Unsupervised Learning of Multiple Languages Using Recurrent Neural Networks

Miquel Perelló Nieto, Mathias Berglund, and Tapani Raiko
Deep Learning and Bayesian Modeling group,
Computer Science Department, Aalto University School of Science
{miquel.perellonieto, mathias.berglund, tapani.raiko} @aalto.fi

Abstract—We explore the capability of Recurrent Neural Networks (RNNs) to capture the statistics of hidden information on textual data. In this work, we focus on the identity of the text language as a hidden variable. We train different RNNs on a corpus containing text in different languages, without providing explicit information about the specific identity of the text. We compare RNNs and N-grams in terms of their prediction error, space complexity, and scalability to larger corpora.

Keywords—Machine learning, connectionism and neural nets, text analysis, language generation, information theory, natural language modeling.

INTRODUCTION

RECURRENT Neural Networks (RNNs) are a special type of Artificial Neural Network (ANN) in which some connections form directed cycles. These loop connections make them well suited for time-domain prediction tasks. This is because they can store an internal memory of previous time steps to help on the new predictions. It is this ability that has led to the extensive use of RNNs on tasks such as handwriting recognition (Graves et al. [1]) and speech recognition (Graves et al. [2]). Because of the depth of their loop connections, they are extremely difficult models to train (Bengio et al. [3]). The application of backpropagation through several layers causes the gradients to tend to zero or infinity. This is called the *"vanishing/exploding gradients problem"*. However, recently, approaches to train Deep Neural Networks (DNNs) have achieved state-of-the-art performance. In the context of image classification, a Deep Belief Network (DBN) was trained with an initial unsupervised phase and a final supervised phase that fine-tuned the parameters (Hinton et al. [4]). Furthermore, using a new adaptation of Hessian-Free Optimization Martens et al. ([5], [6]) managed to train DNNs and RNNs on solving various synthetic problems known to be *effectively impossible* using gradient descent (Hochreiter [7]). Using the same learning approach, Sutskever et al. [6] trained a large Multiplicative Recurrent Neural Network (MRNN) for the text prediction task at the character-level, achieving very good results.

Further work on DNNs has shown that upper hidden units can capture higher order representations of the original data, being able to create a new multi-dimensional space where similar objects can be clustered more easily. As a simple example, from raw pixels of an image, upper layers are able to extract edges, object parts and finally entire objects (from bottom to top layers)(See Zeiler et al [8] [9], Simonyan et al. [10]).

Since it is possible to view an RNN as a very deep architecture, our focus in this work is on extracting, in an unsupervised way, some abstract text representation from the raw characters. One of these hidden representations could be the identity of the language. In our experiments, we train RNNs using text on different languages (in this work Spanish and English) and we demonstrate that RNNs can achieve very good results, comparable to the best N-gram models with *add-one smoothing*.

1 N-GRAM MODELS

N-grams are sequences of n consecutive elements. In text analysis some of the most common levels of analysis, are at the word and/or character level. These two variants do not need any external knowledge about stems or syllables. At the word level, all N-grams are composed of consecutive words. In this case, the number of tokens can be in the order of millions. For example, the second edition of the 20-volume Oxford English dictionary contains full entries for 171,476 different words, excluding the different word inflections that may exist. An approximation of the possible number of tokens can be seen in Brants et al. [11], where the authors analyse publicly accessible web pages and retrieve approximately one trillion word tokens.

Other variants use character-level analysis, where, each N-gram is composed of a consecutive string of characters. In this case, the number of combinations is not as large as that of word-level analysis, but it still grows exponentially with respect to n, the size of the N-grams.

To create these models, we need to count all the possible occurrences of patterns of size n in a corpus. One of the optimal ways to store their frequencies is to use a look-up table. The benefit of this method is that each pattern is hashed and then stored or retrieved in constant time.

With all the corpus frequencies available, it is possible to compute the maximum likelihood estimate (MLE) for each of the possible characters.

$$P(c_n|c_1,\ldots,c_{n-1}) = \frac{\texttt{count}(c_1,\ldots,c_n)}{\texttt{count}(c_1,\ldots,c_{n-1})} \quad (1)$$

Where c_i corresponds to the *i-th* character of the pattern.

However, using MLE could cause problems on non-existent patterns in the training set, but smoothing techniques can be used to overcome that problem. We opted to use add-one smoothing:

$$P(c_n|c_1,\ldots,c_{n-1}) = \frac{\texttt{count}(c_1,\ldots,c_n)+1}{\texttt{count}(c_1,\ldots,c_{n-1})+V} \quad (2)$$

Where V is the vocabulary size.

This and other smoothing techniques can be seen in appendix **??**.

2 RECURRENT NEURAL NETWORKS

The first RNN appeared in 1982 by Hopfield [12]. This RNN was based on minimizing an internal energy state, and was able to store memories in a biological inspired manner. These kind of RNNs are called Hopfield Networks. Since then, multiple architectures have emerged: Simple Recurrent Networks (SRN) by Elman and Jordan 1991 [13], Continuous-time RNN (CTRNN) by Funahashi et. al. 1993 [14], Long Short Term Memory (LSTM) network by Hochreiter and Schmidhuber 1997 [15], Bidirectional RNN (BRNN) by Schuster and Paliwal 1997 [16], Echo State Network (ESN) by Jaeger and Hass 2004 [17].

