RESEARCH ARTICLE

Fallback Variable History NNLMs: Efficient NNLMs by precomputation and stochastic training

Francisco J. Zamora-Martínez[1], **Salvador España-Boquera**[2], **Maria Jose Castro-Bleda**[2]*, **Adrian Palacios-Corella**[2]

1 R&D Department, Veridas S.L., Pol. Ind. Talluntxe II, Tajonar 31192, Spain, **2** Universitat Politècnica de València, Valencia, Spain

* mcastro@dsic.upv.es

Citation: Zamora-Martínez FJ, España-Boquera S, Castro-Bleda MJ, Palacios-Corella A (2018) Fallback Variable History NNLMs: Efficient NNLMs by precomputation and stochastic training. PLoS ONE 13(7): e0200884. https://doi.org/10.1371/journal.pone.0200884

Editor: Marco Maggini, Universita degli Studi di Siena, ITALY

Received: May 22, 2017

Accepted: July 4, 2018

Published: July 26, 2018

Data Availability Statement: Data and software to carry out our experiments: -Data: News-Commentary corpus, http://opus.lingfil.uu.se/News-Commentary.php -Software: +Moses: http://www.statmt.org/moses/ +APRIL-ANN: https://github.com/april-org/april-ann.

Funding: This work was partially supported by the Spanish MINECO and FEDER founds under project TIN2017-85854-C4-2-R (to MJCB). The funders had no role in study design, data collection and

Abstract

This paper presents a new method to reduce the computational cost when using Neural Networks as Language Models, during recognition, in some particular scenarios. It is based on a Neural Network that considers input contexts of different length in order to ease the use of a fallback mechanism together with the precomputation of softmax normalization constants for these inputs. The proposed approach is empirically validated, showing their capability to emulate lower order *N*-grams with a single Neural Network. A machine translation task shows that the proposed model constitutes a good solution to the normalization cost of the output softmax layer of Neural Networks, for some practical cases, without a significant impact in performance while improving the system speed.

1 Introduction

Neural Network Language Models (NNLMs) have drawn the attention of the Natural Language Processing community due to their ability to learn continuous word representations, which allows for a better generalization than that of count-based *N*-grams by using either feed-forward or recurrent Neural Networks (NNs) [1–5]. Speech recognition, handwritten recognition and machine translation are good examples of tasks where NNLMs have shown to improve performance.

1.1 Neural networks language models

The aim of a Language Model (LM) is to estimate the probability of a word w_i given the previous ones, $p(w_i|w_1 \ldots w_{i-1})$. An NNLM is a language model based on NNs, exploiting their ability to learn continuous word representations [1, 2, 6–8]. Feed-forward neural networks are used for language modeling based on the *N*-gram approximation since the past history of word w_i, defined as h_i, is limited to the $N-1$ previous words: $p(w_i|h_i) = p(w_i|w_{i-N+1} \ldots w_{i-1})$. In these NNLMs, the input is composed of the sequence $w_{i-N+1}, \ldots, w_{i-1}$. For a given vocabulary Ω, words are represented by a one-hot or 1-of-$|\Omega|$ vector, leading to a huge NN for large vocabulary tasks. To overcome this problem, a word embedding can be learned by means of a

analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

projection layer, mapping each word into a lower-dimensional real-valued space and sharing the weight matrix [1, 2] (note that the projection layer can be later replaced by a table storing the learned distributed encoding of each word). There exist, nevertheless, other alternatives to estimate the local embedding [6], and the chosen procedure is irrelevant for the approach described in this work.

After this input and the optional projection layer, there can be one or more hidden layers with non-linear activation functions. The last hidden layer is connected to the output layer, usually based on the softmax function to represent the N-gram LM probability distribution:

$$p_j = P(w_i = j | h_i) = \frac{exp(o_j)}{\sum_{k=1}^{|\Omega|} exp(o_k)} \tag{1}$$

where o_k is the output of output neuron k before applying the softmax normalization.

The computational complexity of this NNLM is dominated by the size of this output layer, which needs one neuron for each vocabulary word. This becomes a bottleneck, even if only a few outputs are required, since the computation of the normalization factor requires the sum of all output neurons.

1.2 Previous work

Different approaches have been proposed to overcome the computational cost of computing the normalization term during training. Among them, we can mention importance sampling of the partition function [9] and the Noise-Contrastive Estimation [10]. On the other side, the shortlist approach [2, 11] and hierarchical NNLMs (also called structured output NNLMs) [12, 13] reduce the computational cost of evaluating the models by limiting the size of the softmax layer either by shortening the output vocabulary, or by using some kind of tree structure to hierarchically distribute words into classes, respectively.

While some other approaches elude computing the softmax denominator by training the network with a regularization term to reduce its variance [14, 15], ignoring these normalization constants still degrades the model [16]. We can conclude that these computational issues are not completely solved yet. Indeed, none of the aforementioned solutions has been adopted as the best or as standard by the community.

