as a development set, News2009 as an internal set, and News2010 as a final test.

	Spanish		English		
Set	# Lines	# Words	# Lines	# Words	Voc. size
News-Commentary 2010	80.9K	1.8M	81.0K	1.6M	38,781
News2008	2.0K	52.6K	2.0K	49.7K	–
News2009	2.5K	68.0K	2.5K	65.6K	–
News2010	2.5K	65.5K	2.5K	61.9K	–

Monolingual English corpus

Set	# Lines	# Words
News-Commentary 2010	125.9K	2.97M

N-gram bilingual translation model corpus

Set	# Lines	# Tuples	Voc. size
News-Commentary 2010	80.9K	1.5M	231,981

The phrase-based system combines 14 models, the standard configuration of Moses.[37] All systems were optimized using the MERT procedure on the News2008 set. For comparative purposes, phrase-based systems were trained with Moses, and after tuned and run using Moses and APRIL.

We experimented with one or two NNLMs, depending if the NNLM was only used for the target language model, only for the bilingual translation model, or for both. NNLMs were trained using the same version of the Stochastic Back-propagation algorithm implemented in APRIL.

6.1. *Neural Network Target Language Model using the shortlist approach*

For the target language model, different N-gram orders were tested (values of $N = 2, 3, 4, 5$), which will be identified as NNTaLM-Ngr being N the order. Each model was a combination of three neural networks that differed only on the projection layer size ($P_j = 128, 160, 208$). A hidden layer with $H = 200$ neurons was selected, and a shortlist of the

$|\Gamma'| = 20$K most frequent words was used as an input and output vocabulary for the neural networks. The full vocabulary was formed by $|\Gamma| = 39$K words.

6.2. *Neural Network Translation Model based on statistical classes*

If the vocabulary of the task is large, using the most frequent words will only cover a small portion of the tuples of the translation model. For example, for the News-Commentary 2010 Spanish-English task extracted from the WMT'10, the full tuple vocabulary is $|\Delta| \approx 232$K (see Table 1), which implies that using a shortlist of the 20K more frequent tuples will be only cover a 16% of the 1.5M running tuples contained on training corpora, due to the big sparsity of tuple vocabularies. This proportion will make it useless, so the bilingual N-gram translation model is estimated over statistical classes similarly as in Ref.[50] following these steps:

(1) A distribution of tuples in C statistical classes is computed using command `mkcls` of Giza++.[48] This distribution is not ambiguous, that is, a tuple only belongs to one class. The number of classes C is a parameter that needs to be empirically estimated. The conditional probability of a tuple given its class is computed as:

$$p(z|c) = \frac{count(z|c)}{\sum_{z' \in \Delta} count(z'|c)} \tag{9}$$

being z a tuple of Δ, c a class of C, $count(z|c)$ the count of times that tuple z appears in class c. Note that in our formulation, due to the non-ambiguity of tuple-to-class relation, $count(z|c)$ is equal to $count(z)$.

(2) Every tuple in the training set was substituted by its related class, generating a new version of the training set. The translation model is estimated over this new training set instead of over raw tuples.

(3) In evaluation time, the conditional probability of the translation model is computed as:

$$p(T_i|T_{i-N+1}^{i-1}) \approx p(T_i|c_i) \cdot p(c_i|c_{i-N+1}^{i-1}) \,, \tag{10}$$

being T_i a tuple of hypotheses, c_i the class where tuple T_i belongs, and $p(c_i|c_{i-N+1}^{i-1})$ the probability computed by the class-based NNLM.

Different translation models were trained, changing the order of the N-gram $N = 2, 3, 4, 5$ and the number of statistical classes $C = 100, 300, 500, 1000$. Only the more promising combinations of N and C were tested. Each translation model was a combination of three neural networks that differed only in the projection layer size ($P_j = 64, 96, 128$). Each neural network had a hidden layer with $H = 200$ neurons. The input and output vocabulary were the class vocabulary.

