LSTM Recurrent Neural Networks for Short Text and Sentiment Classification

Conference Paper · May 2017

DOI: 10.1007/978-3-319-59060-8_50

3 authors, including:

Jakub Nowak
Czestochowa University of Technology
9 PUBLICATIONS **106** CITATIONS
SEE PROFILE

Rafal Scherer
Czestochowa University of Technology
154 PUBLICATIONS **1,178** CITATIONS
SEE PROFILE

LSTM Recurrent Neural Networks for Short Text and Sentiment Classification

Jakub Nowak[1], Ahmet Taspinar[2], Rafał Scherer[1]

[1] Computer Vision and Data Mining Lab, Institute of Computational Intelligence, Częstochowa University of Technology
Al. Armii Krajowej 36, 42-200 Częstochowa, Poland
{jakub.nowak, rafal.scherer}@iisi.pcz.pl
http://iisi.pcz.pl
[2] Data Scientist at CGI Nederland
info@ataspinar.com, http://ataspinar.com/

Abstract. Recurrent neural networks are increasingly used to classify text data, displacing feed-forward networks. This article is a demonstration of how to classify text using Long Term Term Memory (LSTM) network and their modifications, i.e. Bidirectional LSTM network and Gated Recurrent Unit. We present the superiority of this method over other algorithms for text classification on the example of three sets: Spambase Data Set, Farm Advertisement and Amazon book reviews. The results of the first two datasets were compared with AdaBoost ensemble of feed-forward neural networks. In the case of the last database, the result is compared to the bag-of-words algorithm. In this article, we focus on classifying two groups in the first two collections, since we are only interested in whether something is classified into a SPAM or an eligible message. In the last dataset, we distinguish three classes.

Keywords: recurrent neural networks, Long Short-Term Memory, Bidirectional LSTM, Gated Recurrent Unit, text classification, sentiment classification

1 Introduction

Human communication is a very complex and complicated process. One of the most important elements in verbal communication are speech sounds combined in words and sentences (phrases). We use dictionaries, i.e. collections of words describing our surroundings, feelings and thoughts. Very important is their order and context. In the paper we classify whole sentences into appropriate groups using recurrent neural networks (RNNs) [19], namely Long Short Term Memory (LSTM) [14] [18], its Birecursive LSTM (BSLTM) variant and Gated Recurrent Unit (GRU). LSTM were developed by Sepp Hochreiter and Jürgen Schmidhuber. The first observations on this subject were made by Sepp Hochreiter in his 1991 thesis but the beginning of LSTM was in 1997 (see [15]). Their recurrent network performed very well in comparison with previous RNNs, mainly thanks

to partial solving vanishing gradient problem[13] in RNNs. In the paper we use a variant with an additional modification, so called "peephole connections", proposed by Gers and Schmidhuber [8][10] with an additional link between the gate layers and the cell state. There are also other LSTM modifications with one especially interesting described in [7], where the Gated Recurrent Unit (GRU) was used [14, 5]. This is a truncated LSTM version without the output gate. This translates into faster network learning but it is recommended to use the classic LSTM network for more complex datasets. According to [23], GRU in some cases performs slightly better then LSTM.

The characteristic feature of the RNN network are gradual calls of the network over time. We do not provide all the input features for a single network call, as is the case in FNNs [2][4] or convolutional networks. In the case of RNNs, we gradually modify the final result. Intuitively, we can compare it to saying consecutive words in a sentence. Each subsequent word modifies the final meaning. In the case of these networks, we can also preserve the order of words in context. If we would like to use FNNs we would have to provide a whole sentence in one go. No matter if the sentence started with one word or another, it is difficult to implement such a FNN with the possibility of marking the occurrence of words. Thus, LSTMs are better in sentence content classification. The advantages of LSTM will be further described in the next chapter. Recursive networks are very widely used. Examples can be found in modelling simple physical processes such as ball bouncing [22]. Their greatest possibilities are seen in the translation of the text [1]. Every call is analyzed and appropriate word is selected in another language. If we use them for classification we are only interested in the result of the last network call.

