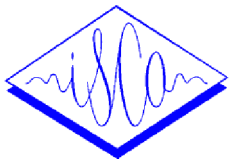


N-gram-based Machine Translation enhanced with Neural Networks for the French-English BTEC-IWSLT'10 task

Francisco Zamora-Martínez[1], *María José Castro-Bleda*[2], *Holger Schwenk*[3]

[1]Departamento de Ciencias Físicas, Matemáticas y de la Computación
Universidad CEU-Cardenal Herrera, Alfara del Patriarca (Valencia), Spain
fzamora@dsic.upv.es
[2]Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Spain
mcastro@dsic.upv.es
[3]Laboratoire d'Informatique d'Université du Maine
Université du Le Mans
FirstName.LastName@lium.univ-lemans.fr

Abstract

Neural Network Language Models (NNLMs) have been applied to Statistical Machine Translation (SMT) outperforming the translation quality. N-best list rescoring is the most popular approach to deal with the computational problems that appear when using huge NNLMs. But the question of "how much improvement could be achieved in a coupled system" remains unanswered. This open question motivated some previous work of us in order to speed the evaluation of NNLMs. Now, this work integrates the NNLM evaluation in the core of the SMT decoder. NNLMs are used in combination with statistical standard N-gram language models under the maximum entropy framework in an N-gram-based SMT system. A reordering decoder builds a reordering graph coupled during a Viterbi decoding.

This N-gram-based SMT system enhanced with NNLMs for the French-English BTEC task of the IWSLT'10 evaluation campaign is described in detail. An improvement between 1.8 and 2.4 BLEU points was obtained from the baseline system to the official primary system. This system has been positioned as second in the automatic evaluation of the IWSLT'10 official results.

1. Introduction

The goal of Statistical Machine Translation (SMT) is the translation of a sentence $\mathbf{f} = f_1 f_2 \dots f_{|\mathbf{f}|}$ from a given source language and source vocabulary $f_i \in \Sigma$, to an equivalent $\hat{\mathbf{e}} = e_1 e_2 \dots e_{|\mathbf{e}|}$ from a certain target language and target vocabulary $e_i \in \Gamma$. Typically this statement is formalised by means of the so-called maximum entropy approach, under a log-linear combination of several models [1, 2]:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{f}, \mathbf{e}), \qquad (1)$$

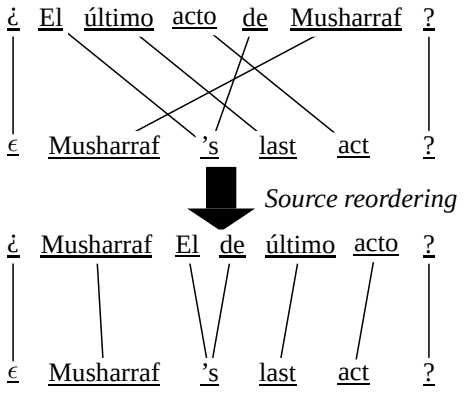
where $h_m(\mathbf{f}, \mathbf{e})$ is a score function representing an important feature for the translation of $\mathbf{f}$ into $\mathbf{e}$, M is the number of models (or features), and λ_m are the weights of the log-linear combination. Typically, the weights λ_m are optimised during the tuning stage. Under the N-gram-based SMT approach, typically two of the combined models are N-gram Language Models (LMs). The traditional N-gram LM is estimated by counting over a text corpus, and it needs the use of smoothing techniques for unseen patterns in the training material.

Continuous space representation of language deals better with unseen patterns, and it has been successfully applied in recent Neural Networks (NNs) approaches to language modelling [3, 4, 5, 6]. However, the use of Neural Network Language Models (NNLMs) [7, 8] in state-of-the-art SMT systems is not so popular. The only comprehensive works are based in [9], where the target LM is presented in the form of a fully-connected Multilayer Perceptron in an N-best list rescoring decoupled step.

The presented system enhances a standard state-of-the-art N-gram-based SMT system with NNLMs via log-linear combination fully integrated in the core of the search procedure of the system.

