Improvements on Automatic Word Codification for Connectionist Machine Translation*

Gustavo A. Casañ and **M. Asunción Castaño**[1]

Abstract. Encouragingly accurate translations have recently been obtained using a connectionist translator called RECONTRA (Recurrent Connectionist Translator). In order to deal with tasks of medium or large vocabularies, distributed representations of the lexicons are required in this translator. A simple connectionist model has been recently designed to automatically obtain word distributed representations. In this paper several learning algorithms were used to train this connectionist encoder aiming to improve the translation rates achieved with the corresponding obtained codifications of the vocabularies involved.

1 INTRODUCTION

In comparison with traditional *knowledge-based* Machine Translation (MT) systems, in recent years *example-based* techniques (so called inductive techniques) have led to successful limited-domain applications. In this paradigm, systems are automatically built from training sets of examples which are large enough, resulting in lower development costs. In this direction, *Neural Networks* can be considered an encouraging approach to MT, as the translation schemes presented in [9] and [18] have empirically shown.

Other connectionist translator, called *RECONTRA* (Recurrent Connectionist Translator), has been recently designed [5] to tackle text-to-text limited domain applications. It directly carries out the translation between both the input and the output languages and, at the same time, automatically learns the semantics and syntax implicit in both languages. In this approach the vocabularies involved in the translations can be represented according to (simple and clear) local codifications. However, in order to deal with large vocabularies, local representations would lead to networks with an excessive number of connections to be trained in an admissible time. Consequently, distributed representations (more compact than local codifications) of both source and target vocabularies are required. This problem was initially approached in [5] and [3], where it was shown how certain types of distributed codifications (which were manually generated) could carry out MT tasks with larger vocabularies. Later a simple connectionist model presented in [4] was designed to automatically create adequate and compact distributed codifications for the vocabularies. The sizes of these codifications were automatically obtained by successively pruning the neural model. In this paper several different learning algorithms have been used to train the encoders in order to compare the translation rates achieved with the corresponding extracted codifications of the vocabularies involved.

The rest of the paper is organized as follows: Section 2 describes the architecture of the RECONTRA translator, as well as the procedure used to train it. Section 3 shows the connectionist architecture of the encoders and describes the learning algorithms used to train them. Section 4 presents the task to be approached in the experimentation and Section 5 reports the translation performances obtained. Finally, Section 6 discusses the conclusions of the experimental process.

2 THE RECONTRA TRANSLATOR

The basic neural topology of the RECONTRA translator is a network presented by Elman in [6]. An Elman network is a simple Recurrent Neural Network in which the activations of the preceding step in the hidden units of the networks are fedback as inputs in the hidden layer. The architecture of RECONTRA includes time delays in the input layer of an Elman network, in order to reinforce the information about past and future events. Figure 1 illustrates the resulting connectionist topology.

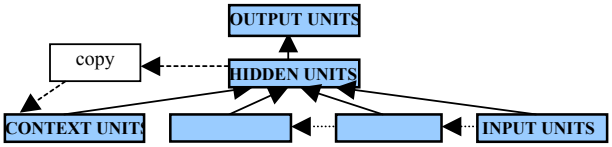

Figure 1. The RECONTRA Translator

In relation to the running of the connectionist architecture, the words of the sentence to be translated are sequentially presented to the input of the network, while the net has to provide the successive words of the corresponding translated sentence.

The RECONTRA translator is trained by using a version of the *Backward-Error Propagation* algorithm [17]. The weights of the net are modified after each input is processed and the corresponding error generated is computed. Consequently an "on-line" algorithm is used (the updating of the weights only after each presentation of the complete learning corpus would lead to a "batch" training algorithm). A sigmoid function (0,1) is assumed as the non-linear activation function and context activations are initialized to 0.5 at the beginning of every input-output pair. The choice of the learning rate term and the

* Partially supported by the Spanish Fundación Bancaja, project P1.1B2002-1.

[1] Dpto. Ingeniería y Ciencia de Computadores Universidad Jaume I. Castellón, Spain. {ncasan,castano}@icc.uji.es

momentum term is carried out inside the unitary bidimensional space which they define, by analyzing the residual mean squared error of a network trained for 10 random presentations of the learning corpus (10 epochs). Training continues for the learning rate and momentum which led to the lowest mean squared error. And the training process stops after a certain number of training epochs is reached.