Relating the reduction of the computational cost during evaluation, we can distinguish between two different scenarios: on the one side, we may need to compute the whole set of outputs and, on the other side, only a few outputs are often required. In the first case, the computation of the partition function is not the bottleneck and approaches such as the hierarchical output may be counterproductive: only approaches such as the differentiated softmax [17], when the overall number of arithmetic operations is reduced, can make a difference.

Let us also observe that, if we are only interested in determining which are the most probable outputs of the network, without requiring their actual normalized values, we can apply several techniques to speed up this process since it is essentially the search of a maximum inner product [18].

The computation of the softmax denominator term is more problematic in the second scenario, where only a little subset of the overall output must be computed in spite of the fact that the rest of the outputs should also be computed, in principle, only to obtain the normalization term.

The description of some practical cases where computing a limited subset of outputs makes sense illustrates the relevance of this scenario: in a speech or in a handwriting recognition task based on HMMs, the system is guided by an LM. This LM can be based on sub-word units (e.g. character-based LMs) or, more usually, word-based. In these word-based systems, the lexicon decoder (e.g. a tree lexicon) proposes a set of word hypotheses whose number is usually much lower than the vocabulary size (e.g. only a few part of the vocabulary is activated each time), so that the LM look-ups are only required for this limited subset. Moreover, in order to properly apply dynamic programming, and due to the fact that hypotheses from different alignments are taken into account, LM scores should be normalized, which makes the use of the softmax normalization unavoidable (for NNLMs using this output layer).

Besides HMMs, more recent recognition systems based in deep learning techniques usually employ the Connectionist Temporal Classification (CTC) loss function [19]. In spite of the fact that CTC based systems are able to emit characters or other units in the absence of an LM, it is possible to include them in several ways either by constraining the paths and using some kind of beam search (instead of greedy decoding) or even by means of re-ranking over the best hypotheses [20]. Similarly, the use of an LM can also be adapted to the encoder-decoder approach usually employed for automatic translation [21].

1.3 Proposed approach

Our work, aiming at reducing the computational cost during evaluation, is based on precomputing some values beforehand. The idea of precomputing values for this purpose is not new: it has been applied to the first hidden layer of NNLMs [15] and to the output layer as well [7, 22, 23]. In [22], a fast locality-sensitive hashing technique is used, together with a sublinear nearest neighbor search, to determine a set of outputs with the highest probabilities. The normalization factor is then computed by discarding the values not present in this set. [23] also proposes the use of approximate nearest neighbor search but makes use of kernel feature maps.

In [7], softmax normalization constants are directly stored for some inputs (e.g. the most probable ones from those appearing in the training set). When an input pattern is not found, we can either compute the denominator on the fly or, much faster, we can apply a fallback mechanism consisting in using a lower order *N*-gram model. This last approach reminds us the backing off mechanism commonly used in count-based *N*-gram LMs [24].

A more extreme precomputation to speed up NNLMs consists in converting them into standard count-based back-off models [25, 26].

The approach described in this work can be considered a simplification of [7]. It consists in using a sole NN that considers input contexts of different length in order to ease the use of the aforementioned fallback mechanism. In this way, when the normalization constant of the full word history is not found in the table, the model emulates a lower order one by replacing the farthest word of the past history for a ⟨DUMMY⟩ symbol. This procedure is applied, if required, until the word history is found in the table. This mechanism is guaranteed to succeed at least when descending down to bigrams, since an array of the same size as the lexicon suffices to cover all possible patterns.

The capability of the NN to emulate lower order models is obtained by properly training it with ⟨DUMMY⟩ inputs, as explained below. This technique reduces the number of model parameters and the computational resources, specially the training cost, initially required in [7] since, now, a single NN replaces several models.

Experiments have been performed for a pure LM task and for a machine translation system, where it is shown that the introduced approximation does not impact performance, while being faster to train and test than related models. Although some decisions have been made in these experiments (e.g. the shortlist approach), we have to remark that the proposed approach is not tied to these decisions and can be used in addition to other techniques.

2 Fallback Variable History NNLM

As stated in the introduction, our goal was to obtain a single NN able to estimate LMs of different orders to easily implement a fallback mechanism based on the memorization of precomputed normalization constants. This is achieved by using an input encoding able to represent contexts of variable size. The proposed model will be called "Fallback Variable History NNLM" (Fallback V-NNLM).

2.1 Emulating lower order *N*-grams by stochastic training

Skipping techniques, in the context of language modeling, consist in estimating several LMs based on different ways of removing words from the past history [27–32]. A natural way of applying this idea to NNLMs is by training a single NN but replacing random positions in the input representing the LM history by a new special null symbol $\langle\text{DUMMY}\rangle$. This idea can also be used to emulate lower order *N*-grams by skipping the required number of farthest words of the context, leading to a sole model, called Variable History NNLM (V-NNLM), that can emulate all the lower orders.

Thus, when requiring the evaluation of different word history lengths, a single NN suffices and a single NN needs to be trained instead of $N - 1$ NNs. To do so, patterns associated with several context sizes (containing the $\langle\text{DUMMY}\rangle$ symbol in zero or more positions of the input layer) should be used.