2. N-gram-based Machine Translation

The phrase-based translation approach is the most extended state-of-the-art SMT solution, and Moses [10] implementation is the most extended decoder. Recently the N-gram-based SMT approach has been presented [11, 12, 13], based on the finite state machine translation framework. The phrases are substituted by bilingual tuples, and the training material is segmented in tuples in an unique way. A bilingual N-gram language model is trained over those tuples, computing the joint probability $p(\mathbf{e}, \mathbf{f})$, approximated at the sentence level. This N-gram translation model is added to the



Extracted tuples		
(¿	,	ϵ)
(Musharraf	,	Musharraf)
(El de	,	's)
(último	,	last)
(acto	,	act)
(?	,	?)

Figure 1: Word alignment between two sentences of the bilingual corpus. The set of extracted tuples is shown below (ϵ represents the empty string).

log-linear combination, in addition with an N-gram language model for the target language, a word and tuple bonus models, lexicon direct and inverse translation models, a "weak" distortion model, and lexicalized reordering models.

2.1. Extracting tuples

A unique segmentation in tuples is extracted from a word alignment of the bilingual corpus following these steps [13]:

1. The source language part of each sentence is reordered to take into account the target word order.
2. Tuples are extracted as the smallest bilingual unit that is possible to define so that no word inside a tuple is aligned with a word outside the same tuple.
3. The source part of each tuple is reordered in increasing order, that is, the words are monotonous inside the tuple, but there could be discontinuities.

This definition builds a segmentation of the reoredered source sentence and the target sentence, where there could be tuples with zero or more words in the source or in the target parts. Zero words in the source part means that the search procedure will need to insert tuples. This behaviour could be hard to implement in the decoder. Then, for simplicity, the target part of empty source tuples is added to the next or the previous tuple maximizing the probability of the lexicon direct and inverse models as described in section 2.2.3.

An example of tuple segmentation is presented in Figure 1. The tuple `(El de, 's)` is an example of discontinuous tuple. This kind of tuple needs a decoding search that takes into account the source words reordering and the discontinuities inside the tuples.

2.2. Modelling

To take into account the implemented constrained reordering search and the tuple-based approach, a generalization of the maximum entropy formula of Equation (1) is:

$$(\hat{\mathbf{T}}, \hat{\varphi}) = \underset{(\mathbf{T},\varphi)}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{T}, \varphi), \qquad (2)$$

where $\mathbf{T} = T_1 T_2 \dots T_{|\mathbf{T}|}$ is a output tuple sequence with each tuple $T_i = (x_i, y_i) \in \Delta$, 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

$$\varphi : \{1, 2, \dots, |\mathbf{f}|\} \rightarrow \{1, 2, \dots, |\mathbf{T}|\}$$

associates a tuple index to 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.

2.2.1. N-gram translation model

The N-gram translation model [13] 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 [12] (Grammar Inference and Alignments for Transducer Inference) technique:

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

where N is the order of the N-gram translation model.

2.2.2. N-gram target language model

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

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

where N is the order of the N-gram target language model.

2.2.3. Lexicon direct and inverse translation models

The lexicon models are computed for the alignment of each source word with each target word inside a tuple, based on the IBM-1 models [14] computation, as:

$$
\begin{aligned}
h_{f2e}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} h'_{f2e}(T_i) \\
h'_{f2e}(x,y) &= \log \frac{1}{(|x|+1)^{|y|}} \prod_{j=1}^{|y|} \sum_{i=0}^{|x|} q(y_j|x_i) \\
h_{e2f}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} h'_{e2f}(T_i) \\
h'_{e2f}(x,y) &= \log \frac{1}{(|y|+1)^{|x|}} \prod_{i=1}^{|x|} \sum_{j=0}^{|y|} q(x_i|y_j)
\end{aligned}
$$

where $q(y_j|x_i)$ and $q(x_i|y_j)$ are the direct and inverse probabilities for the alignment of words x_i and y_j.

2.2.4. "Weak" distortion model

The "weak" distortion model computes a penalization of the reordering of the output hypothesis:

$$
h_d(\mathbf{T},\varphi) = \sum_{i=1}^{|\mathbf{T}|} \Big| last(i-1) + 1 - first(i) \Big|
$$

where $last(i-1) = \max_j \Big\{ j \;\Big|\; \varphi(j) = i-1 \Big\}$ is the position in the input sentence of the last source word in the tuple T_{i-1}, and $first(i) = \min_j \Big\{ j \;\Big|\; \varphi(j) = i \Big\}$ is the position in the input sentence of the first source word in the tuple T_i, being $last(0) = first(0) = 0$.