2 Long Term Short Term Memory Networks

LSTM networks have their origins in RNN from 1980s. Their architecture allows for the accumulation of information during operation and uses feedback to remember previous network call states. It is often possible to find information that FNN networks are like RNN networks with only one call. This statement has a lot of truth in itself since their basic principles are similar to classic neural networks. Some mechanisms used in FNN are also applicable to recurrent networks. Classical recursive networks have a number of disadvantages [3] which narrows their ability to solve more complex problems. The community started to give them a special attention again with the advent of LSTM as it turned out that this technique performs much better than classical recursive networks. Detailed description of the architecture can be found for example in [18]. When using this structure, we activate the same cell every time, modifying the state of its internal structure. The most important elements of LSTM cells are

- cell state – the state of the cell passed in sequence after to the next steps,
- forget gate – the gate that decides what information should be omitted,
- input gate – a gate that decides what should be forwarded to the next activation.

In short, we care about remembering some information that is crucial for the final result and it is important to have some information omitted during the operation of the network, as not everything affects positively the network performance. A general network model used in this article is shown in Fig. 1. Each circle

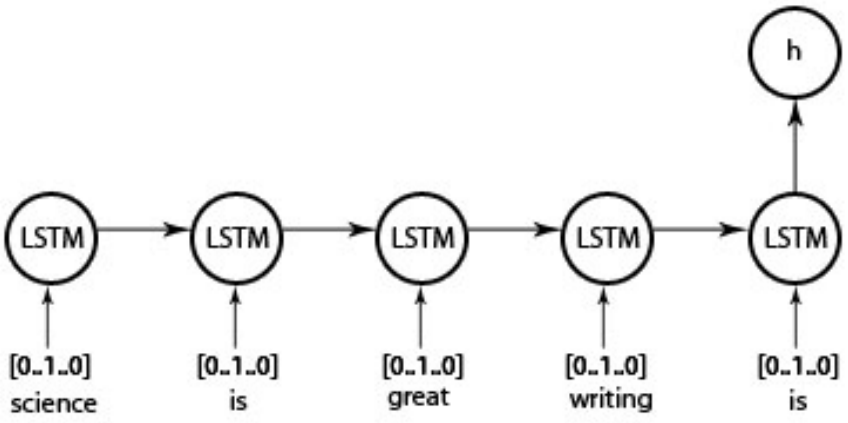

Fig. 1. Example of LSTM network call for text classification.

represents an LSTM cell. The input is given as a one-hot vector representing one word. Output h will return the result we are interested in.

2.1 Bidirectional LSTM

A single LSTM layer can, however, be too small in some examples. In the experiment we have checked whether the use of two LSTM layers gives better results. This model differs from the previous one in that the content is both propagated forward and backward through the network. Unlike in FNN it is not backpropagation learning. Originally, this structure is taken from [12]. The bidirectional network example is shown in Fig. 2.

2.2 Gated Recurrent Unit

Gated Recurrent Unit (GRU) was first described by Cho et al. in [7] and has a lot in common with LSTM. Forget and input gates are merged into a single "update gate", so was the cell state and hidden state. A more accurate comparison of LSTM to GRU can be found in [6]. In both cases, they are RNNs where the most important is their last state, which has the most influence on the current state of cells. Most importantly, in the case of GRU, control is done through the update gate.

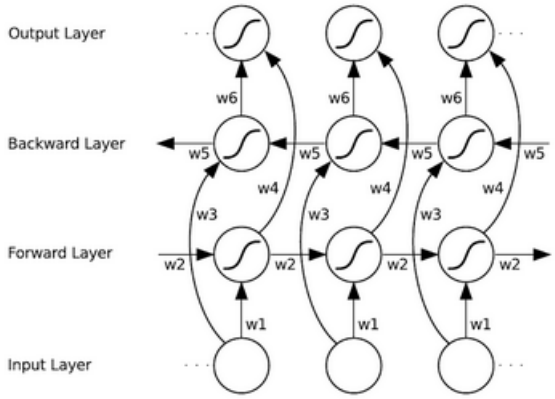

Fig. 2. Bidirectional LSTM [11]

.