In some of these networks, the real problem is in training them. Some of them use backpropagation as a learning method but Hochreiter [7] and Bengio et. al. [3] showed that the gradient decays (or explodes) exponentially in deep networks. Some solutions to this problem involves using sparsified connections and Hessian Free optimization (Martens et al. [5], [6], [18]). These solutions were shown to avoid some of gradient descents' problems, using which, DNNs and RNNs were able to be trained on pathological synthetic datasets which were previously arduous to learn with other techniques.

Recently Sutskever et. al. [6] trained a special kind of RNN named Multiplicative RNNs (MRNN) that enabled each input unit (character) to train their own hidden-to-hidden weight matrix, augmenting their expressiveness. Using this new architecture they got very good results that outperformed standard RNNs.

In this work, we focus on comparing the performance of N-gram models and a standard RNN with hidden to hidden loop connections, see Figure 1a.

RNNs are modeled with equations:

$$\boldsymbol{h_t} = \tanh(\boldsymbol{W_{hx}x_t} + \boldsymbol{W_{hh}h_{t-1}} + \boldsymbol{b_h}) \quad (3)$$

$$\boldsymbol{o_t} = \boldsymbol{W_{oh}h_t} + \boldsymbol{b_o} \quad (4)$$

Where x_t, h_t and o_t are the input, hidden and output units values on time step t respectively, b_h and b_o are hidden and output biases, and W_{ij} are the transformation matrix with the first $(_i)$ and second $(_j)$ subindex indicating destination and origin respectively.

For training the RNNs, we can use backpropagation through time (Rumelhart et al. [19]). This method consists of unfolding the network for some time-steps t and backpropagating the error through time (see Figure 1b). However, training a network with several layers could make the gradients to explode or vanish. In order to mitigate this problem we can sparsify the hidden-to-hidden connections and decrease the weights of each unit so as to ensure that ensure that the largest eigenvalue is smaller than one.

3 METHOD

In this section we describe the methodology we followed when conducting the experiments. First, we describe the datasets and the preprocessing steps carried out on them in Section 3.1. Second, we explain how we created the N-gram models in 3.2. Next, we explain how we trained the RNN and the hyper-parameters used in each model (3.3). Finally, we explain how we compared the performance of the N-gram models with that of the RNN in Section 3.4.

3.1 Datasets

We needed at least one corpus per language. In this initial setting we opted for British English and Spanish language corpora.

The English dataset is part of Wikipedia 2011, and it contains about one and a half billion characters. On the other hand, the Spanish dataset contains half a billion characters and is part of the "Total body of European Union (EU) law applicable in the EU Member States" from the years $[1958 - 2006]$. This corpus is available

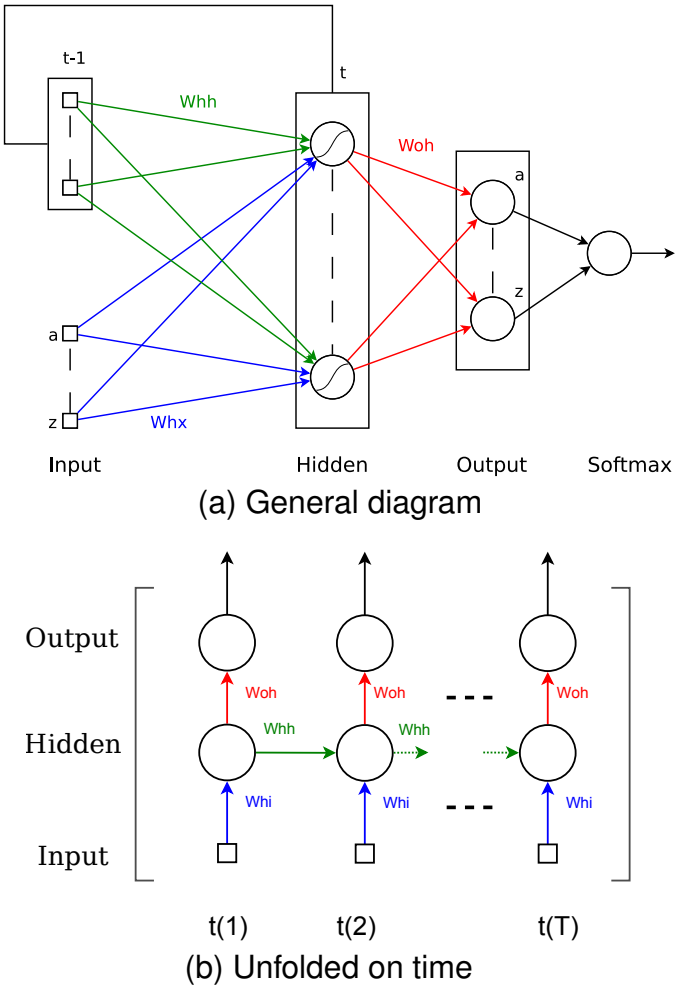


Fig. 1. **RNN diagrams**

from the Joint Research Center of the European Commission [20].