2.2.5. Lexicalized reordering model

The effect of the addition of this model was evaluated in the experimentation section. Six different models were trained, based on the Moses [10] lexicalized reordering model. Given φ, the current tuple i and the previous tuple $i-1$, the orientation o_d between current tuple and previous tuple is:

$$
o_d = \begin{cases} M, & \text{iif } first(i) - last(i-1) = 1, \\ S, & \text{iif } first(i) - last(i-1) = -1, \\ D, & \text{iif } |first(i) - last(i-1)| \neq 1, \end{cases}
$$

where M, S, and D indicate Monotonous, Swap, and Discontinuous orientation. The probability of that orientation being $t = T_i$ is defined as:

$$
p_d(t|o_d) = (1-\beta)\frac{Count(o_d,t)}{\sum_{o'_d} Count(o'_d,t)} + \beta \frac{Count(o_d)}{\sum_{o'_d} Count(o'_d)},
$$

where $\beta \in [0,1]$ is a smoothing weight, and $\frac{Count(o_d)}{\sum_{o'_d} Count(o'_d)}$ a smoothing factor ("a priori" distribution of o_d).

It is possible to compute the same probability of the inverse orientation (previous tuple respect to the current tuple) as $p_i(t|o_i)$, defining o_i as the orientation o_d, and changing $first(i)$ with $first(i-1)$, and $last(i-1)$ with $last(i)$, and being $t = T_{i-1}$. The three possible orientations, and the two possible directions, define the six lexicalized reordering models:

$$
\begin{aligned}
h_{R1}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} \log p_d(T_i|o_d = M) \\
h_{R2}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} \log p_d(T_i|o_d = S) \\
h_{R3}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} \log p_d(T_i|o_d = D) \\
h_{R4}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} \log p_i(T_{i-1}|o_i = M) \\
h_{R5}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} \log p_i(T_{i-1}|o_i = S) \\
h_{R6}(\mathbf{T},\varphi) &= \sum_{i=1}^{|\mathbf{T}|} \log p_i(T_{i-1}|o_i = D)
\end{aligned}
$$

When the orientation between the tuples is not the corresponding, the value of the probability is 1.

2.2.6. Word and tuple bonus models

These two models penalize the number of inserted words (WIP) and inserted tuples (TIP) generated by the system as follows:

$$
\begin{aligned}
h_{wip}(\mathbf{T}) &= |\mathbf{e}| \\
h_{tip}(\mathbf{T}) &= |\mathbf{T}|
\end{aligned}
$$

where $\mathbf{e}$ is the target language sequence of words corresponding to the sequence of tuples $\mathbf{T}$.

2.3. Decoding

The decoder has been implemented with the April toolkit [15], developed in our research team for pattern recognition and image processing tasks. The search procedure was divided in three coupled steps:

- The source word constrained reordering graph was generated in a topological order, following local constraints (a node is expanded covering every source word position where the distance between the first uncovered position and the last covered position is less than a certain given value). This approach is inspired

by those presented in [16, 17, 18]. Our system scores the reordering hypothesis in the core of the SMT system. In this step it is possible to apply histogram pruning, using an approximation of the future cost of each reordering hypothesis based in Moses future cost computation.

- This graph is extended by forming tuples: every sequence of source words is substituted with an edge between the first and last node of the sequence, tagged with all the possible tuples that were found given the source words. Source words order is restricted to be in increasing order inside a tuple. The graph is generated in a topological order. In this step the lexicon models, and word and tuple bonus models are added to the graph edges. This step is based in the GIATI [12] technique.

- A Viterbi decoding with beam search and histogram pruning, using the N-gram target language model(s), N-gram bilingual translation language model(s), and the reordering models, is performed. This step outputs the 1-best, an N-best list, or a word graph. GIATI technique allows to use the N-gram bilingual translation language model as a stochastic finite state transducer that generate the output sentence $\mathbf{e}$ given the best sequence of tuples $\mathbf{T}$.