6.3. *Analysis of the experimental results*

Different toolkit and MERT configurations were tested on the experimentation. Specifically, the configurations were:

- Moses: default Moses configuration using the same training and development/test sets.
- APRIL-PB: our decoder configured to use Phrase-based translation models. MERT procedure is executed using our decoder.
- Moses*: Moses configured to use the weights obtained as the best ones from APRIL-PB.
- APRIL-NG: our decoder using the N-gram-based translation models.

Performance of the tuning experiments with different Neural Network Target Language Models is shown in Table 2. All these results are obtained by integrating the NNLMs in the decoding stage, using the smoothed Fast NNLM approach. For comparison purposes, the PPL is measured linearly combining the NNLM with the standard N-gram, even when during decoding stage this combination is log-linear. The loss of PPL corresponding to the smoothed Fast NNLM is 8 absolute points in the worst case (approximately 4% relative loss), in contrast with the improvement of 44 points respect to the baseline (approximately 16% of relative improvement).

Relating automatic assessment measures for translation, several ones have been proposed in the literature to model the correspondence between the output produced by the SMT system and a reference. BLEU (BiLingual Evaluation Understudy)[51] and TER (Translation Edit Rate)[52] are currently ones of the most used in the field. BLEU is based on the geometric average of a modified N-gram precision, so that the higher the value the better. By contrast, TER is a modification of the Word Er-

Table 2. NC-WMT'10 task, tuning experiments: BLEU/TER for the News2009 set using different Neural Network Target Language Models (NNTaLM-Ngr). Perplexity (PPL) results are also shown linearly combining NNLMs and standard N-grams.

System	BLEU	TER	PPL	
APRIL-NG baseline	20.2	60.4	269	
			Fast NNLM	Std. NNLMs
+ NNTaLM-2gr	20.3	60.4	246	246
+ NNTaLM-3gr	20.6	60.2	231	227
+ NNTaLM-4gr	20.9	59.9	226	217
+ NNTaLM-5gr	20.8	60.0	225	213

ror Rate (WER) which allows to take into acount word reorderings, trying to evaluate the cost of a post-editing in order to correct the output of the SMT system. So, as with WER, the lower the value the better. The best BLEU/TER result is obtained by the $N = 4$ target language model The improvement respect to the baseline is 0.7 absolute points of BLEU (3.5% of relative improvement), and 0.5 absolute points of TER (0.8% of relative improvement).

Results with different Neural Network Translation Models are shown in Figure 3. The best model is NNTrM-300-4gr, a 4-gram translation model estimated with $C = 300$ statistical classes. The improvement respect to the baseline is 0.7 absolute points of BLEU (3.5% of relative improvement) and 0.4 absolute points of TER (0.6% of relative improvement).

Following, performance of combining the best 4-gram target language model with the two best translation models (4-grams with 300 and 500 classes) is shown in Table 3. It is observed that the best performance, obtained with the 4-gram translation model with 500 classes (NNTrM-500-4gr model) achieves an improvement of 1 absolute point of BLEU (5% of relative improvement) and 0.7 points of TER (1.2% of relative improvement). BLEU improvements are statistical significant under a "pairwise comparison" test[53] for a confidence interval of 95%.

The final results with the News2009 internal test set and the official test set News2010 are shown in Table 4.

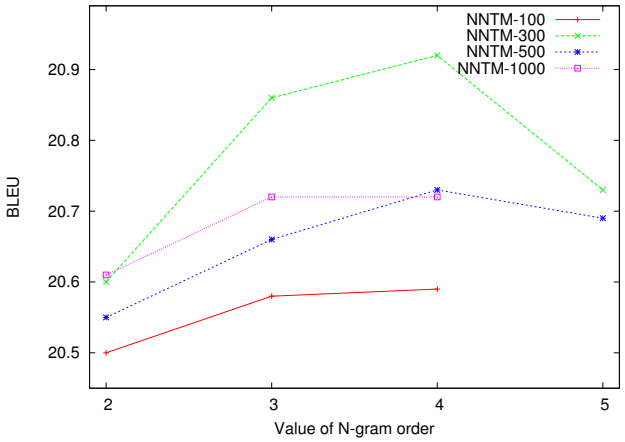


Figure 3. NC-WMT'10 task, tuning experiments using the News2009 set and different values of N (value of N-gram order) and C (statistical classes) for the Neural Network Translation Model (NNTrM).