Both datasets were preprocessed and cleaned as per the following steps:

- All XML and meta-information was removed.
- The Spanish dataset contained the special character "ñ", and a large fraction of the words were accentuated. We therefore decided to replace all occurrencies of 'ñ' by 'n' and remove all accents. This was done in order to avoid possible clues about the text language.
- Only 86 characters were used in all the experiments, and we replaced all extra characters by the number sign #.
- All sentences shorter than fifty characters were removed (that is because our RNN was trained with fifty time-steps, and N-grams were trained with the same

TABLE 1
Initial and cleaned corpus

Language Version	Corpus Sentences	 Words
English	**Wikipedia**	
Original	3.368.700	228.878.857
Cleaned	3.222.670	205.400.638
Spanish	**JRC**	
Original	3.136.154	78.481.522
Cleaned	1.826.099	67.516.261

TABLE 2
Training data

Dataset	Sentences	Characters
en/sp train	2.921.758	899.037.749
en/sp test	730.440	225.019.457
en test	644.980	250.128.490
sp test	365.220	83.510.776

amount of textual data).

- All sentences containing words larger than thirty characters were removed. Most of these words were sequences without semantic meaning like numbers or hyperlinks. However, smaller invalid words could not be filtered in our data.

Table 1 is a summary of the corpus before and after the cleaning process (this table does not include the initial meta-information and XML data, it has been previously removed to show the real size of the corpus).

We used 80% of each corpus for training while the remaining 20% was used for testing. With these divisions, we created a mixed corpus that contained all the Spanish data, and an equal number of randomly selected sentences from the English dataset. We didn't use validation data in all the experiments, as we only experimented with two different RNN architectures, and we did not tune their training hyperparameters.

Table 2 shows the resulting mixed dataset with half of the data from each language.

More information about these datasets and their preprocessing steps is available in appendix **??**.

3.2 N-gram models

To generate the N-gram frequency table, we started from the cleaned training corpus. We split all training data into all possible character-level patterns of size n. Each of these patterns is introduced in a look-up table. If the pattern already existed in the look-up table, its corresponding frequency was incremented. If not, a new entry was created with its corresponding frequency set to 1.

With the frequency table built, we could compute the probability of any N-gram ending with one of the eighty six characters in constant time. We only needed to retrieve from the table, all the eighty six possible patterns and get their respective frequencies. With these frequencies, we could compute the MLE or other estimates.

3.3 RNN

We used the first fifty characters of each sentence to train the RNNs. The remainder was not used in this work as opposed to the N-gram models that got a larger amount of training data possibly giving the N-gram models an advantage [1]. The parameters of the RNN were updated using backpropagation through time.

Specific details about the RNN architecture used and training procedure are provided in section 4.

3.4 Performance measure

Standard N-gram models were used as a basis of comparison to asses the performance of our RNNs.

All models were trained with a corpus of mixed language text without any label or order that might reveal the identity of the language. After training, these models were tested separately on a test set with all languages or only one of them.

To compare the performance of different models we used the cross-entropy error measure:

$$H(p,q) = -\sum_{x} p(x) \log q(x) \tag{5}$$

1. We used only the first fifty characters of each sentence because of technical reasons in the implementation.

Where p is the real probability mass and q is our prediction.

Apart from giving us a measure to compare models, it also can be viewed as the amount of information needed on average to predict the next character, expressed in number of bits. In our case, when trying to predict the next character of a pattern, the expected amount of bits to codify the probability distribution of all eighty six possible characters is about 4.5 nats ($\ln(86)$) or 6.5 bits ($\log_2(86)$).

In this particular case the real probability is selected to be a Dirac delta function where its value is one in the actual sample output and zero otherwise. For this reason, we can reduce Eq. 5 to being the negative log-probability of predicting the correct character.

$$H(p, q) = -\log q(x_p) \tag{6}$$

To measure test performance, we used all the patterns on the test data set of size n for each N-gram model. We used the first $n-1$ characters as a context and computed the prediction error on the last character. In the case of RNNs the initial thirty characters of each sentence were used as a context while the last twenty characters were used to compute the error.

4 Experiments

During the experiments we first trained the N-gram models with n varying from one to nine (inclusive).

Models were trained in an English/Spanish corpus without labels. After training, these models were tested separately on English, Spanish and English/Spanish (mixed) test sets.

4.1 N-grams

As the amount of patterns increases exponentially with size of n, storing 9-grams in a table would result in the storage of 86^9 different patterns with their respective probabilities. That could be a problem if the corpus contained all possible patterns. However, we have a finite corpus and the sentences in a specific language do not allow all the possible combinations of patterns. This can be seen in the Figure 2 where the model sizes are in log scale, but the size growth is not linear (in log scale).

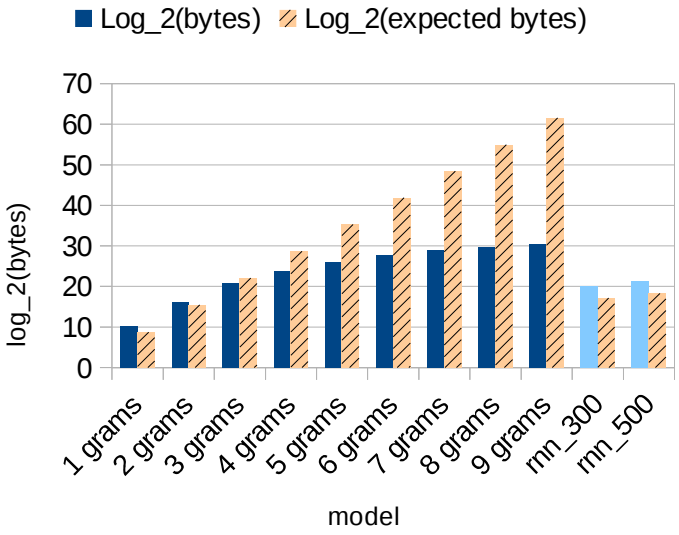


Fig. 2. **Model and expected sizes** in bytes on a log 2 scale

4.2 RNNs

We trained two different RNNs, one with 300 hidden connections and another one with 500. Both of them have 86 input and output connections (representing each of the possible characters). The hidden units compute a *hyperbolic tangent* (*tanh*) of the inputs and their previous hidden state, and the output units computes a *soft-max* in order to make a prediction.