The decoder could be extended with NNLMs during the Viterbi decoding, or in a N-best list rescoring decoupled stage. In this work, we explore the first approach.

3. Neural Network Language Models

Under a statistical framework, a language model is used to assign to every possible word sequence $\mathbf{s}$ an estimation of the a priori probability of being the correct system response $p(\mathbf{s}) = p(s_1 \dots s_{|\mathbf{s}|})$. Statistical language models are usually based on the prediction of each linguistic unit in the sequence given the preceding ones [19]:

$$p(\mathbf{s}) = \prod_{i=1}^{|\mathbf{s}|} p(s_i | s_1^{i-1}) \,, \tag{3}$$

where $s_1^{i-1} = s_1 \dots s_{i-1}$ denotes the history from which unit s_i has to be predicted. The number of parameters to estimate becomes intractable as the length of the sentence increases. N-gram models are the most extended method to reduce this number approximating the probability of a word as if only the last $N-1$ words have influence:

$$p(\mathbf{s}) \approx \prod_{i=1}^{|\mathbf{s}|} p(s_i | s_{i-N+1}^{i-1}) \,. \tag{4}$$

A Neural Network Language Model (NNLM) is a statistical LM which follows the same Equation (4) as N-grams and where the probabilities that appear in that expression are estimated with a NN [3, 5, 7]. The model naturally fits under the probabilistic interpretation of the outputs of the NNs: if a NN, in this case a Multilayer Perceptron (MLP), is trained as a classifier, the outputs associated to each class are estimations of the posterior probabilities of the defined classes [20].

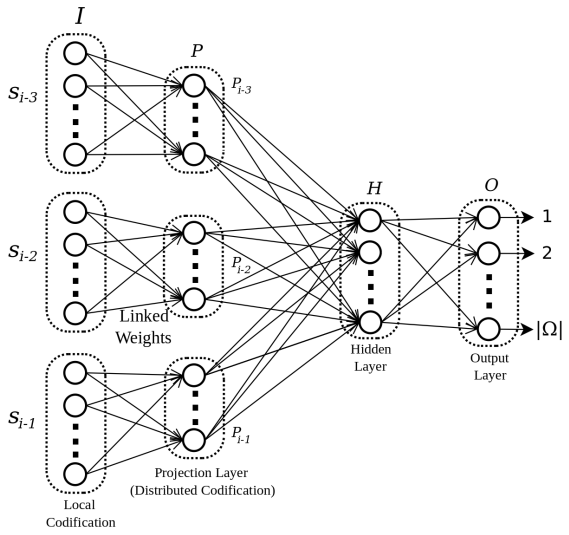


Figure 2: Architecture of the continuous space NNLM in the training stage. The input words are $s_{i-N+1}, \dots, s_{i-1}$ (in this example, the input words are s_{i-3}, s_{i-2}, and s_{i-1} for a 4-gram). I, P, H and O are the input, projection, hidden and output layer, respectively, of the MLP.

The training set for a LM is a sequence $s_1 s_2 \dots s_{|\mathbf{s}|}$ of words from a vocabulary Ω. In order to train a NN to predict the next word given a history of length $N-1$, each input word must be encoded. A natural representation is a local encoding following a "1-of-$|\Omega|$" scheme. The problem of this encoding for tasks with large vocabularies (as is typically the case) is the huge size of the resulting NN. We have solved this problem following the ideas of [3, 7], learning a distributed representation for each word. Figure 2 illustrates the architecture of the feed-forward NN used to estimate the NNLM:

- The input is composed of words $s_{i-N+1}, \dots, s_{i-1}$ of Equation (4). Each word is represented using a local encoding.

- P is the projection layer of the input words, formed by $P_{i-N+1}, \dots, P_{i-1}$ subsets of projection units. The subset of projection units P_j represents the distributed encoding of input word s_j. The weights of this projection layer are linked, that is, the weights from each local encoding of input word s_j to the corresponding subset of projection units P_j are the same for all input words j. Tipically the linear activation function is used for this layer. After training, the codification layer is removed from the network by pre-computing a table of size $|\Omega|$ which serves as a distributed encoding.

- H denotes the hidden layer, with the *hyperbolic tangent* as activation function.

- The output layer O has $|\Omega|$ units, one for each word of the vocabulary, with *softmax* as activation function.