This can be easily achieved by stochastically perturbing the input. This is equivalent, but simpler, to replicate the training data for each *N*-gram order (replacing the farthest words by the $\langle\text{DUMMY}\rangle$ and shuffling the patterns afterwards). The *N*-gram order chosen for each training pattern has been stochastically sampled from an a priori distribution (a uniform distribution in the reported experiments).

Moreover, in order to achieve a better generalization, several models have been trained so that our final V-NNLM will be an ensemble of them. This ensemble could, in turn, be distilled into just one model afterwards using the ideas of [33].

2.2 The fallback strategy

Let us see how to combine V-NNLMs, based on emulating $N - 1$ NNLMs into one NNLM, and the idea of precomputing normalization constants. To this end, let us detail the fallback mechanism. It is convenient to distinguish the procedures required beforehand and at test time during decoding.

Once the V-NNLM has been trained, the softmax normalization constants of the most probable input contexts have to be precomputed and stored for each *N*-gram order down to bigrams. Bigrams deserve a special treatment since the entire set of contexts, which is of size $|\Omega|$, is precomputed. In practice, we can take as the most probable contexts from the set of *N*-grams those with a count greater than a given threshold.

The required process to implement the fallback mechanism in test time is as follows:

1. Take into account the input of the V-NNLM consisting of an $(N - 1)$-gram.
2. Look for the constant associated with this input. If the constant is not found in the table associated with the input length, remove the last element from the input, and go back to 2.
3. Fill the input context with the $\langle\text{DUMMY}\rangle$ symbol in order to obtain the original input size and evaluate the V-NNLM up to the last hidden layer. Note that this step is compatible with the precomputation of values to speed up the first hidden layer as described, for instance, in [15].

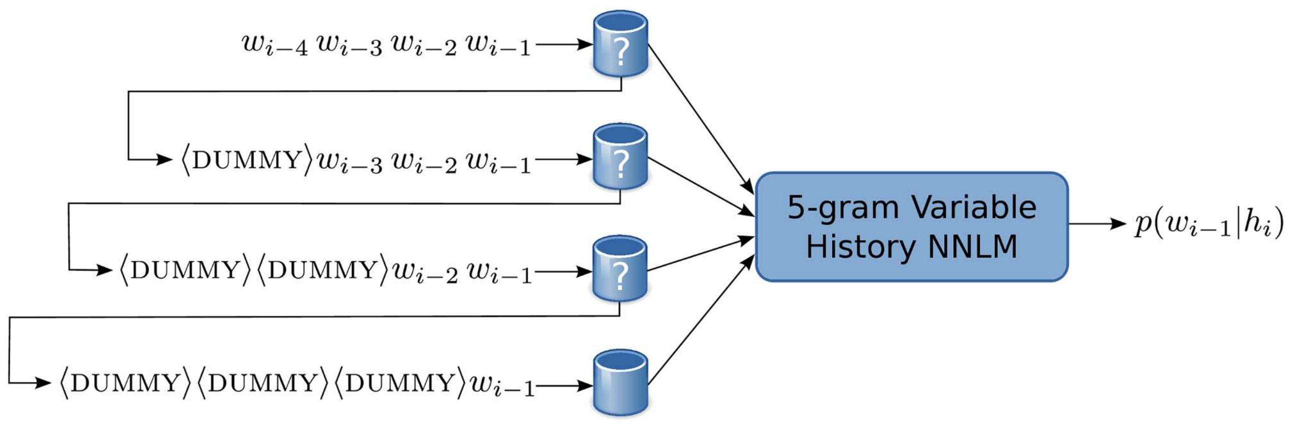
$w_{i-4}\, w_{i-3}\, w_{i-2}\, w_{i-1}$
?
$\langle\text{DUMMY}\rangle w_{i-3}\, w_{i-2}\, w_{i-1}$
?
$\langle\text{DUMMY}\rangle\langle\text{DUMMY}\rangle w_{i-2}\, w_{i-1}$
?
$\langle\text{DUMMY}\rangle\langle\text{DUMMY}\rangle\langle\text{DUMMY}\rangle w_{i-1}$
5-gram Variable History NNLM
$p(w_{i-1}|h_i)$

Fig 1. Scheme of a 5-gram Fallback Variable History NNLM (Fallback V-NNLM). It is composed of one 5-gram Variable History NNLM (V-NNLM) and four precomputed tables of constants. If the softmax normalization constant is found in the 5-gram table for the input $w_{i-4}w_{i-3}w_{i-2}w_{i-1}$, the query is processed and the probability $p(w_i|h_i) = p(w_i|w_{i-4}w_{i-3}w_{i-2}w_{i-1})$ is computed. If not, the query is delegated to the same V-NNLM but with the input $\langle\text{DUMMY}\rangle$ w_{i-3} $w_{i-2}w_{i-1}$ at the 4-gram table, and so on.

https://doi.org/10.1371/journal.pone.0200884.g001

4. Evaluate the neurons of the output layer for the set of desired words and use the constant found in step 2.

Let us remind that point 2 is guaranteed to succeed since the table associated with bigrams contains the entire vocabulary. This whole process, illustrated in Fig 1, composes the Fallback Variable History NNLM (Fallback V-NNLM) model.