3 Experiments

3.1 Experimental Setup

We used three datasets, i.e. Spambase Data Set (1999), Farm Advertisement (2011) and Amazon book reviews Data Set (2016). The SMS spam dataset contains a set of labeled messages that are collected from a UK forum. The data set consists of 13.4 % spam messages and 86.6% eligible messages [20]. A second database was collected from text advertisements found on 12 websites that deal with various farm animal related topics. There are 53.3% accepted messages and 46.7% rejected messages [12]. The last dataset contains 213.335 book reviews for 8 different books. The experiment was limited to five books. In selected books the amount of negative and neutral opinions was higher. For this experiment, the opinions are divided to three classes and presented in 1-5 scale. Score 1 and 2 are labeled as negative, score 3 represents a neutral comment, 4 and 5 represent a positive comment. From the set we took only comment titles, farther content were not considered. The aim was to classify the content as negative, neutral or positive on the basis of the opinion title. Table 1 contains the percentage distribution of opinions of selected books. The number of negative and neutral opinions was doubled for training to reduce imbalance with the number of positive opinions.

In the this section we compare the results of RNNs with the bag-of-word algorithm which consists in choosing characteristic words for a given class. We assume that some words give more meaning to a given sentence, and if the number of occurrences of the characteristic words for the assumed class is large then we associate the sequence with this class. We can use various textual features [17]. The SPAM SMS problem is still very important as despite the age of tech-

Table 1. A summary of the Amazon dataset

Title	Bad	Neutral	Good	Total number of texts
The Martian	2.31%	13.34%	84.35%	22571
The Goldfinch	16.22%	23.57%	60.22%	22863
Fifty Shades of Grey	36.2%	7.65%	56.24%	32977
Gone Girl	22.64%	11.94%	65.42%	41974
The Girl On The Train	19.92%	11.2%	68.87%	37139

nology we still obtain promotional offers of various products or services. In the case of short messages the spam filter should be almost faultless as it is very important for users to receive all legitimate messages. The last case considers a problem of book description and evaluation.

3.2 Grammar and Punctuation in Sequences

All data were normalized in the way that grammar and punctuation did not have influence on the results. It was important to obtain only a pure word stream which was better for neural network training. Namely, we took the following assumptions:

- only small letters were used,
- all special marks were deleted (e.g. . , %, ?),
- all numerical data were replaced by "SPEC-NUM" mark,
- for every dataset we created a special dictionary of unique words.

3.3 Word Representation

We created dictionaries of unique words for all three datasets without ordering them. During training, the data from the dictionaries were used to modify LSTM on the basis of only one word, (similarly to [16]) and an example is shown in Fig. 1. The number of input vector elements is equal to the number of words in the dictionary. Every word is represented by one vector value. We create as many input vectors as the number of words in one sequence. In this case the sequence is one sentence, one set of words, for which the classification result will be expected. For a FNN network it is possible to conclude the whole sequence by using one vector. All used words were represented by "1" (binary TRUE) and unused words "0" (binary FALSE). For RNN it is usually necessary to create a so-called one-hot vector where the value will be available only for one word what is the simplest possible method of representing input data.

3.4 Training

LSTM network learning is a specific process. The idea of recursive networks described earlier does not imply a fixed length of phrase and hence the number of network calls. In this case, we are required to use a special learning method called

back propagation through time (BPTT) [9, 24]. This algorithm was created with the intention of recursive networks. It is derived from the classical BP algorithm for FNNs except that the gradient is propagated backwards by all network calls, which entails a high memory requirement. For the first dataset (SPAM SMS) it was not so much important as sentences were not too long, so the the network was unfolded reasonable number of times. Much more demanding was the case with the Farm Advertisement dataset as the dictionary was much longer and so were the sentences. With 2 GB memory on the NVIDIA GeForce GT 740M GPU, the calculation ends with an error of insufficient memory available. To solve this problem, learning in this case was transferred to the Intel i5 2.6 GHz CPU with 12 GB RAM, however the memory was not fully utilized in this experiment. The network was implemented and trained using Microsoft CNTK [11] with the SGD (Stochastic Gradient Descent) algorithm.

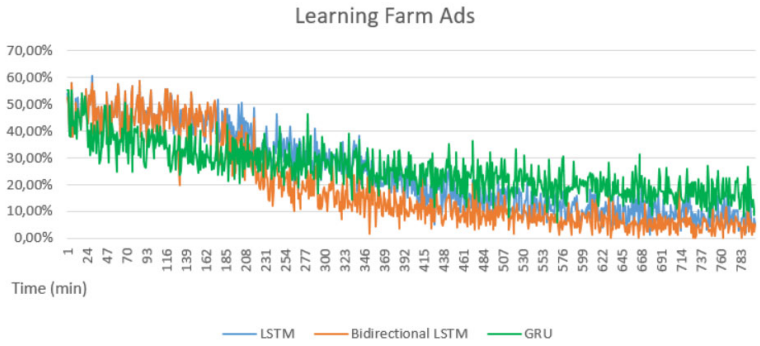

Fig. 3. Learning error.