With regard to the translated message provided by the RECONTRA model, the network continuously generates output activations. In order to interpret the activations provided at a given time cycle, the word associated to the preestablished codification of the target vocabulary which is nearest to such activations is searched for. *Word Error Rate (WER)* is computed by comparing the obtained and expected translations corresponding to every source sentence in the test sample using a conventional Edit-Distance procedure [10]. In this way, the number of insertions, deletions and substitutions errors required to transform a sentence into the other are obtained. The WERs reported here correspond to the ratio of the total number of errors with respect to the total number of edit (total error+non-error) operations.

3 THE CODIFICATIONS GENERATOR

In our MT experimentation, we should represent the words of the vocabularies involved. In order to get distributed lexicons representations for the RECONTRA translator automatically, several neural techniques could be employed [5] [14] [2] [11]. The method adopted in this paper was a simple but effective one presented in [4] which is described immediately after.

The codifications obtained by this encoder are subsymbolic distributed codifications in which the units have continuous-valued activations, in the range [0,1]. A *distributed representation* is defined [8] as one in which each concept or element is represented over several units, and in which each unit participates in the representation of several concepts. When no meaning can be assigned to any particular unit these representations are called *subsymbolic distributed codifications*.

3.1 Network architecture

The encoder used to code the vocabularies of the MT task is a Multilayer Perceptron (MP) trained to produce the same output as the input (a word of the vocabulary). In order to take into account the context in which a word appears, the corresponding previous and following words in a sentence are also shown at the output of the MP, resulting the topology that can be seen in Figure 2. When the MP is trained enough, the activations of the hidden units have developed its own representations of the input/output words and can be considered the codifications of the words in the vocabulary, taking into account the context in which each word appears. Consequently, the size of the (unique) hidden layer of the MP decides the size of the distributed codifications obtained.

To automatically determine an adequate size of these codifications and not being dependent on a human expert, a pruning algorithm is applied on the hidden layer. It consists of removing the units which are not necessary for the representation of the vocabularies. The pruning algorithm used is the *Skeletonization* algorithm [13]. After pruning the network, a word is presented to the MP and the activations of the hidden units are extracted. These activations are assumed to be the representations developed by the MP encoders for the words of the vocabulary.

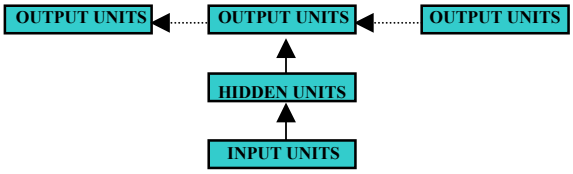

Figure 2. A Multilayer Perceptron with output delays

As the results shown in [4] revealed, when the input word has less importance at the output than its contexts, the codifications of similar part of speech words are too similar to be adequate for translation purposes; that is because the translator can not differentiate between the different nouns or verbs, for instance. Thus, the emphasis in the codification process should be (equated or) put on the input word over its context. This is achieved by repeating the input word several times at the output window of the encoder. According to this a possible output window format of size 4 for the x-th input word, $\mathbf{w}_x$, of a sentence $w_1\ w_2\ w_3$ …could be $w_{x-1}\ \boldsymbol{w_x}\ \boldsymbol{w_x}\ w_{x+1}$, where w_{x-1} is the previous word in the context of this sentence and w_{x+1}, the following word. To simplify the nomenclature, from now on, we will refer to such example as $x\text{-}1\ \boldsymbol{x}\ \boldsymbol{x}\ x\text{+}1$.