This NN predicts the posterior probability of each word of the vocabulary given the history, i.e., $p(s_i|s_{i-N+1}\dots s_{i-1})$ is computed by feeding the encoded input words $s_{i-N+1},\dots,s_{i-1}$ to the input layer. A single forward pass of the MLP gives $p(\omega|s_{i-N+1}\dots s_{i-1})$ for every word $\omega \in \Omega$.

The advantages of the connectionist approach to language modeling are due to their automatic estimation (as with statistical LM), the lowest (in general) number of parameters of the obtained models and the automatic smoothing performed by the neural networks estimators. Estimation of NNLM in ambitious tasks is needed in order to fulfill these advantages. This is not trivial: the larger the lexicon is, the larger the number of parameters the neural network needs. Using a distributed representation for each word of the vocabulary is a successful approach for tasks with large lexica, but the high computational cost of using NNLMs remains a problem.

3.1. Fast evaluation of NNLMs

In order to compute the language model probability of a given sentence, the number of N-gram conditional probabilities to be computed is roughly the length (in words) of the sentence. However, when the same language model is used in speech or handwritten recognition or in translation tasks, the number of language model look-ups is typically huge since there are many different hypotheses to be considered.

The activation function of the hidden neurons is usually the logistic or the hyperbolic tangent. The output units can use the logistic or the softmax activation function whose values lie between zero and one. The softmax activation function [20] also ensures that the output values sum to one:

$$o_i = \frac{exp(a_i)}{\sum_{j=1,\dots,|\Omega|} exp(a_j)}, \tag{5}$$

being a_i the activation value of the i-th output unit and o_i its output value.

The softmax gives much better results in term of perplexity than the logistic function, but requires the computation of every output value, even though only some of them are used, due to the normalization term of Equation (5). The computation of the output layer dominates the cost of the forward pass in a typical NNLM topology. This problem has been noticed before in the literature and two solutions were proposed to this regard [7]:

- To decrease the size of the output layer by removing less frequent words. This sort list approach requires smoothing language modeling techniques to estimate the conditional probability of rare words.

- To collect and to group together all the N-grams which share a common $N-1$ prefix, since all of them require the same forward pass. Note that this can also be useful when other activation functions are used because the number of hidden layers computations is anyway reduced. Unfortunately, this is not always possible in some on-line systems.

Our approach consists on pre-computing and storing the softmax normalization constants most probably needed during the LM evaluation, since the cost of retrieving this value from a table is negligible compared with the cost of computing it [21]. A space/time trade-off has to be considered: the more space is dedicated to store pre-computed softmax normalization constants, the more time reduction can be obtained. When a given normalization constant is not found, it can be computed on-the-fly or some kind of smoothing must be applied. We have followed the latter idea: when a softmax normalization constant is not found, another simpler model (for instance, a lower order NNLM or statistical N-gram model) is used.[1]

The implementation of NNLMs has been performed with our pattern recognition toolkit April [15].

3.2. Unknown words probability estimation

In order to estimate the probability of *unknown words*, the NNLMs needs to be trained over a restricted vocabulary $\Omega' \subset \Omega$ composed by the most frequent words in the training corpora. This restricted vocabulary speeds up the training and evaluation steps reducing the size of the neural network. Then, every word in Ω which does not belong to Ω' is considered as an unknown word adding to the input and output of the NNLM a new unit that represents this class. Using the following function $\mathcal{G}: \Omega \to \Omega'$:

$$\mathcal{G}(s_i) = \begin{cases} s_i, & \text{iif } s_i \in \Omega', \\ s_{unk}, & \text{iif } s_i \notin \Omega'. \end{cases}$$

and asuming that all words in $\Omega - \Omega'$ are equally probable:

$$p(s_i|s_{i-N+1}^{i-1}) = \begin{cases} p_{NN}(s_i|\mathcal{G}'(s_{i-N+1}^{i-1})), & \text{iif } s_i \in \Omega', \\ \dfrac{p_{NN}(s_{unk}|\mathcal{G}'(s_{i-N+1}^{i-1}))}{|\Omega| - |\Omega'|}, & \text{iif } s_i \notin \Omega', \end{cases} \tag{6}$$

where p_{NN} is the conditional probability function computed by the NNLM and the $\mathcal{G}'$ function was extended over sequences of words.