The learning process can be compared on the example of Farm Ads. The internal parameters of the cells have been set to comparable values. Fig. 3 shows that GRU achieved the worst result, but it was achieved in 346 minutes of the learning process. After this time, it oscillated around the constant value. Unlike in the case of GRU, LSTM and BLSTM needed a longer learning time, but the training error become lower. In this case, the graph shows that BLSTM after 277 minutes had smaller error than the LSTM.

4 Results

Graph in Fig. 4 shows an example of LSTM operation on one of the used datasets. For the RNN network, we can see the classification result after consecutive words in the sequence. Each line represents the result returned by the network. In this case, the sequence can be classified into three different classes. It is important to note how certain words affect the result, for example "cool" greatly increases the positive and the neutral outputs and sets low in the "bad" output.

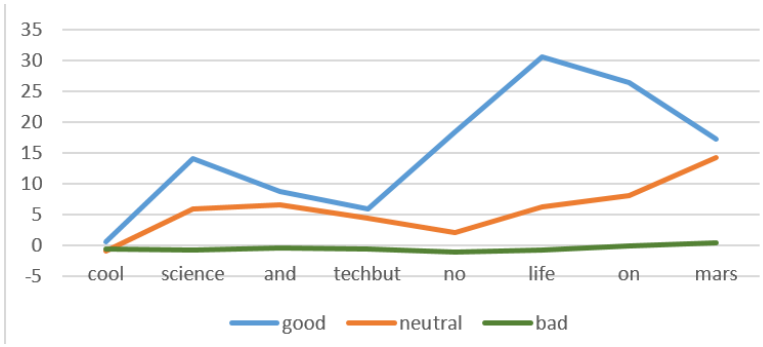

Fig. 4. An example of the incremental LSTM operation.

4.1 Results all datasets

The results of the experiment are presented in Tables 2, 3 and 4. The result

Table 2. Results obtained on the SPAM Collection dataset.

Dictionary size	9054
Number of outputs	2 (spam, ham)
Dimension, hidden layer	140
Accuracy, LSTM	99.798%
Accuracy Bidirectional LSTM	99.834%
Accuracy GRU	**99.945%**

Table 3. Results obtained on the Farm Ads dataset.

Dictionary size	54856
Number of outputs	2 (spam, ham)
Dimension, hidden layer	80
Accuracy, LSTM	94.497%
Accuracy Bidirectional LSTM	**96.017%**
Accuracy GRU	87.521%

from the first dataset is very satisfactory (Tab. 2) . The dictionary we worked on contained only 9,000 words as during our empirical tests it turned out that increasing this vocabulary did not improved the network performance and only training time was increased. In the case of the Amazon book reviews dataset (Tab. 4) we created one dictionary for all books. It should also be noted that GRU is not a good choice for classification based on very short phrases such as

Table 4. Results obtained on the Amazon book reviews dataset.

Dictionary size	16201
Number of outputs	3 (good, neutral, bad)
Dimension, hidden layer	140
Accuracy, LSTM	84.415%
Accuracy Bidirectional LSTM	**86.4%**
Accuracy GRU	75.821%

book titles and is better suited for simple datasets such as for example the SMS SPAM dataset.

The results obtained on the first two datasets can be compared to ones from [20]. Table 5 presents results from the literature and our results. It was more diffi-

Table 5. Results of classification accuracy comparison of various algorithms on two datasets.

Metod	SPAM Collection	Farm Ads
LSTM	99.798%	94.497%
Bidirectional LSTM	99.834%	96.017%
GRU	99.945%	87.521%
FNN [20]	98.58%	94.34%
AdaBoosting [20]	98.98%	84.03%

cult to compare the outcome of the experiments on the third dataset. At the time of writing, the only available results were based on [21] where the bag-of-words model classifies whether the opinion is negative or positive with 60% accuracy. In our case, we tried to classify into three different classes and in the best case we obtained 86.4% accuracy. We worked on the comment titles whereas in [21] whole comments were analyzed. Nevertheless, both LSTM and BLSTM resolved the problem more accurately. However, LSTM training consumes more computational resources and requires more time than the bag-of-words algorithm.