Compared with [7], a slight degradation in perplexity (PPL) and system performance could be expected from the fact that a ⟨DUMMY⟩ symbol is used to shorten the input context of the NN instead of training a dedicated model for each *N*-gram order. However, as observed in the experimental results, the performance loss is negligible in our case study and this is compensated by a simpler implementation, a reduced number of model parameters and a faster training process.

3 Experimental Setup I: Emulating lower order *N*-gram NNLMs

The following experiments will test the capability of V-NNLMs to emulate lower order *N*-grams.

3.1 Corpus

The proposed experimentation is based on the English part of the News-Commentary 2010 corpus, and the test sets of 2008 and 2010 editions of the *Workshop of Machine Translation* [34, 35]. Table 1 shows some statistics of this database (English part) which will be used to train the proposed LMs.

Table 1. Lines and words of the News-Commentary corpus (English part).

Set	# lines	# words
Training (News-Commentary 2010)	125.8K	2.9M
Validation (News 2008)	2.0K	49.7K
Test (News 2010)	2.5K	61.9K
Total	130.3K	3.0M

https://doi.org/10.1371/journal.pone.0200884.t001

3.2 Model training

Many techniques aiming at reducing the computational cost of the output normalization layer are independent and compatible with the approach described in this work. We have combined one of them, namely, the shortlist approach [2, 11]. The shortlist approach consists in training the NNLM over a restricted vocabulary (known as *shortlist*) $\Omega' \subset \Omega$ composed by the most frequent words in the training corpora.

The vocabulary of the training set consists of $|\Omega| = 38\,793$ words. In our work, the shortlist is fixed to the $|\Omega'| = 20\,000$ most frequent words. In addition to the shortlist approach described previously for the output layer, we have measured the effect of using the same shortlist as the input vocabulary: every Out-Of-Shortlist (OOS) word (i.e. those from $\Omega - \Omega'$) is replaced by the OOS identifier at the input as well. We obtained indistinguishable results in previous experiments by using the whole vocabulary or the shortlist as the input vocabulary [36]. For that reason, we decided to use the restricted vocabulary (shortlist words plus a neuron associated with the OOS words) both at the input and at the output layer of the neural network.

OOS word probabilities can be computed by adding a new output neuron to compute $p(OOS|w_{i-n+1}\ldots w_{i-1})$. That is, o_{OOS} is the activation of the output neuron corresponding to all OOS words, and is an estimate of $\sum_{w \in OOS} p(w|w_{i-n+1}\ldots w_{i-1})$. This probability mass is then distributed among all OOS words. We have followed the approach described in [37] where a standard count-based unigram computed over OOS words is used in this distribution.

Backpropagation algorithm and L2 regularized cross-entropy loss function are used to train the networks: we have trained not only the V-NNLM but also a different NNLM for each N-gram order, from bigrams up to 5-grams, in order to evaluate the V-NNLM capability to emulate lower order models. Three different NNs have been trained for each model in order to perform an ensemble by means of a linear combination, leading to a total of 15 NNs. The three models combined in each case differ in the projection layer which comprise 128, 160, and 208 neurons, respectively. All of them have a hidden layer with 200 neurons. These numbers are based on previous experimentation [38], and were selected to improve PPL on the validation set.

In order to train the V-NNLM to deal with inputs of several context lengths, the context length of each training pattern is stochastically sampled from a uniform distribution ranging from 1 up to $N - 1$ and the appropiate number of $\langle \text{DUMMY} \rangle$ symbols are used to fill the input at the farthest positions.

After training the networks, the tables of precomputed softmax constants, for each possible N-gram order (ranging from bigram up to 5-grams) are precomputed. The size of these tables, which depend on the number of distinct N-grams found in the training data, are $20K$ for bigrams, $650K$ for trigrams, $1.79M$ for 4-grams and $2.42M$ for 5-grams, which are compactly stored by means of a trie data structure based on hashing.

Finally, in order to study the effect of the initialization weights of the different NNs, we have replicated the whole experiment 15 times by using, at each replica, a different set of random initialization seeds. Since each replica comprises the training of 15 NNs, the total number of trained NNs amounts to a total of 225.

3.3 Experiment: Emulating lower order *N*-grams with V-NNLMs

In order to evaluate the capability of V-NNLMs to emulate a set of standard NNLMs, we have measured the PPL of the validation and test sets by using the regular NNLMs of different N-gram orders, on the one hand, as well as the trained 5-gram V-NNLM, on the other. These values are shown in Table 2. Reported values correspond to the averaged PPL over 15 different

Table 2. Averaged PPL measures for the News-Commentary validation and test sets. Measures given by regular NNLMs and by 5-gram V-NNLMs without using the fallback method. No combination with count-based models has been performed.