4. Experimentation

4.1. Corpus

Table 1 summarizes the size of the French-English BTEC-IWSLT'10 task corpus in lowercase and tokenized form, but preserving the punctuation marks. The tokenization

[1] Note that storing the softmax normalization constants for a bigram NNLM only needs a $|\Omega|$-sized table.

Table 1: Statistics of the French-English IWSLT'10 bilingual corpora statistics in tokenized and lowercase form. Development sets figures are calculated as the concatenation of the available 16 multiple references. The size of the N-gram bilingual corpus is also shown.

	French			English		
	# lines	# words	# vocabulary	# lines	# words	# vocabulary
BTEC + CSTAR'03 (Train)	28K	273K	**9 954**	28K	256K	7 599
IWSLT'04 (Dev2)	8K	72K	3 281	8K	67K	2 191
IWSLT'05 (Dev3)	8K	72K	3 144	8K	68K	2 271
Total	44K	417K	11 389	44K	392K	**8 394**

	N-gram bilingual translation corpus		
	# lines	# tuples	# vocabulary
BTEC + CSTAR'03 (Train)	28K	214K	41 406

was done using the script `tokenizer.perl` from the WMT'10. The French vocabulary was extracted from the Train partition (9 954 words), and the English vocabulary from the concatenation of all available data (8 394 words). The Dev2 partition was reserved for tuning parameters of all the models. Dev3 was a internal test set for selecting the best system. All the models were trained over the lowercased and tokenized Train partition.

4.2. N-gram-based SMT baseline system

The training material for the baseline system is the 20K sentences from the BTEC training partition and the 16 references of the CSTAR'03 development set (Train set). A 3-gram bilingual translation model was trained over the tuple segmentation of the corpus obtained from the Gizza++ [22] words alignments (using the heuristic `grow-diag-final-and`). The lexicalized reordering model was trained over the same material. A 3-gram target language, result of the combination of two language models, for BTEC and for CSTAR'03 English corpora, was trained. For language model combination, the perplexity over the Dev2 development set was optimized. The result with baseline system for Dev2 development set was performed with and without lexicalized reordering model (see Table 2). The log-linear combination coefficients λ_m where optimized over the Dev2 development set, maximizing the BLEU score by means of the MERT [23] procedure. The local reordering window was set to 6. The baseline system combines from 7 to 13 models in the log-linear search, depending of the use or not of the lexicalized reordering models. The extension of the baseline with NNLMs has 1 or 2 more models, depending if the NNLM was only used for the target language model, only for the bilingual translation model, or for both. After all experimentation, the final primary submitted system is composed of 15 models.

4.3. NNLMs set up

All the NNLMs were trained using the Stochastic Backpropagation algorithm, with replacement and weight decay, minimizing the cross-entropy error function.

A 3-gram NNLM to be used as translation model, Neural Network Translation Model (NNTM), was trained over the tuple segmentation of the training material. The training data was randomly partitioned into two disjoint sets, a training set with 24K sentences, and a validation set with 4K sentences. The NNTM is a linear combination of 4 neural networks with $P_j = 128, 160, 192, 256$ and $H = 200$, respectively. The restricted vocabulary was set to the 9K most frequent tuples from the $|\Delta| = 41$K tuples.

A 4-gram NNLM to be used as LM for the target language, Neural Network Target Language Model (NNTLM), was trained over the English part of the training material. The Dev2 development set was used as a validation set. The NNTLM is a linear combination of 4 neural networks with $P_j = 128, 160, 192, 256$ and $H = 200$, respectively. The restricted vocabulary was set to the 5K more frequent words from the vocabulary of $|\Gamma| = 8$K words.

For these two NNLMs, the following extension of p_{NN} in Equation (6) was used:

$$p_{NN}(s_i|s_{i-N+1}^{i-1}) = \sum_{j=1}^{4} \alpha_j \cdot p_{NNj}(s_i|s_{i-N+1}^{i-1}), \quad (7)$$

where p_{NNj} is the conditional probability function of the j NN, and α_j its corresponding linear combination coefficient, under the restrictions: $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$, $\alpha_i \in [0, 1]$. The coefficients were optimized by means of an Expectation-Maximization procedure to minimize the corresponding validation set perplexity. The Mert procedure was repeated from the begining to tune the log-linear combination coefficients, adding the new features to the baseline log-linear combination (keeping the standard N-gram language models scores).