The initial set of weights for input-to-hidden and hidden-to-output connections was chosen from a uniform distribution sample from the interval $[-0.1, 0.1]$. However, hidden-to-hidden weights were sparsified. They are initialized from a normal distribution with zero mean and unit variance. Only fifteen of the connections per neuron are conserved, while the rest is set to zero. After that, the weights are normalized using spectral normalization. This normalization helps preventis weights from *exploding* or *vanishing* when training across several time steps. We set the spectral radius to 0.95.

The learning rate of all the connections was 0.001, and momentum was 0.5. We did not use regularization nor learning rate decay.

In order to train with BPTT, we unfolded the RNNs through fifty time steps, and then used normal backpropagation to update the weights.

5 RESULTS

In this section we explain the results we obtained. We analyzed different performance measures in each subsection. First we compare their performance in terms of error rate 5.1. Then, we compare their variance 5.2, model size 5.3, and computation time 5.4. [2]

5.1 Error rate

N-gram models show a decreasing error rate with increasing value of n until their error starts increasing from n equal to seven (see Table 3 and set of Figures 3 and 4). The reason is that as the size of the patterns grow, it becomes less likely that the patterns in the training set also appear in the test set. In this case the probability of the last character is reduced to $1/86$, resulting in a worse prediction than that for smaller patterns. Concerning the two different languages, there is a small difference between their prediction. In case of Spanish, 7-grams gave the best results, while in English, 6-grams performed better.

The performance of RNNs increases with the number of hidden units and number of epochs (see Figure 3). We got the best results with 100 units and three epochs, which performed similarly to 5 and 6-grams. But the fact that increasing the number of epochs decreases the test error is an indication that the model is not overfitting the training data. Hence, it should be possible to decrease the error rate if we had more computational resources.

If we compare the best N-gram model (7-gram) and the best RNN model (500 neurons and 3 epochs), the difference in mean cross-entropy is in the range of $[0.04, 0.8]$.

If we compare the prediction error on English and Spanish datasets, we see that Spanish has a better prediction performance for all the models (Figure 4c). After analyzing the training corpus, we realized that the Spanish corpus is composed of law terms, and some sentences are really common in this field. Consequently, the Spanish corpus becomes simpler to learn and it gets better prediction performance.

2. The results of N-grams from one to three have not been included in the figures or tables because their performance was not comparable with the best models.

5.2 Variance

Variance in N-grams is constantly increasing with respect to the pattern size n. If we focus on the best N-gram models (6 and 7-grams) their standard deviation is in the order of 0.05. On the contrary, RNNs have a larger variance that does not show any significant variation with respect to the number of hidden neurons and epochs. In this case their standard deviation is about 0.06, which is similar to N-grams with $n = 8$.

5.3 Model size

Regarding the model size, N-grams are known to require an exponential amount of space with respect to n. But as our training set is limited and linguistic rules restrict the possible character combinations, the size seems to stop growing in an exponential manner (Figure 2). However, we know that increasing the amount of training data would increase the N-gram model size, which is not the case for RNNs. The number of parameters of our RNNs can be computed using the next equation:

$$hidd * (in + 1) + hidd * (hidd + 1) + hidd * out \quad (7)$$

Where *hidd*, *in*, *out*, are the number of hidden, input and output units respectively, and the ones represent the bias terms. In these experiments the largest RNNs have about 300,000 parameters (with five hundred hidden units $= 500 * (86 + 1) + 500 * (500 + 1) + 500 * 86$).

We can compare the model size of N-grams and RNNs by considering the most similar models with respect to their mean error. These are 5-grams, 6-grams and rnn500_3 (500 hidden units and 3 epochs). In these cases the RNN is between 26 and 79 times smaller than 5 and 6-grams.

5.4 Computation time

Training N-gram models requires linear time with respect to the text size, it uses constant time for table look-ups to increase or initialize its value. Then its training needs roughly a few hours depending on the value of n. However, as the table grows in size almost exponentially

TABLE 3
Average and standard deviation of cross-entropy error

lang.	4grams	5grams	6grams	7grams	8grams	9grams	rnn300	rnn500	rnn300_2	rnn500_2	rnn300_3	rnn500_3
averages												
mix	1.7713	1.4701	1.3273	**1.3193**	1.3996	1.5361	1.6108	1.5343	1.5244	1.4386	1.4768	**1.3984**
en	1.7972	1.5144	**1.3933**	1.4148	1.5310	1.7131	1.7460	1.6806	1.6560	1.5768	1.6112	**1.5363**
sp	1.7452	1.4259	1.2614	**1.2238**	1.2680	1.3590	1.4754	1.3881	1.3928	1.3004	1.3424	**1.2603**
std												
mix	**0.0342**	0.0384	0.0432	0.0520	0.0623	0.0731	0.0666	0.0669	0.0671	0.0670	0.0668	**0.0665**
en	**0.0282**	0.0316	0.0361	0.0437	0.0531	0.0626	**0.0587**	0.0592	0.0590	0.0591	0.0592	0.0588
sp	**0.0401**	0.0438	0.0488	0.0580	0.0687	0.0788	0.0725	**0.0722**	0.0728	0.0723	0.0725	**0.0722**