6.3.1. *Baseline systems comparison*

Differences between Moses and our decoder APRIL are not significant (APRIL-PB, Moses*, APRIL-NG), being the Phrase-based translation models, APRIL-PB, the faster decoder.

6.3.2. *NNLMs in a totally coupled decoding algorithm on the News2010 test set*

Adding both neural network translation and target language models to the N-gram-based APRIL-NG system achieves an improvement of 0.9 BLEU points (4% of relative improvement), and 0.9 TER points (1.6%). These differences are statistically significant under a pairwise comparison test using a 95% confidence interval. On the other hand, adding a target

Table 3. NC-WMT'10 task, tuning experiments: BLEU/TER for the News2009 set combining a 4-gram Neural Network Target Language Model (NNTaLM-4gr) with 4-gram Neural Network Translation Models (NNTrMs) with different number of classes $C = 300, 500$.

System	BLEU	TER
APRIL-NG baseline	20.2	60.4
+ NNTaLM-4gr	20.9	59.9
+ NNTaLM-4gr + NNTrM-300-4gr	21.1	59.7
+ NNTaLM-4gr + NNTrM-500-4gr	21.2	59.7

Table 4. NC-WMT'10 task, final experiments: BLEU/TER for the News2009 internal test set and the official News2010 test set. NNTrM stands for Neural Network Translation Models and it is a 4-gram estimated with 500 classes (NNTrM-500-4gr). NNTaLM stands for Neural Network Target Language Model and it is a 4-gram (NNTaLM-4gr). The averaged number of seconds per sentence (Time) was measured for each system. Time of rescoring approach is the sum of both stages: decoding plus rescoring.

	News2009		News2010		
System	BLEU	TER	BLEU	TER	Time
Moses	20.4	60.3	22.6	57.8	0.6
APRIL-PB	20.6	60.3	22.7	57.8	0.4
Moses*	–	–	22.6	57.9	0.6
APRIL-NG	20.2	60.4	22.7	58.0	0.8
Integrating smoothed Fast NNLMs in the decoder					
APRIL-PB + NNTaLM	21.2	59.8	23.2	57.5	1.8
APRIL-NG + NNTaLM	20.9	59.9	23.2	57.4	1.8
APRIL-NG + NNTrM	20.7	60.0	23.3	57.6	1.6
APRIL-NG + NNTaLM + NNTrM	**21.2**	**59.7**	**23.6**	**57.1**	2.5
Integrating on-the-fly Fast NNLM (standard NNLMs) in the decoder					
APRIL-PB + NNTaLM	–	–	23.3	57.3	384.3
APRIL-NG + NNTaLM + NNTrM	–	–	23.7	57.1	177.3
Rescoring 2000-uniq-best list with standard NNLMs					
APRIL-PB + NNTaLM	21.1	59.9	23.4	57.3	3.9
APRIL-NG + NNTaLM	20.9	60.0	–	–	–
APRIL-NG + NNTrM	20.6	60.2	–	–	–
APRIL-NG + NNTaLM + NNTrM	21.1	59.8	23.5	57.4	4.3

language model to the Phrase-based APRIL-PB system achieves an improvement of 0.4 BLEU and TER points.

6.3.3. *NNLMs n-best list rescoring using 2000-uniq-best lists*

First of all, note that for rescoring, standard NNLMs are used instead of smoothed Fast NNLMs. For the N-gram-based APRIL-NG system, the integration of the translation model and the target language model in the decoding stage achieves better results than their use in a decoupled rescoring step for both News2009 and News2010 test sets. However, the Phrase-based APRIL-PB system obtains better results by using the rescoring approach than by the integrated approach. In all cases the differences are not statistically significant, though very interesting