3.2 Training procedures

In previous experiments with the MP encoder described in the above subsection [4], it was trained using the classical Backward-Error Propagation algorithm. In order to obtain better codifications which led to higher translation rates with RECONTRA, other learning methods could be adopted for the encoders of the vocabularies. To this end, in this paper three supervised training algorithms were tried:

- *Backward-Error Propagation* (BEP) algorithm [17]: It is the most common learning algorithm to train MPs. This algorithm was also adopted to train the RECONTRA translator. At this time, we also used a momentum term and a learning rate term for updating the values of the connections. The choice of these parameters was carried out inside the bidimensional space which they defined, by analyzing the residual mean squared error of a network trained for 10 random presentations of the complete learning corpus (10 epochs). The (on-line) updating of the weights continued for the learning rate and momentum that led to the lowest mean squared error over the learning corpus.
- *Resilient back PROPagation* (RPROP) [16]: It is a local adaptive batch learning scheme. The basic principle of RPROP is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the

direction of the weight update. Every time the partial derivative of the corresponding weight changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the 'update value' is decreased (in our case using a fixed value). If the derivative retains its sign, the 'update-value' is slightly increased in order to accelerate the convergence in shallow regions.

- *Scalated Conjugate Gradient* (SCG) [12]: It is a conjugate gradient method; that is, a general purpose second order technique that helps minimize goal functions of several variables. Like standard BEP, SCG iteratively tries to get closer to the minimum, but not down the gradient of the error function as BEP. A SCG will proceed in a direction which is conjugate to the directions of the previous steps. Thus the minimization performed in one step is not partially undone by the next, as it is the case with standard BEP and other gradient descent methods.

In order to automatically obtain a possible size of the codifications of the lexicons, the encoder network was pruned using the *Skeletonization* algorithm [13] to remove the least necessary hidden neurons. The pruning mechanism consisted on training the MP for 10 epochs and selecting a hidden unit to be removed. The new pruned MP was trained for 10 more epochs and pruned again. The pruning process was repeated until there were no more hidden units to be removed. After this, the pruned MP which had the minor residual mean squared error was selected for further training. This additional training was required since experimental results (not reported here) showed that the codificactions of the different words of the vocabulary extracted at this time from the pruned encoders were very similar.The training process (usually) stopped after 3,000 epochs and a sigmoid function (0,1) was assumed as the non-linear function.

If the learning algorithms described above were adopted for both training and pruning the MPs, at the same time, the final trained MPs could have different numbers of pruned hidden neurons. Consequently, the results achieved in the translation process could not be exactly compared. Comparisons among translation rates should be made, for instance, after training a given pruned encoder with different learning algorithms.

4 THE MACHINE TRANSLATION TASK

The task chosen in this paper to test the encoders of the vocabularies required for the RECONTRA translator was a subtask of the Spanish-to-English text-to-text Traveller MT task, called the *Mini-Traveller task.* It was designed within the first phase of the EuTrans project [1] and is restricted to a limited semantic domain. The task approaches typical situations of a traveller at the reception of a hotel in a country whose language he/she does not speak. It includes sentences in which the traveller notifies his/her departure, asks for the bill, asks and complains about the bill and asks for his/her luggage to be moved. Some examples of this task are shown in Table 1. It has 178 different Spanish words, and 140 English words. The best WER obtained with other inductive methods was around 2% (see [5] and [15] for more details).

Table 1. Pairs of sentences from the Mini-Traveller task

Spanish:	¿ Está incluido el recibo del teléfono en la factura ?
English:	Is the phone bill included in the bill ?
Spanish:	He de marcharme el día veintisiete de febrero a las siete y media de la tarde .
English:	I should leave on February the twenty-seventh at half past seven in the afternoon .

The corpora adopted in the translation task were sets of text-to-text pairs which consisted of a sentence in the Spanish source language and the corresponding sentence in the English language. They were automatically built by using a set of stochastic syntax-directed translation schemata [7]. A learning set of 5,000 pairs of sentence and a test set of 1,000 sentences were adopted. The average size of the sentences in Spanish was 8.6 and 8 for the English sentences.

The corpora used for the training of the MP encoders were sets of text-to-text pairs, each of them consisting of an input word and the same input word together with its context (the preceding and following words in a sentence) as output. All pairs were extracted from sentences which appeared in the training corpus employed for the translation task. All the repeated pairs extracted from the translation corpus appeared only once in the training set of the MP. There were no test corpora for the codification process; it was indirectly evaluated later in the translation process.