TABLE 4
Model size on $\log_2$ scale

	1grams	2grams	3grams	4grams	5grams	6grams	7grams	8grams	9grams	rnn300	rnn500
actual	10.10	16.15	20.90	23.87	26.06	27.68	28.88	29.80	30.54	**20.12**	21.36
expected	8.75	15.44	22.09	28.71	35.30	41.88	48.44	55.00	61.54	**17.12**	18.36

with n, depending on the implementation and the memory restrictions, the training time can increase exponentially and might even become impossible to train because of space requirements. Once it is trained, loading the entire model in memory can be prohibitive in some devices and can require a few minutes. Once loaded, the model can predict the next character in constant time it only needs to retrieve eighty six possible endings and compute their probabilities.

RNNs, on the other hand, require between 5 and 8 days of training per epoch, but their memory restrictions are very low. In our case, we trained two different RNNs for 3 epochs lasting around 42 days in total (21 on 3 epochs per model). Nevertheless, once trained, the model requires very little memory, and each prediction is done in constant time.

6 Discussion

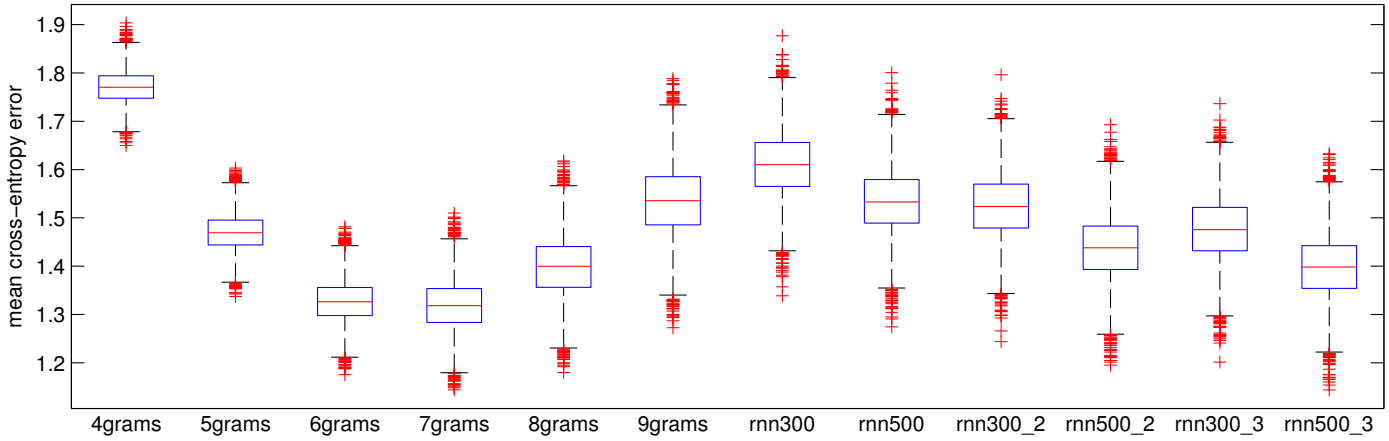
In this work, we trained RNNs on texts in multiple languages. After training, the networks predict the most probable next characters given an initial context in any of the trained languages. We analyzed the performance they achieved without tuning their hyperparameters.

We showed that RNNs got results comparable to 5 and 6 N-grams, while the best N-gram models where 6 and 7-grams. Furthermore, we saw that while the N-gram models starts overfitting, RNNs still had room for improvement through increasing the number of training epochs, and possibly tuning their hyperparameters.

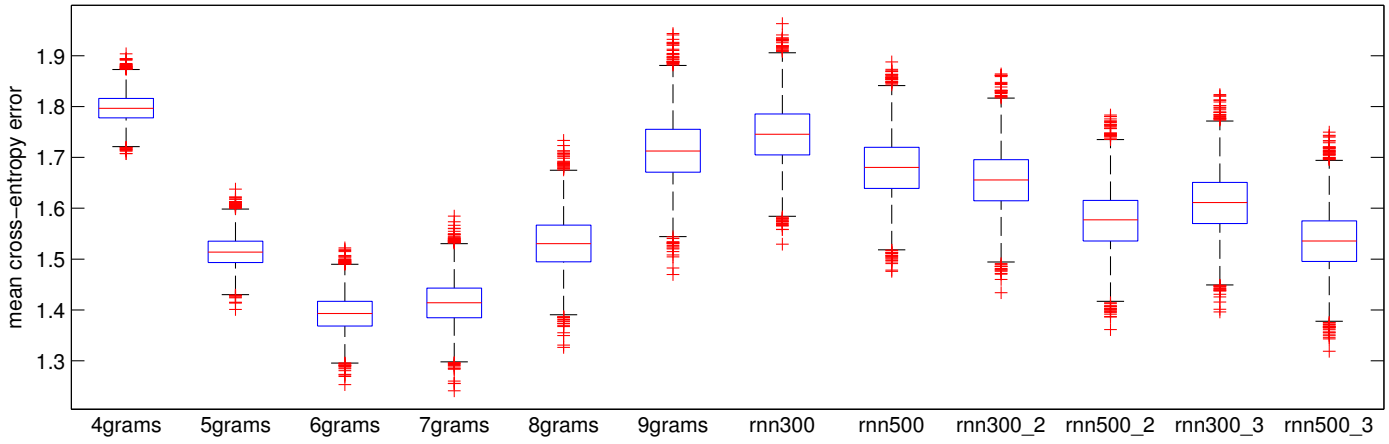
Although N-grams got the best results on these experiments, RNNs have been shown to get very good results with an extremely small model size. The RNNs are about 26 to 79 times smaller than the best N-gram models (2.6MB vs 490MB respectively). The compression rate that RNNs can achieve compared to N-grams can be very useful on small devices with low space capabilities. Furthermore, the number of parameters to load the N-grams model can be prohibitive if we need to load everything in memory.