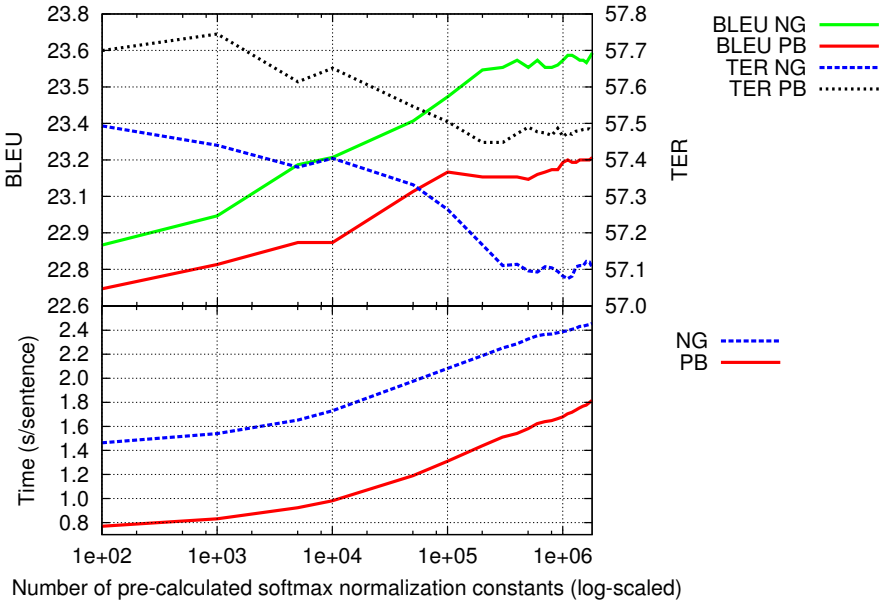


Figure 4. BLEU/TER and time versus number of pre-calculated softmax normalization constants during decoding of News2010 test set with smoothed Fast NNLM: PB refers to phrase-based system, and NG to N-gram-based system.

because, even when the integrated approach uses a "degraded" version of NNLMs, its quality is competitive compared with the rescoring approach. It is also important to notice here that integrated system is faster than the decoupled one (almost double faster, 1.8 averaged number of seconds per senteces versus 3.9, and 2.5 versus 4.3, respectively, see last column of Table 4). This comparison, in terms of performance and in terms of time, clarifies the merits of our approach.

6.3.4. *Trade-off between quality and time*

Figure 4 represents the evolution of BLEU and TER for N-gram-based APRIL-NG and Prase-based APRIL-PB systems, depending on the number of precomputed softmax normalization constants. APRIL-NG shows better accuracy when increasing the number of constants, due to the incorporation of two kinds of NNLMs, one for target LM and other for bilingual translation LM. This behavior is found for both figures, BLEU and TER. However, N-gram-based APRIL-NG needs more decoding time due to the same reason. Figure 5 shows the percentage of LM look-ups: a first cache level hit; a 4-gram NNLM hit; a 3-gram hit; or a bigram hit. The number of cache hits is not affected by the number of precomputed constants. For the target language model, the Prased-based APRIL-PB system shows more cache hits. The usage of order N NNLM is similar in all cases, showing that exists a big room for improvement increasing the number of 4-gram hits. The number of cache hits at the translation model decreases drastically compared to the target language model, which is the principal reason for the computational difference between APRIL-NG and APRIL-PB. Impressive speedup is achieved following the smoothed Fast NNLM (more than 70 times faster) compared to the integration of on-the-fly Fast NNLM (see Table 4).

6.3.5. *Comparing SMT systems output*

Finally, TER was computed between all pairs obtained from the following systems: the integrated system without smoothing, the integrated smoothed Fast NNLM system, and using rescoring of n-best lists in a non-integrated NNLM system (see Table 5). Integrated systems show more similar outputs, even when one is using standard NNLMs and the other smoothed Fast NNLMs. Differences between rescoring and integrated systems suggests deeper future research to improve the integration of NNLMs in the system.