5 EXPERIMENTAL RESULTS

We used MP encoders with different output windows (of 4 up to 8 word delays) with the adequate changes in the number of input and output units due to the different sizes of the English and Spanish vocabularies. The initial number of hidden units at the beginning of the pruning process was set to 25 or 100 neurons. Non-pruned MP with 25 hidden neurons were also used.

With regard to the topology of the RECONTRA, previous experiments on this task [5] suggested to adopt 180 hidden units and an input window of 8 delayed words (with format 3+1+4) for the RECONTRA translators, since they led to adequate translation rates. The number of hidden units of the pruned or non-pruned encoders determined the size of the input and output layer of the RECONTRA translator. The training of these translators was always stopped after 3000 epochs. All these experiments were done using the Stuttgart Neural Network Simulator [19].

5.1 Results training the MPs with the BEP algorithm

First the task was approached using codifications extracted from MPs in which the hidden layer had a pre-established fixed-size of 25 neurons to code each vocabulary. An output window of size 6 and 4 repetitions of the input word at the output was adopted. These encoders were trained using the BEP algorithm.

The learning set of the translation task was codified according to the codifications extracted from the above trained MP

encoders. A RECONTRA translator was trained and evaluated. The test translation eror rate obtained is shown in Table 2. For comparison purposes, the results for an experiment with (binary) manual codifications of the vocabularies and the same topology for RECONTRA are also shown. The sizes of the codifications adopted for the Spanish and English vocabularies (expressed by |Spanish| and |English|, respectively) are displayed, too. As we can see, the translation WER obtained using automatic codifications is slightly higher than that obtained with the (best) manual ones. It should be noted that these hand-made codifications corresponded to coarse representations in which similar codifications were assigned to words that were semantically near. On the other hand, the automatic codifications used in the experiments were more compact holistic representations, which increased the complexity of the translation process.

Table 2. Translation WERs using manual codifications and automatic codifications of fixed size 25 extracted from MPs with output window format x-1 **x x x x** x+1

Codifications		Translation
Type	| Spanish| / | English |	WER
Automatic	25/25	4.26%
Manual	61/52	1.40%

In a second series of experiments, the pruning method was applied to the hidden layer of the encoders. Different output window formats for the MPs (from 4 up to 8 word delays) were tried. MPs initially had 25 hidden neurons and were successively pruned after each 10 epochs. Then, the topology of the pruned MPs which had led to the least error was selected. The resulting encoders were later trained for 3,000 more epochs. Codifications were later extracted for the resulting MPs. The translation process was carried out using these codifications. Table 3 shows the translation WERs achieved on the test set. The sizes obtained for the codifications of the vocabularies are also specified. The results revealed that the WERs were slightly higher than those achieved without pruning the encoders.

Table 3. Translation WERs using automatic pruned codifications from MPs with different features

MP Encoders		Translation
Output window format	| Spanish | / | English |	WER
x-1 **x x** x+1	11/9	9.38%
x-1 **x x x x** x+1	11/11	8.11%
x-2 x-1 **x x x x** x+1 x+2	14/13	7.83%
x-1 **x x x x x x** x+1	11/9	9.16%

New experiments (not reported here) were repeated using translators with larger hidden layers. However, the test translation WERs obtained were just slightly lower but at the cost of more computationally expensive networks.

5.2 Results training the MPs with other algorithms

The task was later approached using the two other learning algorithms presented in point 4 (RPROP and SCG) to train (for 3,000 more epochs) the pruned MPs which had provided the best pruned codifications in the previous section. Codifications were extracted from the resulting trained encoders (at 1,000 and 3,000 epochs) and evaluated on the translator. The test translation WERs obtained are shown in Table 4.

The results show how the RPROP algorithm did not produce a significant improvement over BEP, and also that SCG was definitely a worse learning method. Looking at the evolution (not reported in the paper) of the mean squared error obtained during the MPs training, we noticed that SCG had found a local minimum very soon. And we also observed that the mean squared error achieved with the SCG algorithm was bigger than that found using BEP. But BEP seemed to move around this minimum, increasing and decreasing the mean squared error in a short range. Combining BEP to approach a better local minimum (with minor mean squared error) and SCG to find exactly this minimum could be a good solution to the problem. The translation results achieved training pruned MPs for 100 epochs with BEP and continuing the training with SCG until 3,000 epochs can be also seen in Table 4. As expected, the final mean squared error was reduced and the translation results obtained (even after only 1000 epochs) were better than those obtained using only BEP to train the encoders. The pruned encoders were also trained using a combination of BEP (for 100 epochs) and RPROP (up to 3,000 epochs). However, the WERs achieved were higher than those achieved combining the BEP and SCG algorithms.