Model	Validation set PPL			
	N-gram order			
	2	3	4	5
Bigram NNLM	412.6 ± 1.0	–	–	–
Trigram NNLM	–	345.9 ± 0.9	–	–
4-gram NNLM	–	–	327.0 ± 1.1	–
5-gram NNLM	–	–	–	319.4 ± 1.0
V-NNLM	423.3 ± 1.3	356.3 ± 0.9	334.9 ± 0.8	326.5 ± 0.7
Model	Test set PPL			
	N-gram order			
	2	3	4	5
Bigram NNLM	407.7 ± 1.1	–	–	–
Trigram NNLM	–	338.6 ± 1.0	–	–
4-gram NNLM	–	–	320.6 ± 1.1	–
5-gram NNLM	–	–	–	312.6 ± 0.9
V-NNLM	418.2 ± 1.3	350.0 ± 1.0	329.4 ± 0.9	320.2 ± 0.8

https://doi.org/10.1371/journal.pone.0200884.t002

experiments (as mentioned in the previous section), accompanied by the corresponding 95% confidence intervals. Let us remark that the V-NNLMs results are computed without using the fallback method.

Although the (averaged) PPL values for each *N*-gram NNLM are slightly better than the ones obtained by the corresponding V-NNLMs, as expected, the difference is not significant enough to harm the translation system performance, as we will observe in Section 4. In contrast, the advantages of using the V-NNLM include the fact of training and using a sole NN, instead of $N - 1$ NNs, to implement the fallback approach. We believe that these advantages clearly outweight the slightly observed degradation in PPL. We can also observe, from the confidence intervals, that the different initialization of the NN weights does not produce relevant differences in the observed PPL.

3.4 Experiment: Comparing standard LMs and Fallback V-NNLM

In this section, we are going to compare the PPL for the validation and test sets given by standard LMs and by the Fallback V-NNLM. The Fallback V-NNLM is composed by just one V-NNLM of maximum *N*-gram order and by $N - 1$ tables of precomputed softmax normalization constants.

For this experimentation, a count-based 4-gram has been trained with the SRI toolkit [39], and linearly combined with the Fallback V-NNLM as it is usually done with standard NNLMs. SRI has been configured to use modified Kneser-Ney smoothing with interpolation and unknown word probability computation. The linear combination weights are optimized to minimize the PPL in the validation set.

For comparison purposes, SRI models and a regular NNLM (where the normalization constant is actually computed) for each *N*-gram were also tested. The PPL measures, averaged over the 15 different experiments, are shown in Table 3. These values exhibit the excellent behaviour of the proposed model with respect to the corresponding SRI models in isolation. We can also observe that the gap from regular NNLMs to Fallback V-NNLMs is very small. Again, the low value of the confidence interval widths confirms the fact that the differences in

Table 3. Averaged PPL measures for the News-Commentary validation and test sets. Measures given by SRI *N*-gram models, by regular NNLMs, and by the proposed Fallback V-NNLMs. Let us remark that both NNLMs and Fallback V-NNLMs are linearly combined with the 4-gram count-based model.

	Validation set PPL		
N-gram order	SRI	NNLM	F.V-NNLM
2	332	252.0 ± 0.2	252.6 ± 0.1
3	308	231.1 ± 0.3	237.9 ± 0.2
4	305	221.0 ± 0.4	233.0 ± 0.2
5	305	216.6 ± 0.4	232.2 ± 0.2
	Test set PPL		
N-gram order	SRI	NNLM	F.V-NNLM
2	409	244.8 ± 0.2	245.4 ± 0.1
3	377	223.4 ± 0.3	230.3 ± 0.2
4	297	213.8 ± 0.4	225.8 ± 0.2
5	297	209.0 ± 0.3	225.0 ± 0.3

https://doi.org/10.1371/journal.pone.0200884.t003

the initialization of the NN weights does not produce noticeable differences in PPL. For that reason, the machine translation experiments of the following section will be conducted using only one of the 15 identical experimentation replicas.

4 Experimental Setup II: Models in a machine translation system

In this experimental framework, the proposed Fallback V-NNLM is used for a translation task.

4.1 Corpus

The experiments were also performed with the Spanish-English task of the News-Commentary 2010 corpus, from the *Workshop of Machine Translation 2010* (WMT'10) [35]. Statistics from this corpus are shown in Tables 1 and 4. These numbers were computed after cleaning, tokenization and lowercase preprocessing. The tokenization step was carried out by using the script `tokenizer.perl` from the WMT'10. The English vocabulary was extracted from the 80.9K sentences with lengths up to 40 words. 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.

4.2 Baseline translation system

The baseline translation experiments follow the well established phrase-based statistical machine translation approach [40] where the atomic units to be translated are whole sequences of words. It has been trained by using the open-source machine translation toolkit Moses [41] in its standard setup and using the configuration by default.