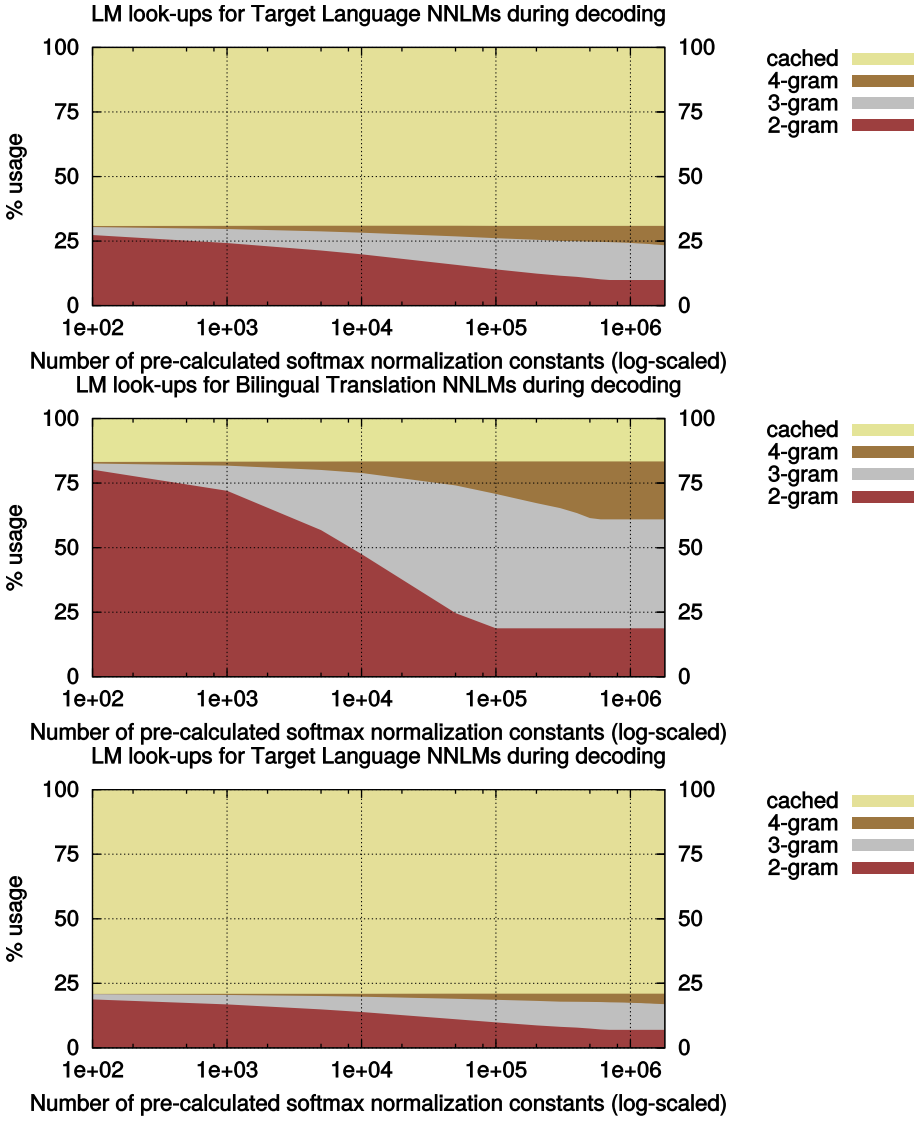


Figure 5. % usage of N orders versus number of pre-calculated softmax normalization constants during decoding of News2010 test set with smoothed Fast NNLM: (top) N-gram-based system % usage for the target language model (NNTaLM); (middle) N-gram-based system % usage for the translation model (NNTrM); (bottom) phrase-based system % usage for the target language model (NNTaLM).

Table 5. NC-WMT'10 task, final experiments for the official News2010 test set. Comparison of TER between best system outputs. Systems: Integrated *Standard* NNLM system without smoothing; *Integrated* smoothed Fast NNLM; *Rescoring* n-best list using a standard NNLM system.

	Phrase-based SMT			N-gram-based SMT	
	Integrated	Rescoring		Integrated	Rescoring
Standard	5.5	10.2	Standard	4.6	10.6
Integrated	–	9.0	Integrated	–	11.2

7. Conclusions

Different combinations of NNLMs used as both N-gram-based translation models and as target language models have been tested on a medium-sized

vocabulary task extracted from international machine translation evaluations. Our SMT system is one of the first to fully integrate NNLMs for both target and translation models into the decoding stage, breaking the traditional approach based on n-best lists rescoring in a decoupled stage. This integration into the decoder allows to more strongly influence the translation quality. To fulfill this integration, the computational issues associated with NNLMs were solved by memorization of the softmax normalization constants and by degrading the models using smoothing techniques. This approach leads to a system between two and three times slower than the reference system, but more than 70 times faster than an integrated NNLMs system without smoothing.