Regarding computation time, N-grams are much faster to train than RNNs (some hours against more than 5 days respectively). However, once they are trained, RNNs make predictions more quickly than N-grams, with the exception of some N-gram implementations, or if they are completely loaded in memory and run in constant time.

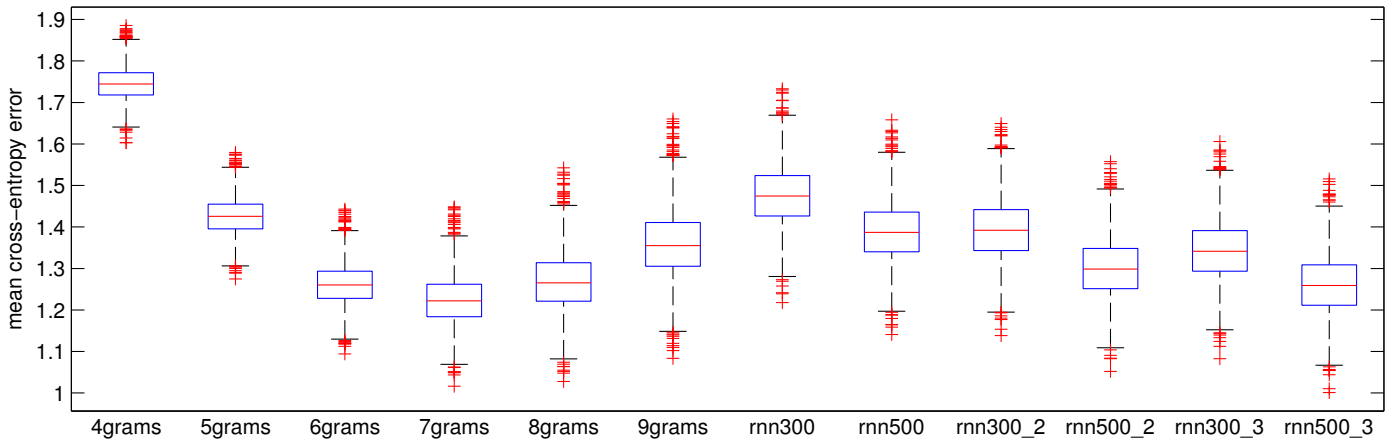
We noticed an obvious difference between the error rate for the Spanish and English test sets. This problem was likely to have been caused by the Spanish corpus, which contained