4.4. Results

Every result is shown in `case+punc` form. A standard Moses recasing model, and the script `detokenizer.perl` from the WMT'10 were used. Table 2 summarizes the performed experiments for tuning the system. Three different factors were evaluated:

- **Effect of the lexicalized reordering model**. For the baseline system, the use of this reordering model only gives a small improvement, 0.2 points of BLEU, the same when the NNLMs are used in the rescoring step. Nevertheless, when the NNLMs are integrated in the decoder search, the improvement is larger, 0.8 points of BLEU. This means that the integration of more data in the decoding stage is positive.
- **Effect of NNTLM and NNTM**. The addition of NNTLM to the log-linear combination achieves an improvement of 0.5 points of BLEU over our baseline. The addition of NNTM achieves an improvement of 0.2 points. Nevertheless, the addition of both models, NNTLM and NNTM, achieves an improvement of 1.1 points of BLEU. When the reordering model is also added to the combination, this number goes up to 1.7 points of BLEU.
- **Effect of NNLMs integrated in decoding vs. rescoring**. The results nearly are the same for both scenarios when the reordering models are not in the log-linear combination. But when the reordering models are used, the integrated system achieves an improvement of 0.5 points of BLEU over the rescoring system. This could be explained because more models need a better optimization in the MERT procedure. Comparing the final integrated system with our baseline system, 1.9 points of BLEU improvement were achieved.

4.5. Official results

Table 3 summarizes the official BLEU score obtained for the baseline system and the best system. For this table two different experimental frameworks were tested. The first set uses the same systems than those in Table 2. The second set of experiments uses the systems trained over all the available data (BTEC training + 4 development sets with 16 multiple references). The log-linear combination scores were the same in the two sets, but the standard N-gram target LM was obtained as the result of a combination of single N-gram target LMs for each of the 4 available sets. The combination coefficients were optimized with a cross-validation procedure splitting each set in 10 parts. A stochastic backpropagation algorithm with resampling, and with different resampling coefficient for each set, was used to train the NNTLMs.

From the final results it is possible to observe that there is a clear improvement when the NNLMs are added to the log-linear combination in the integrated system, but it is not clear the advantage of using all the available data for the final submission. The official result of our system is 63.6 and 53.6 for test09 and test10 respectively, 1.8 and 2.4 points of BLEU over the baseline system.

Table 2: BLEU results for the Dev2 (tuning) and Dev3 (internal test set) partitions. The number of processed source words per second was measured for each system. NNTM is the addition of the NN for the N-gram translation model, NNTLM is the addition of the NN for the N-gram target language model, and R is the addition of the lexicalized reordering model.

System	Dev2	Dev3	Wrds/Sec.
N-gram-based	65.8	65.2	21
+ R	65.8	65.4	21
Integrating NNLMs in the decoder			
+ NNTLM	67.0	65.7	8
+ NNTM	66.6	65.4	10
+ NNTLM + NNTM	67.4	66.3	7
+ NNTLM + NNTM + R	67.7	67.1	7
Rescoring 1000-best list			
+ NNTLM + NNTM	66.9	66.4	–
+ NNTLM + NNTM + R	67.8	66.6	–

Table 3: Official BLEU results for the test sets IWSLT'09 (Test1) and IWSLT'10 (Test2) partitions, primary results are bolded. Every result is with the integrated system.

System	Test1	Test2
N-gram-based + R	61.4	52.2
+ NNTLM + NNTM + R	62.5	54.3
Adding all available data		
N-gram-based + R	61.8	51.2
+ NNTLM + NNTM + R	**63.6**	**53.6**

5. Conclusions

An N-gram-based SMT system enhanced with NNLMs for the French-English BTEC task of the IWSLT'10 evaluation campaign was presented. An improvement between 1.8 and 2.4 BLEU points was obtained between the baseline system and the official primary system. The performance in time of the NNLMs integrated system only decreased 3 times over the baseline, due to the NNLMs are integrated in the decoder search using a novel technique [21]. This integration in the decoding stage achieves an improvement of 0.5 points of BLEU over a rescoring of 1000-best list in the experimentation presented. Even the big improvement obtained over the BLEU score, the use of all the available data was not clearly a good idea for the IWSLT task, may be due to overtraining. This system has been positioned as second in the automatic evaluation of the IWSLT'10 official results.