Our translation system is competitive with state-of-the-art Phrase-based Moses systems.[37] A class-based bilingual translation model trained as an NNLM is enough to enhance the results of a conventional N-gram-based system. Obviously, better translation models, not only based on statistical classes, would achieve better results, and a further research is needed in this line. The integration of "degraded" NNLMs (smoothed Fast NNLMs) in the decoding algorithm has been shown competitive with standard NNLMs used in an n-best list rescoring approach, slightly improving the results obtained with standard NNLMs and making the integrated system faster than the n-best list resconring approach. However, there is still a room for improvement on the coupling of NNLMs in the decoding stage of the search algorithm.

Thus, to summarize our primary contributions:

(1) A fast approach for using integrated NNLMs in the SMT decoder. To circumvent the expensive computation of the softmax normalization constants, we pre-compute and memorize some normalization constants for contexts of different orders (the most frequent ones), which are stored in a trie. At decoding time, smoothing, if needed, is performed by using the normalization constant for the largest subset of the context that is stored in their trie.
(2) Smart caching during decoding of both the scores for the N-grams and the hidden state for contexts. The hidden state caching avoids potentially expensive forward propagation from the input to the hidden layer.
(3) A detailed comparison of integrated decoding vs rescoring on a medium-scale vocabulary machine translation task. To do so, our phrase based and N-gram based machine translation systems were augmented with both neural network target language and translation models.

It is important to remark that the techniques presented in this work are compatible with many other improvements that have recently appeared:

- Integration of Structured Output Layer NNLMs[18] at the decoding stage is a very promising future work in order to better exploit the integrated system versus the rescoring n-best aproach.
- It is straightforward to combine the ideas presented in this paper using Noise Contrastive Estimation,[21,24] allowing substantial less time for training and evaluation using normalized output probabilities.

In the future, more experimentation with bigger sized vocabularies will be performed to study the scalability of the proposed approach.

Bibliography

1. H. Schwenk, Continuous space language models, *Comput. Speech and Lang.* **21**(3) (2007) 492–518.
2. D. Bahdanau, K. Cho and Y. Bengio, Neural machine translation by jointly learning to align and translate, *CoRR* **abs/1409.0473v6** (2015) 1–15.
3. J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz and J. Makhoul, Fast and robust neural network joint models for statistical machine translation, *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014, pp. 1370–1380.
4. Y. LeCun, Y. Bengio and G. Hinton, Deep learning, *Nature* **521** (05 2015) 436–444.
5. A. Ortiz, J. Munilla, J. M. Górriz and J. Ramírez, Ensembles of Deep Learning Architectures for the Early Diagnosis of the Alzheimers Disease, *Int. J. Neural Syst.* **26**(07) (2016) p. 1650025.
6. K. Papineni, S. Roukos and T. Ward, Maximum likelihood and discriminative training of direct translation models, *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998, pp. 189–192.
7. F. Och and H. Ney, Discriminative training and maximum entropy models for statistical machine translation, *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002, pp. 295–302.