(a) Test on English and Spanish

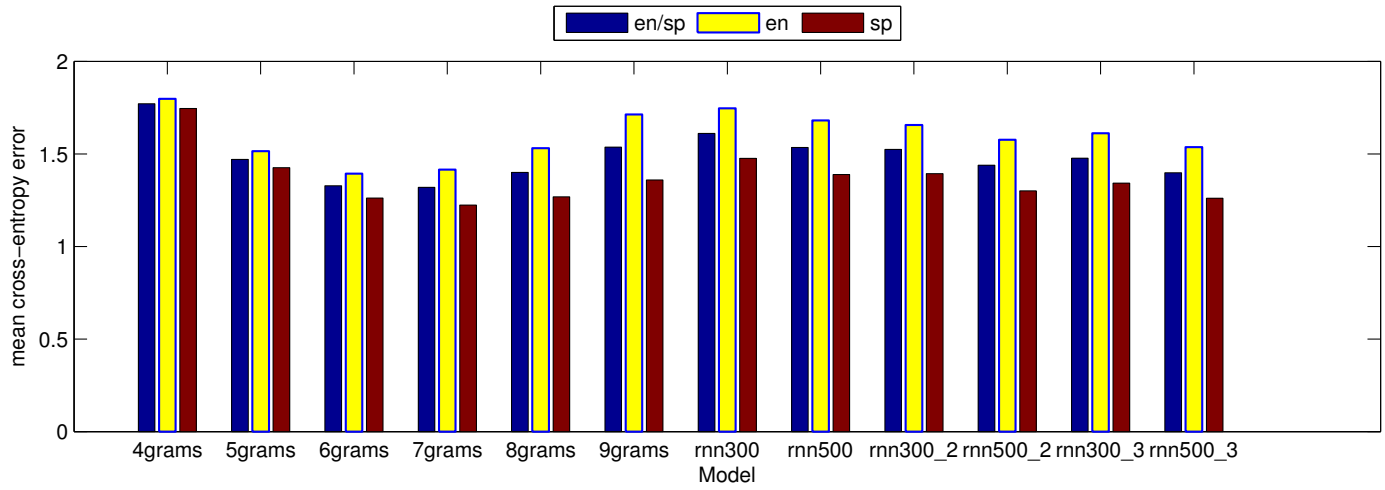


(b) Test on English

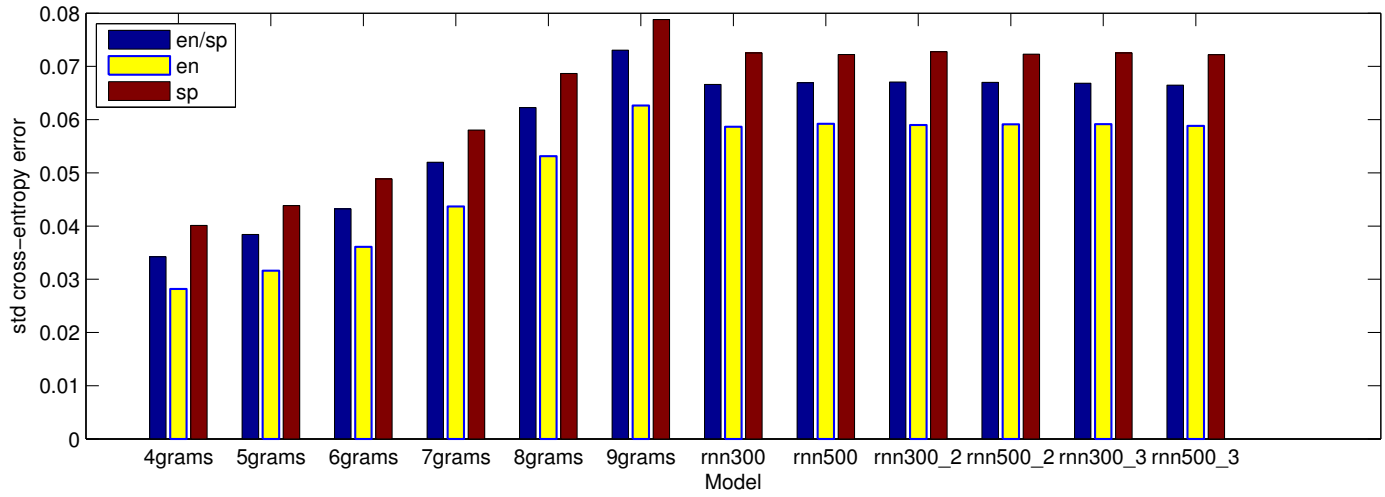


(c) Test on Spanish

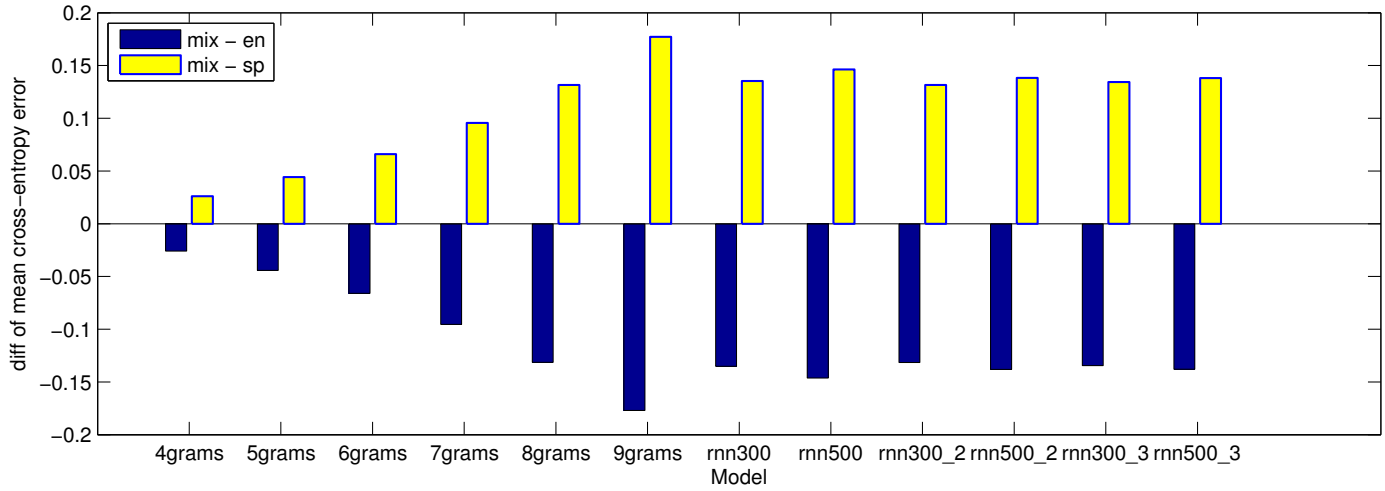
Fig. 3. **Model comparison** on different test sets with their mean cross-entropy error



(a) Mean cross-entropy error



(b) Standard deviation of cross-entropy error



(c) Cross-entropy difference between mixed, English and Spanish predictions: the upper part indicates that the error on the mixed corpus is bigger than that on the Spanish corpus, while bottom part indicates that the error in the mixed corpus is smaller than that on the English corpus.