5 Conclusion

Proper understanding of written texts by the reader is sometimes difficult if the author does not express them clearly. Computer text undestanding is even more challenging. Attempts that use e.g. the bag-of-word algorithm are not yet fully satisfatory. The paper shows that much better solutions are recursive neural networks, especially in both LSTM and BLSTM form. In some cases it is also possible to use GRU. Classical FNNs generally give slightly worse results and only in the case of the Farm Ads dataset they are very close to LSTM.

In the Amazon book reviews dataset there are texts that are classified as negative as well as positive. In this case, the full comment description is necessary, limiting to the review titles only does not give perfect results and achieving

100% correct classification is not possible. It also important to take into account that RNN networks do not handle well very long phrases. Some problems appear already during the learning process, i.e. large memory requirements and a large value of error after more network calls. Moreover, in a long sentence, its first part can represent one class and the second part another one.

LSTM, BLSTM and GRU seems to be very similar constructions but as it has been shown in the experiments that GRU is well suited in simple cases where we want to obtain fast results. Advantages of GRU are faster training as they require less number of epochs to obtain the final result. Results achived by LSTM and BLSTM networks are more precise, but it is necessary to spend more time for the training process. GRU does not work with very short phrases such as in the case of third dataset. Each of the presented configurations has advantages and disadvantages and the type of dataset should be taken into account when the decision on the choice of the RNN network model is made.

References

1. Auli, M., Galley, M., Quirk, C., Zweig, G.: Joint language and translation modeling with recurrent neural networks. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA, Association for Computational Linguistics (October 2013) 1044–1054
2. Bas, E.: The training of multiplicative neuron model based artificial neural networks with differential evolution algorithm for forecasting. Journal of Artificial Intelligence and Soft Computing Research **6**(1) (2016) 5–11
3. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2) (1994) 157–166
4. Bertini Junior, J.R., Nicoletti, M.d.C.: Enhancing constructive neural network performance using functionally expanded input data. Journal of Artificial Intelligence and Soft Computing Research **6**(2) (2016) 119–131
5. Britz, D.: Recurrent neural network tutorial, part 4 implementing a gru/lstm rnn with python and theano. `http://www.wildml.com/` (October 27, 2015)
6. Chen, C., Xia, L.: Recurrent neural network and long short-term memory. `http://ace.cs.ohiou.edu/~razvan/courses/dl6890/presentations/lichen-lijie.pdf`
7. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
8. Gers, F.A., Schmidhuber, E.: Lstm recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks **12**(6) (2001) 1333–1340
9. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with lstm. Neural computation **12**(10) (2000) 2451–2471
10. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with lstm recurrent networks. Journal of machine learning research **3**(Aug) (2002) 115–143
11. Graves, A.: Supervised sequence labelling. In: Supervised Sequence Labelling with Recurrent Neural Networks. Springer (2012) 5–13
12. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks **18**(5) (2005) 602–610

13. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02) (1998) 107–116
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8) (November 1997) 1735–1780
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997) 1735–1780
16. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
17. Murata, M., Ito, S., Tokuhisa, M., Ma, Q.: Order estimation of japanese paragraphs by supervised machine learning and various textual features. Journal of Artificial Intelligence and Soft Computing Research **5**(4) (2015) 247–255
18. Olah, C.: Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs` (August 2015)
19. Patgiri, C., Sarma, M., Sarma, K.K.: A class of neuro-computational methods for assamese fricative classification. Journal of Artificial Intelligence and Soft Computing Research **5**(1) (2015) 59–70
20. Shuang Bi, W.Z.: Cs294-1 final project algorithms comparison deep learning neural network — adaboost — random forest. `http://bid.berkeley.edu/cs294-1-spring13/images/0/0d/ProjectReport(Shuang_and_Wenchang).pdf` (May 15, 2013)
21. Taspinar, A.: Sentiment analysis with bag-of-words. `http://ataspinar.com/2016/01/21/sentiment-analysis-with-bag-of-words/` (Januari 21, 2016)
22. Trask, A.: Anyone can learn to code an lstm-rnn in python. `https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/` (November 15, 2015)
23. Trofimovich, J.: Comparison of neural network architectures for sentiment analysis of russian tweets. (June 14, 2016)
24. Williams, R.J., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. Backpropagation: Theory, architectures, and applications **1** (1995) 433–486