Table 4. Translation WERs using automatic codifications (of size 14 for Spanish and 13 for English) extracted from pruned MPs with output format x-2 x-1 **x x x x** x+1 x+2 and 25 initial hidden units trained with different algorithms

Training method	| Epochs | BEP+Other	Translation WER
BEP	1,000 + 0	6.90%
	3,000 + 0	7.83%
RPROP	0 + 1,000	7.02%
	0 + 3,000	6.06%
SCG	0 + 1,000	14.04%
	0 + 3,000	10.77%
BEP+RPROP	100+900	7.31%
	100+2,900	6.98%
BEP+SCG	100+900	4.9%
	100+1,900	5.56%

5.3 A different starting point

The results obtained at this point using automatic codifications were not as good as those obtained using manual codifications. One possible reason could be that we were being too demanding on the codifications, and that we could need more units to code the vocabularies. Consequently, we began the pruning process with MPs encoders which had 100 initial hidden units instead of the 25 units used in the above sections. They had an output window format *x-2 x-1* ***x x x x*** *x+1 x+2*. The resulting pruned hidden layers had 49 neurons for the Spanish encoder and 72 neurons for the English one. The training of these encoders continued for 3,000 more epochs using the BEP algorithm. The pruned encoders were also

trained for 100 and 500 epochs with the BEP algorithm and afterwards up to 3,000 epochs with the SCG algorithm.

After the training of the PM encoders, codifications were extracted and used to train RECONTRA models. Table 5 shows the translation WERs obtained on the test corpus. Most of them were better than those reported in Table 4 when smaller decoders were used. And these translation results were almost equal to those achieved with the best manual codifications, shown in Table 2. With regard to the mechanism for training the pruned MP encoders, we again observed (by looking at the evolution –not reported here- of the mean squared error obtained during the MPs training) that after training the encoder for 100 epochs with the BEP algorithm, the SCG method found a local minimun and stopped the learning. Beginning with MPs trained for 500 epochs with the BEP algorithm, the translation results were better. However, quite similar WERs were obtained using only BEP for the training of the pruned encoders.

Table 5. Translation WERs using automatic pruned codifications (of size 49 for Spanish and 72 for English) extracted from MPs with output window format x-2 x-1 **x x x x** x+1 x+2 and 100 initital hidden units

MP Encoder		Translation
Training method	| Epochs | BEP + Other	WER
BEP	1,000 + 0	2.28%
	3,000 + 0	1.85%
BEP+SCG	100+900	10.06%
	100+2,900	10.35%
BEP+SCG	500+500	2.58%

6 CONCLUSIONS AND FUTURE WORK

A method for automatically extracting word distributed representations for the RECONTRA translator [5] from a simple neural architecture was recently presented [4]. In this paper several learning algorithms were used to train this connectionist encoder aiming to improve the quality of the extracted codifications and, consequently, the corresponding translation results with RECONTRA: *Backward-Error Propagation* (BEP) [17], *Resilient back PROPagation* (RPROP) [16] and *Scalated Conjugate Gradient* (SCG) [12]. These algorithms were tested on a limited-domain MT task called the Mini-Traveller task [1]. The experimentation showed that, on the one hand, BEP and RPROP obtained similar translation WERs; SCG was clearly the worst algorithm since it fell quite soon into a local minima which did not leave. On the other hand, the combination of BEP and SCG led to translation results which were as good as those obtained using only BEP but ussually spending less training time. However, the method could be improved in the future and it would probably led to still better translation results.

A method for helping the exploration of the adequacy of the automatic codifications, provided by the encoders, could be also convenient. New algorithms for training the translators could be tried. And more complex MT tasks with larger vocabularies and higher semantic domains, which were nearer to real MT tasks, should be approached with this method.