6. Acknowledgements

Thanks to Patrick Lambert for the tokenization configuration file for French, and to the LIUM SMT team for their valuable discussions during the 2010 summer. Work partially supported by the Spanish *Ministerio de Ciencia e Innovación* (TIN2010-18958).

7. References

[1] K. Papineni, S. Roukos, and T. Ward, "Maximum likelihood and discriminative training of direct translation models," in *Proc. of ICASSP*, 1998, pp. 189–192.

[2] F. Och and H. Ney, "Discriminative training and maximum entropy models for statistical machine translation," in *Proc. of ACL*, 2002, pp. 295–302.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model," *Journal of Machine Learning Research*, vol. 3, no. 2, pp. 1137–1155, 2003.

[4] H. Schwenk and J.-L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Proc. of ICASSP*, 2002, pp. 765–768.

[5] M. Castro-Bleda and F. Prat, "New Directions in Connectionist Language Modeling," in *Computational Methods in Neural Modeling*, ser. LNCS. Springer-Verlag, 2003, vol. 2686, pp. 598–605.

[6] H. Schwenk, D. Déchelotte, and J.-L. Gauvain, "Continuous space language models for statistical machine translation," in *Proceedings of the COLING/ACL*, 2006, pp. 723–730.

[7] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, 2007.

[8] Y. Bengio, "Neural net language models," *Scholarpedia*, vol. 3, no. 1, p. 3881, 2008.

[9] H. Schwenk, "Continuous space language models for statistical machine translation," *The Prague Bulletin of Mathematical Linguistics*, vol. 93, 2010.

[10] 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," in *Proc. of ACL*, 2007, pp. 177–180.

[11] F. Casacuberta, E. Vidal, and J. M. Vilar, "Architectures for speech-to-speech translation using finite-state models," in *Proc. of the Workshop on Speech-to-Speech Translation: Algorithms and Systems*, 2002, pp. 39–44.

[12] F. Casacuberta and E. Vidal, "Machine translation with inferred stochastic finite-state transducers," *Comput. Linguist.*, vol. 30, pp. 205–225, 2004.

[13] J. B. Mariño, R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-jussà, "N-gram-based machine translation," *Comput. Linguist.*, vol. 32, pp. 527–549, 2006.

[14] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: parameter estimation," *Comput. Linguist.*, vol. 19, no. 2, pp. 263–311, 1993.

[15] S. España-Boquera, F. Zamora-Martínez, M. Castro-Bleda, and J. Gorbe-Moya, "Efficient BP Algorithms for General Feedforward Neural Networks," in *IWINAC'07*, ser. LNCS. Springer, 2007, vol. 4527, pp. 327–336.

[16] C. Tillmann and H. Ney, "Word reordering and a dynamic programming beam search algorithm for statistical machine translation," *Comput. Linguist.*, vol. 29, no. 1, pp. 97–133, 2003.

[17] G. Sanchis and F. Casacuberta, "N-best reordering in statistical machine translation," in *IV Jornadas en Tecnologia del Habla*, Zaragoza, Spain, 2006, pp. 99–104.

[18] M. R. Costa-jussà, J. M. Crego, P. Lambert, M. Khalilov, J. A. R. Fonollosa, J. B. Mari no, and R. E. Banchs, "Ngram-based statistical machine translation enhanced with multiple weighted reordering hypotheses," in *WSMT'07*, 2007, pp. 167–170.

[19] F. Jelinek, *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[20] C. M. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.

[21] F. Zamora-Martínez, M. Castro-Bleda, and S. España-Boquera, "Fast Evaluation of Connectionist Language Models," in *IWANN*, ser. LNCS. Springer, 2009, vol. 5517, pp. 33–40.

[22] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[23] F. Och, "Minimum Error Rate Training in Statistical Machine Translation," in *Proc. of ACL*, Sapporo, Japan, 2003, pp. 160–167.