8. H. Schwenk and J.-L. Gauvain, Connectionist language modeling for large vocabulary continuous speech recognition, *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002, pp. 765–768.
9. Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, A Neural Probabilistic Language Model, *J. Mach. Learn. Res.* **3**(2) (2003) 1137–1155.
10. M. J. Castro-Bleda and F. Prat, New Directions in Connectionist Language Modeling, *Computational Methods in Neural Modeling*, *LNCS* **2686** (Springer-Verlag, 2003), pp. 598–605.
11. H. Schwenk, D. Dèchelotte and J. L. Gauvain, Continuous space language models for statistical machine translation, *Proc. of the Joint Conference ACL/Coling*, 2006, pp. 723–730.
12. Y. Bengio, Neural net language models, *Scholarpedia* **3**(1) (2008) p. 3881.
13. L. H. Son, A. Allauzen and F. Yvon, Continuous Space Translation Models with Neural Networks, *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, 2012, pp. 39–48.
14. A. Vaswani, Y. Zhao, V. Fossum and D. Chiang, Decoding with large-scale neural language models improves translation, *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1387–1392.
15. F. Morin and Y. Bengio, Hierarchical probabilistic neural network language model, *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005, pp. 246–252.
16. A. Mnih and G. Hinton, A Scalable Hierarchical Distributed Language Model, *Advances in Neural Information Processing Systems*, **21** 2009, pp. 1081–1088.
17. T. Mikolov, A. Deoras, D. Povey, L. Burget and J. Cernocký, Strategies for Training Large Scale Neural Network Language Models, *Proc. of the Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011, pp. 196–201.
18. L. Hai-Son, I. Oparin, A. Alluzen, J.-L. Gauvain and F. Yvon, Structured Output Layer Neural Network Language Model, *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, **11**2011, pp. 5524–5527.
19. H. S. Le, I. Oparin, A. Messaoudi, A. Allauzen, J.-L. Gauvain and F. Yvon, Large vocabulary soul neural network language models., *Proc. of Interspeech*, 2011, pp. 1469–1472.
20. Y. Shi, W.-Q. Zhang, M. Cai and J. Liu, Efficient one-pass decoding with nnlm for speech recognition, *IEEE Signal Process. Lett.* **21** (2014) 377–381.
21. M. Gutmann and A. Hyvärinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 297–304.
22. A. Mnih and Y. W. Teh, A fast and simple algorithm for training neural probabilistic language models, *Proc. of the 29th International Conference on Machine Learning (ICML)*, 2012, pp. 1751–1758.
23. V. Mnih and G. E. Hinton, Learning to label aerial images from noisy data, *Proc. of the 29th International Conference on Machine Learning (ICML)*, 2012, pp. 567–574.
24. A. Mnih and K. Kavukcuoglu, Learning word embeddings efficiently with noise-contrastive estimation, *Advances in Neural Information Processing Systems*, **26** 2013, pp. 2265–2273.
25. T. Mikolov, S. Kombrink, L. Burget, J. Cernocký and S. Khudanpur, Extensions of recurrent neural network language model, *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011, pp. 5528–5531.
26. Y. Hu, M. Auli, Q. Gao and J. Gao, Minimum translation modeling with recurrent neural networks, *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2014, pp. 20–29.
27. A. Graves, S. Fernández, F. Gomez and J. Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, *Proc. of the 23rd International Conference on Machine learning (ICML)*, 2006, pp. 369–376.
28. A. Emami and L. Mangu, Empirical study of neural network language models for arabic speech recognition, *Proc. of the Workshop on Automatic Speech Recognition Understanding (ASRU)*, 2007, pp. 147–152.
29. F. Zamora-Martínez, V. Frinken, S. España-Boquera, M. J. Castro-Bleda, A. Fischer and H. Bunke, Neural network language models for off-line handwriting recognition, *Pattern Recogn.* **47**(4) (2014) 1642–1652.
30. J. Park, X. Liu, M. J. Gales and P. C. Woodland, Improved Neural Network Based Language Modelling and Adaptation, *Proc. of Interspeech*, 2010, pp. 26–30.
31. H. Schwenk and P. Koehn, Large and diverse language models for statistical machine translation, *Proc. of the International Joint Conference on Natural Language Processing (IJCNLP)*, 2008, pp. 661–666.
32. F. Zamora-Martínez, M. J. Castro-Bleda and S. España-Boquera, Fast Evaluation of Connectionist Language Models, *Proc. of the Int. Work-Conference on Artificial Neural Networks (IWANN)*, 2009, pp. 33–40.
33. C. Allauzen, M. Mohri and B. Roark, Generalized algorithms for constructing statistical language models, *Proc. of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, 2003, pp. 40–47.
34. E. Arisoy, S. F. Chen, B. Ramabhadran and A. Sethy, Converting neural network language mod-