REFERENCES

[1] Amengual, J.C., Castaño, M.A., Castellanos, A., Llorens, D., Marzal, A., Prat, F., Vilar, J.M., Benedí, J.M., Casacuberta, F., Pastor, M. & Vidal, E. (2000). *The Eutrans-I Spoken Language System*. Machine Translation (Vol. 15, pp. 75—102). Kluwer Academic Publishers.

[2] Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research* (Vol. 3, pp. 1137-115). Mit Press.

[3] Casañ, G.A. & Castaño, M.A. (1999). Distributed Representation of Vocabularies in the RECONTRA Neural Translator. In *Procs. of the 6th European Conference on Speech Communication and Technology*. (Vol. 6, pp. 2423—2426). Budapest, Hungary.

[4] Casañ, G.A. & Castaño, M. A. (2003). Automatic Word Codification for the RECONTRA Connectionist Translator. In F.J. Perarles, A.J.C. Campilho, N. Pérez de la Blanca, A. Sanfeliú (Eds.). *Lecture Notes in Computer Science: Pattern Recognition and Image Analysis* (Vol. 2652, pp. 168—175). Springer-Verlag.

[5] Castaño, M.A. (1998). Redes Neuronales Recurrentes para Inferencia Gramatical y Traducción Automática. *Ph.D. dissertation*, Universidad Politécnica de Valencia, Spain.

[6] Elman, J.L. (1990). Finding Structure in Time. *Cognitive Science* (Vol. 2, no. 4, pp. 279—311).

[7] González, R.C. & Thomason, M.G. (1978). Syntactic Pattern Recognition, An Introduction. Addison-Wesley.

[8] Hinton, G.E., McClelland, J.L. & Rumelhart, D.E. (1986). Distributed Representations. In Rumelhart, D.E. and McClelland, J.L. (Eds.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press. Cambridge, MA.

[9] Koncar, N. & Guthrie, G. (1994). A Natural Language Translation Neural Network. *Procs. of the Int. Conf. on New Methods in Language Processing* (pp. 71—77). Manchester, UK.

[10] Marzal, A. & Vidal, E. (1993). Computation of Normalized Edit Distance and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol.15, no. 9).

[11] Miikkulainen, R.P. & Dyer, M.G. (1991). Natural Language Processing with Modular Neural Networks and Distributed Lexicon. *Cognitive Science* (Vol. 15, pp. 393—399).

[12] Möller, M.F. (1993). A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks* (Vol. 6, pp. 525—533).

[13] Mozer, M.C. & Smolensky, P. (1990). Skeletonization: a Technique for Trimming the Fat from a Network via Relevance Assessment. In D.S. Touretzky, Morgan Kaufmann (Eds.). *Advances in Neural Information Processing* (Vol. 1, pp. 177-185).

[14] Pollack, J.B. (1990). Recursive Distributed Representations. *Artificial Intelligence* (Vol. 46, pp. 77—105).

[15] Prat, F., Casacuberta F. & Castro, M.J. (2001). Machine Translation with Grammar Association: Combining Neural Networks and Finite-State Models. *Procs of The Second Workshop on Natural Language Processing and Neural Networks* (pp. 54-64). Tokyo, Japan.

[16] Riedmiller, M. & Braun, H. (1993). A Direct Adaptative Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceeding of the IEEE International Conference on Neural Networks 1993*.

[17] Rumelhart, D., Hinton, G. & Williams, R. (1986). Learning Sequential Structure in Simple Recurrent Networks. In D.Rumelhart, J.L. McClelland and the PDP Research Group (Eds.) *Parallel Distributed Processing: Experiments in the Microstructure of Cognition* (Vol. 1). MIT Press.

[18] Waibel, A., Jain, A.N., McNair, A.E., Saito, H., Hauptmann, A.G. & Tebelskis, J. (1991). JANUS: A Speech-to-Speech Translation System using Connectionist and Symbolic Processing Strategies. *Procs. of the International Conference on Acustic, Speech and Signal Processing* (pp. 793—796).

[19] Zell, A. et al. (1995). SNNS: Stuttgart Neural Network Simulator. User manual, Version 4.1. *Technical Report no. 6195*, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, Germany.