Table 4. Statistics of the bilingual Spanish-English task of the News-Commentary 2010 corpus.

Corpora	**Spanish**		**English**	
	# lines	**# words**	**# lines**	**# words**
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

https://doi.org/10.1371/journal.pone.0200884.t004

Table 5. Baseline results using Moses and our decoder for the News2010 test set. Results using a count-based 4-gram as the target language model. Time is the average decoding time (in seconds) per sentence measured on an Intel(R) Core(TM) i5 CPU 750 @2.67GHz CPU with 16GB RAM.

System	BLEU	TER	Time (s/sentence)
Moses	22.6	57.8	0.6
Our system	22.7	57.8	0.4

https://doi.org/10.1371/journal.pone.0200884.t005

The word alignments to extract phrases have been obtained using Giza++ [42] with the heuristic `grow-diag-final-and`.

Besides the phrase table, phrase-based translation requires a language model of the target language. To this end, statistical *N*-gram LMs were trained with the SRI toolkit [39].

The overall trained system is composed by 14 models which are combined following the maximum entropy approach:

- Seven reordering models corresponding to the `msd-bidirectional-fe`.
- Language model of the target language.
- Four translation models: inverse phrase translation probability, inverse lexical weighting, direct phrase translation probability, direct lexical weighting.
- Number of phrases penalty.
- Number of words penalty.

All systems were optimized using the MERT procedure [43] on the News2008 set. A detailed description of the translation system can be found in [44].

Table 5 shows the obtained baseline performance for the News2010 test set, along with the average time to decode each sentence, using Moses and the April-ANN toolkit (https://github.com/april-org/april-ann). All the numbers are computed over lower-cased and tokenized sentences.

4.3 Translation experiments

The LMs of the target language were trained, for the baseline system, with the SRI toolkit [39]. In this section, the same NNLM and Fallback V-NNLM connectionist LMs described in Section 3, trained using the April-ANN toolkit, have been used to rerank a list of *n*-best hypotheses (in this experiment, $n = 1000$) generated by our system. Table 6 shows the BLEU and TER of the translations given by the systems, after reranking, along with the decoding times, computed as the average seconds per sentence of the translation plus the rescoring step times. All the numbers are computed over lower-cased and tokenized sentences.

Table 6. BLEU and TER for the News2010 test set for NNLM and Fallback V-NNLM models, for different *N*-gram orders. Time is computed as the average seconds per sentence of the translation plus the rescoring step times.

	BLEU		TER		Time (sec./sentence)	
N-gram order	NNLM	F.V-NNLM	NNLM	F.V-NNLM	NNLM	F.V-NNLM
2	22.9	22.9	57.5	57.5	0.5	0.5
3	23.3	23.1	57.3	57.4	0.7	0.5
4	23.4	23.2	57.2	57.4	1.5	0.6
5	23.2	23.1	57.3	57.4	2.6	0.6

https://doi.org/10.1371/journal.pone.0200884.t006

In the case of NNLMs, we have taken advantage of the tables of normalization constants precomputed for the Fallback V-NNLM model. Note that this is a gain over conventional NNLM implementations relating the computational cost. Therefore, when the softmax constant is not found, it is computed exactly and memorized for future use during the decoding procedure for the same sentence, while the Fallback V-NNLM applies the fallback mechanism in that case.

The performance shows negligible differences between the Fallback V-NNLMs and NNLMs in both BLEU and TER. Regarding decoding times, observe that the use of precomputed values for the regular NNLMs leads to identical computations as the Fallback V-NNLMs since the entire vocabulary is precomputed. Otherwise stated, they are equivalent models for bigrams and lead to the very same results. When using trigrams, Fallback V-NNLMs are slightly faster and this difference in time increases as the *N*-gram order grows.

5 Conclusions and future work

This work presents a connectionist language model, called Fallback V-NNLM, that addresses the computational cost, during the evaluation phase, caused by the softmax activation output. Although many techniques have been proposed in the literature to tackle this problem, it cannot be considered a solved issue and, indeed, none of the proposed techniques has been adopted as standard by the community. The approach proposed in this work does not pretend to solve this problem in its generality but, rather, to provide an effective way to address it in some particular, albeit quite relevant, cases.

This model is composed of a unique NN, called V-NNLM, which can emulate lower order NNLMs by varying the context input. This model is used along with a collection of tables of precomputed softmax normalization constants in order to avoid the computation of the entire set of output neurons during decoding. When a normalization constant is not found, a fallback mechanism is followed. The capability of V-NNLMs to emulate lower order NNLMs greatly simplifies the use of this fallback mechanism.

As other techniques based on precomputing some values ahead of time, our approach has scalability problems with the *N*-gram order and with the lexicon size. However, compared with the conversion into standard count-based back-off models [25, 26], we can mention that a single precomputed constant in our approach is able to deal with a number of transitions equal to the lexicon size.

Experimental evidence on a statistical machine translation task conducted on the bilingual Spanish-English part of the News corpus shows a considerable speed-up when using NNLMs at the expense of a negligible loss in performance.