els into back-off language models for efficient decoding in automatic speech recognition, *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 8242–8246.
35. R. Wang, M. Utiyama, I. Goto, E. Sumita, H. Zhao and B.-L. Lu, Converting Continuous-Space Language Models into N-gram Language Models for Statistical Machine Translation, *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 845–850.
36. P. Koehn, F. J. Och and D. Marcu, Statistical phrase-based translation, *Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technologies (NAACL HLT)*, 2003, pp. 48–54.
37. P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbst, Moses: open-source toolkit for statistical machine translation, *Proc. of the Association for Computational Linguistics (ACL)*, 2007, pp. 177–180.
38. F. Prat, F. Casacuberta and M. J. Castro, Machine Translation with Grammar Association: Combining Neural Networks and Finite-State Models, *Proc. of the Second Workshop on Natural Language Processing and Neural Networks*, 2001, pp. 53–60.
39. F. Casacuberta, E. Vidal and J. M. Vilar, Architectures for speech-to-speech translation using finite-state models, *Proc. of the Workshop on Speech-to-Speech Translation: Algorithms and Systems*, 2002, pp. 39–44.
40. F. Casacuberta and E. Vidal, Machine translation with inferred stochastic finite-state transducers, *Comput. Ling.* **30** (2004) 205–225.
41. J. B. Mariño, R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. Fonollosa and M. R. Costa-jussà, N-gram-based Machine Translation, *Comput. Ling.* **32** (2006) 527–549.
42. M. R. Costa-jussà, J. M. Crego, P. Lambert, M. Khalilov, J. A. Fonollosa, J. B. Mariño and R. E. Banchs, Ngram-based statistical machine translation enhanced with multiple weighted reordering hypotheses, *Proc. of the Second Workshop on Statistical Machine Translation (WMT)*, 2007, pp. 167–170.
43. H.-S. Le, T. Lavergne, A. Allauzen, M. Apidianaki, L. Gong, A. Max, A. Sokolov, G. Wisniewski and F. Yvon, Limsi@wmt12, *Proc. of the Workshop on Statistical Machine Translation (WMT)*, 2012, pp. 330–337.
44. P. F. Brown, V. J. Della-Pietra, S. A. Della-Pietra and R. L. Mercer, The mathematics of statistical machine translation: parameter estimation, *Comput. Ling.* **19**(2) (1993) 263–311.
45. S. España-Boquera, F. Zamora-Martínez, M. J. Castro-Bleda and J. Gorbe-Moya, Efficient BP Algorithms for General Feedforward Neural Networks, *Bio-inspired Modeling of Cognitive Tasks*, *LNCS* **4527** (Springer, 2007), pp. 327–336.
46. F. Zamora-Martínez, S. España-Boquera, J. Gorbe-Moya, J. Pastor-Pellicer and A. Palacios-Corella, APRIL-ANN toolkit, A Pattern Recognizer In Lua with Artificial Neural Networks (2013), https://github.com/pakozm/april-ann.
47. F. Zamora-Martínez, M. J. Castro-Bleda and H. Schwenk, N-gram-based Machine Translation enhanced with Neural Networks for the French-English BTEC-IWSLT'10 task, *Proc. of the 7th International Workshop on Spoken Language Translation (IWSLT)*, 2010, pp. 45–52.
48. F. J. Och and H. Ney, A Systematic Comparison of Various Statistical Alignment Models, *Comput. Ling.* **29**(1) (2003) 19–51.
49. A. Stolcke, SRILM: an extensible language modeling toolkit, *Proc. of the International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 901–904.
50. T. Mikolov, S. Kombrink, L. Burget, J. Cernocky and S. Khudanpur, Extensions of recurrent neural network language model, *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5528–5531.
51. K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, BLEU: A Method for Automatic Evaluation of Machine Translation, *Proc. of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, 2002, pp. 311–318.
52. M. Snover, B. Dorr, R. Schwartz, L. Micciulla and J. Makhoul, A study of translation edit rate with targeted human annotation, *Proc. of the Association for Machine Translation in the Americas*, 2006, pp. 223–231.
53. P. Koehn, Statistical significance tests for machine translation evaluation, *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, 2004, pp. 388–395.