Fig. 4. **Comparison of predictions depending on the language**

a large number of law-specific sentences that are repeated in training and test sets. As an example, the sentence "Visto el Tratado constitutivo de la Comunidad Europea" is repeated in the training set 12944 times and in the test set 3185. This problem can be solved by removing repetitions. A better solution would be to augment the training corpus with other texts available on the internet, such as the Spanish Wikipedia corpus, in order to mitigate the importance given to repeated sentences.

Further work could focus on improving the RNNs on this particular task: running more epochs, tuning their hyperparameters, increasing the training size or adding more languages. Furthermore, it should be possible to extend the same approach to other hidden statistics: texts from different authors, or different categories (Technology, Economics, Psychology or other topics), and investigate how RNNs encode this information in their hidden units.

ACKNOWLEDGMENT

The author would like to thank Vikram Kamath, Ehsan Amid, Karmen Dykstra and Antti Rasmus for their feedback and suggestions on writing this report.

REFERENCES

[1] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, May 2009. [Online]. Available: http://ieeexplore.ieee.org/ielx5/34/4804117/04531750.pdf?tp=&arnumber=4531750&isnumber=4804117http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4531750&tag=1

[2] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," May 2013, pp. 6645–6649. [Online]. Available: http://ieeexplore.ieee.org/ielx7/6619549/6637585/06638947.pdf?tp=&arnumber=6638947&isnumber=6637585http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6638947&tag=1

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 5, no. 2, pp. 157–66, Jan. 1994. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18267787

[4] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 1554, pp. 1527–1554, 2006. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527

[5] J. Martens, "Deep learning via Hessian-free optimization," *Proceedings of the 27th International Conference on Machine Learning*, 2010. [Online]. Available: http://machinelearning.wustl.edu/mlpapers/paper_files/icml2010_Martens10.pdf

[6] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," *Proceedings of the ...*, 2011. [Online]. Available: http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Sutskever_524.pdf

[7] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," *Master's thesis, Institut fur Informatik, Technische ...*, 1991. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Untersuchungen+zu+dynamischen+neuronalen+Netzen#0

[8] M. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *arXiv preprint arXiv:1311.2901*, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901http://arxiv.org/pdf/1311.2901.pdf

[9] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," *2011 International Conference on Computer Vision*, pp. 2018–2025, Nov. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126474

[10] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv preprint arXiv:1312.6034*, pp. 1–8, 2013. [Online]. Available: http://arxiv.org/abs/1312.6034

[11] T. Brants and A. Franz, "Web 1T 5-gram Version 1 LDC2006T13," Philadelphia: Linguistic Data Consortium, 2006.

[12] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/6953413

[13] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine Learning*, vol. 7, no. 2-3, pp. 195–225, Sep. 1991. [Online]. Available: http://link.springer.com/article/10.1007/BF00114844http://link.springer.com/content/pdf/10.1007%2FBF00114844.pdf

[14] K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S089360800580125X

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6795963http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

[16] M. Schuster and K. K. Paliwal, "Bidirectional recurrent

neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997. [Online]. Available: http://ieeexplore.ieee.org/ielx4/78/14188/00650093.pdf?tp=&arnumber=650093&isnumber=14188http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=650093&tag=1

[17] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr. 2004. [Online]. Available: http://www.sciencemag.org/content/304/5667/78http://www.ncbi.nlm.nih.gov/pubmed/15064413http://www.sciencemag.org/content/304/5667/78.full.pdfhttp://www.sciencemag.org/content/304/5667/78.short

[18] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," *...on Machine Learning (ICML-11 ...*, 2011. [Online]. Available: http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Martens_532.pdf

[19] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=FJblV_iOPjIC&oi=fnd&pg=PA213&dq=Learning+representations+by+back-propagating+errors&ots=zZCp6iI2WQ&sig=omUZ_4jXuKbkZb54ZmhHrAPCEGk

[20] R. Steinberger, B. Pouliquen, and A. Widiger, "The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages," *arXiv preprint cs/ ...*, pp. 2142–2147, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1397BA8B80FE7A34EE883EAE36515AEB?doi=10.1.1.145.5767&rep=rep1&type=pdfhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.5767http://arxiv.org/abs/cs/0609058

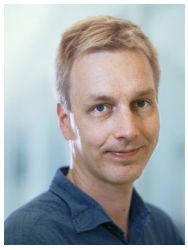
Miquel Perelló Nieto received his B.S. degree in Technical Engineering on Computer Science from Technical University of Catalonia in 2011. He is currently pursuing a double M.S degree on Artificial Intelligence (from a consortium between Technical University of Catalonia, Barcelona University and Rovira i Virgil University) and M.S on Machine Learning and Data Mining (from Aalto University School of Science). His primary research interest is artificial intelligence.

He is a Research Assistant within the Deep Learning and Bayesian Modeling research group in the department of Computer Science led jointly by Prof. Juha Karhunen and Assistant Prof. Tapani Raiko.

PLACE PHOTO HERE

Mathias Berglund received his M.Sc. degree in Industrial Engineering and Management in 2011 from Aalto University and his MSc degree in Management and Economics from the London School of Economics in 2009.

He is currently a PhD student within the Deep Learning and Bayesian Modeling research group in the department of Computer Science led jointly by Prof. Juha Karhunen and Assistant Prof. Tapani Raiko.



Tapani Raiko received his D.Sc. degree in Computer Science in 2006 from Helsinki University of Technology. He is an Assistant Professor (tenure track) and an Academy Research Fellow at Aalto University School of Science. His research focus is deep learning.