We believe that the proposed Fallback V-NNLM model makes the integration of NNLMs into the decoding stage (instead of rescoring *n*-best lists) much more feasible.

The proposed technique is compatible with many other techniques used both in training and evaluation, such as Noise-Contrastive Estimation [10], the shortlist approach [2, 11] (already used in the experiments), or the use of deep architectures [45], to mention a few.

Acknowledgments

Work partially supported by the Spanish MINECO and FEDER founds under project TIN2017-85854-C4-2-R (to MJCB). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Author Contributions

Investigation: Francisco J. Zamora-Martínez, Salvador España-Boquera, Maria Jose Castro-Bleda, Adrian Palacios-Corella.

Software: Francisco J. Zamora-Martínez, Salvador España-Boquera, Maria Jose Castro-Bleda, Adrian Palacios-Corella.

Writing – original draft: Francisco J. Zamora-Martínez, Salvador España-Boquera, Maria Jose Castro-Bleda, Adrian Palacios-Corella.

Writing – review & editing: Francisco J. Zamora-Martínez, Salvador España-Boquera, Maria Jose Castro-Bleda, Adrian Palacios-Corella.

References

1. Bengio Y, Ducharme R, Vincent P, Jauvin C. A Neural Probabilistic Language Model. Journal of Machine Learning Research. 2003; 3(2):1137–1155.
2. Schwenk H. Continuous space language models. Computer Speech and Language. 2007; 21(3): 492–518. http://dx.doi.org/10.1016/j.csl.2006.09.003.
3. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural Language Processing (Almost) from Scratch. J Mach Learn Res. 2011; 12:2493–2537.
4. Mikolov T, Yih WT, Zweig G. Linguistic Regularities in Continuous Space Word Representations. In: Proc. of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). Association for Computational Linguistics; 2013. p. 746–751. Available from: http://research.microsoft.com/apps/pubs/default.aspx?id=189726.
5. Józefowicz R, Vinyals O, Schuster M, Shazeer N, Wu Y. Exploring the Limits of Language Modeling. CoRR. 2016;abs/1602.02410.
6. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. CoRR. 2013;abs/1301.3781.
7. Zamora-Martínez F, Castro-Bleda MJ, España-Boquera S. Fast Evaluation of Connectionist Language Models. In: International Work-Conference on Artificial Neural Networks (IWANN). vol. 5517 of LNCS. Springer; 2009. p. 33–40. Available from: https://link.springer.com/chapter/10.1007/978-3-642-02478-8_5.
8. Zamora-Martínez F, Frinken V, España-Boquera S, Castro-Bleda MJ, Fischer A, Bunke H. Neural network language models for off-line handwriting recognition. Pattern Recognition. 2014; 47(4): 1642–1652. http://dx.doi.org/10.1016/j.patcog.2013.10.020.
9. Bengio Y, Senecal JS. Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model. IEEE Transactions on Neural Networks. 2008; 19(4):713–722. https://doi.org/10.1109/TNN.2007.912312 PMID: 18390314
10. Mnih A, Teh YW. A fast and simple algorithm for training neural probabilistic language models. CoRR. 2012;abs/1206.6426.
11. Schwenk H, Dèchelotte D, Gauvain JL. Continuous space language models for statistical machine translation. In: Proc. of the COLING/ACL; 2006. p. 723–730.
12. Morin F, Bengio Y. Hierarchical probabilistic neural network language model. In: Proc. of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS); 2005. p. 246–252.
13. Son LH, Oparin I, Allauzen A, Gauvain JL, Yvon F. Structured Output Layer Neural Network Language Model. In: Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). vol. 11; 2011. p. 5524–5527.
14. Shi Y, Zhang WQ, Cai M, Liu J. Efficient One-Pass Decoding with NNLM for Speech Recognition. IEEE Signal Processing Letters. 2014; 21:377–381. https://doi.org/10.1109/LSP.2014.2303136
15. Devlin J, Zbib R, Huang Z, Lamar T, Schwartz R, Makhoul J. Fast and robust neural network joint models for statistical machine translation. In: Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics; 2014. p. 1370–1380.
16. Paul B, Phil B, Hieu H. OxLM: A neural language modelling framework for machine translation. The Prague Bulletin of Mathematical Linguistics. 2014; 102(1):81–92. https://doi.org/10.2478/pralin-2014-0016
17. Chen W, Grangier D, Auli M. Strategies for training large vocabulary neural language models. CoRR. 2015;abs/1512.04906.

18. Shrivastava A, Li P. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In: Advances in Neural Information Processing Systems; 2014. p. 2321–2329.

19. Graves A, Fernández S, Gomez F, Schmidhuber J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine learning. ICML'06. New York, NY, USA: ACM; 2006. p. 369–376. Available from: http://doi.acm.org/10.1145/1143844.1143891.

20. Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks. In: International Conference on Machine Learning; 2014. p. 1764–1772.

21. Gulcehre C, Firat O, Xu K, Cho K, Bengio Y. On integrating a language model into neural machine translation. Computer Speech & Language. 2017; 45:137–148. https://doi.org/10.1016/j.csl.2017.01.014.

22. Vijayanarasimhan S, Shlens J, Monga R, Yagnik J. Deep networks with large output spaces. CoRR. 2014;abs/1412.7479.

23. Rastogi P, Van Durme B. Sublinear Partition Estimation. CoRR. 2015;abs/1508.01596.

24. Katz S. Estimation of Probabilities From Sparse Data for the Language Model Component of a Speech Recognizer. IEEE Trans on Acoustics, Speech, and Signal Processing. 1987; 34(3):400–401. https://doi.org/10.1109/TASSP.1987.1165125

25. Arisoy E, Chen SF, Ramabhadran B, Sethy A. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. In: Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2013. p. 8242–8246.

26. Wang R, Utiyama M, Goto I, Sumita E, Zhao H, Lu BL. Converting Continuous-Space Language Models into N-gram Language Models for Statistical Machine Translation. In: Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP); 2013. p. 845–850.

27. Huang X, Alleva F, Hon HW, Hwang MY, Lee KF, Rosenfeld R. The SPHINX-II speech recognition system: an overview. Computer Speech and Language. 1993; 7(2):137–148. https://doi.org/10.1006/csla.1993.1007

28. Rosenfeld R. A maximum entropy approach to adaptive statistical language modelling. Computer Speech & Language. 1996; 10(3):187–228. http://dx.doi.org/10.1006/csla.1996.0011.

29. Ney H, Essen U, Kneser R. On structuring probabilistic dependences in stochastic language modeling. Computer Speech and Language. 1994; 8(4):1–38. https://doi.org/10.1006/csla.1994.1001

30. Martin S, Hamacher C, Liermann J, Wessel F, Ney H. Assessment of smoothing methods and complex stochastic language modeling. In: European Conference on Speech Communications and Technology (Eurospeech). vol. 5. Budapest, Hungary; 1999. p. 1939–1942.

31. Siu M, Ostendorf M. Variable n-grams and extensions for conversational speech language modeling. IEEE Trans Speech and Audio Processsing. 2000; 8(1):63–75. https://doi.org/10.1109/89.817454

32. Goodman JT. A Bit of Progress in Language Modeling—Extended Version. One Microsoft Way Redmond, WA 98052: Microsoft Research; 2001. MSR-TR-2001-72.

33. Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network. CoRR. 2015;abs/1503.02531.

34. Callison-Burch C, Koehn P, Monz C, Schroeder J, Fordyce CS, editors. Proc. of the Third Workshop on Statistical Machine Translation. Columbus, Ohio: Association for Computational Linguistics; 2008. Available from: http://www.aclweb.org/anthology/W/W08/W08-03.

35. Callison-Burch C, Koehn P, Monz C, Peterson K, Zaidan O, editors. Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR. Uppsala, Sweden: Association for Computational Linguistics; 2010. Available from: http://www.aclweb.org/anthology/W10-17.

36. Zamora-Martínez FJ. Aportaciones al modelado conexionista de lenguaje y su aplicación al reconocimiento de secuencias y traducción automática, PhD Thesis, Universitat Politècnica de València; 2012.

37. Schwenk H, Koehn P. Large and diverse language models for statistical machine translation. In: Proc. of the International Joint Conference on Natural Language Processing (IJCNLP); 2008. p. 661–666.

38. Palacios-Corella A. Adaptación de skip-gramas a modelos conexionistas del lenguaje; 2015. Master Thesis. Universitat Politècnica de València.

39. Stolcke A. SRILM: an extensible language modeling toolkit. In: Proc. of the International Conference on Spoken Language Processing (ICSLP); 2002. p. 901–904.

40. Koehn P, Och FJ, Marcu D. Statistical Phrase-based Translation. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology—Volume 1. NAACL'03. Stroudsburg, PA, USA: Association for Computational Linguistics; 2003. p. 48–54.

41. Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, et al. MOSES: open-source toolkit for statistical machine translation. In: Proc. of ACL; 2007. p. 177–180.

42. Och FJ, Ney H. A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics. 2003; 29(1):19–51. https://doi.org/10.1162/089120103321337421

43. Och FJ. Minimum Error Rate Training in Statistical Machine Translation. In: Proc. of the ACL. Sapporo, Japan; 2003. p. 160–167.

44. Zamora-Martinez F, Castro-Bleda MJ. Efficient Embedded Decoding of Neural Network Language Models in a Machine Translation System. International Journal of Neural Systems. 2018. https://doi.org/10.1142/S0129065718500077 PMID: 29631501

45. Arisoy E, Sainath TN, Kingsbury B, Ramabhadran B. Deep Neural Network Language Models. In: Proc. of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT. Montréal, Canada: Association for Computational Linguistics; 2012. p. 20–28. Available from: http://www.aclweb.org/anthology/W12-2703.