Proceedings of the Federated Conference on Computer Science and Information Systems pp. 239–243
DOI: 10.15439/2016F134
ACSIS, Vol. 8. ISSN 2300-5963

Predicting Dangerous Seismic Activity with Recurrent Neural Networks

Karol Kurach
University of Warsaw
Email: kk236085@mimuw.edu.pl

Krzysztof Pawlowski
University of Warsaw
Email: kpawlowski236@gmail.com

***Abstract*—In this paper we present a solution to the AAIA'16 Data Mining Challenge. The goal of the challenge was to predict, from multivariate time series data, periods of increased seismic activity which may cause life-threatening accidents in underground coal mines. Our solution is based on Recurrent Neural Network with Long Short-Term Memory cells. It requires almost no feature engineering, which makes it easily applicable to other domains with multivariate time series data. The method achieved the 5th place in the AAIA'16 competition, out of 203 teams.**

I. Introduction

UNDERGROUND coal mine workers are exposed to a life-threatening danger in a form of seismic events. To improve workers' safety, it is crucial to predict those phenomena in advance. However, knowledge-based safety monitoring systems that are currently deployed in coal mines sometimes fail to forecast such occurrences early enough. The goal of the AAIA'16 Data Mining Challenge: Predicting Dangerous Seismic Events in Active Coal Mines competition [12] was to design methods that could improve reliability of seismic activity prediction.

The task is an instance of a classification problem with unbalanced data provided in a form of multivariate, non-stationary time series. We present a solution based on Recurrent Neural Network with Long Short-Term Memory cells. The proposed model is generic and does not rely on the domain knowledge. It requires only minimal feature preprocessing and no feature engineering or feature selection steps. The solution achieved a competitive 5th place in the AAIA'16 competition.

The rest of the paper is organized as follows. In Section II we give an overview of the related work. The details of the AAIA'16 challenge are described in Section III. Section IV gives a brief introduction to Recurrent Neural Networks and Long Short-Term Memory cells. In Section V we describe the details of our architecture, training and model selection. Finally, Section VI summarizes the paper.

II. Related Work

Seismic hazard and rock bursts pose a threat to miners' lives and overall safety of the coal mining operation. One of the techniques for addressing this problem is to monitor the sensor readings' with automated algorithms. Originally, natural earthquake seismology approaches have been used to deal with the problem [3]. Mine-induced seismicity can be assessed using mechanisms of mine tremors, such as magnitude, moment, stress drop and seismic efficiency [16] or using seismic tomography [10]. More recently, typical machine learning approaches have been used – such as Random Forests [4] and other nonlinear methods [5], including Support Vector Machines or Naive Bayes Classifier.

The recent IJCRS 2015 Data Challenge competition [13] provided an opportunity to compare different approaches on the data set coming from coal mining. Although the goal was a bit different (to predict dangerous levels of methane concentration), the data shared similar characteristics – being an example of a time series, multivariate prediction problem with concept drift. Most of the top solutions relied heavily on feature engineering, either manual or automatic, such as: automatic variable construction [1], window-based feature engineering [8], hand-crafted features [15] or thousands of automatically generated features [21][22].

III. Challenge description

A. Data

The aim of the AAIA'16 competition was to predict relative likelihood of seismic events in coal mines based on the recorded measurements. It is an instance of supervised learning classification task, with most of the data given in a form of non-stationary multivariate time series. The data is split into 5 training sets and a single test set. All the training sets together contain $133,151$ records, while the test set contains $3,860$ records. Each record describes a period of 24 hours and consists of:

- an identifier of the main working site and 12 characteristics related to the whole period of 24 hours, such as total energy of seismic bumps registered in the last 24 hours,
- 22 times series with 24 numeric per-hour aggregated measurements, such as energy of the strongest seismic bump within a given hour.

Thus, in total each record contains 541 values. As mentioned previously, the records are grouped into 5 training sets and a single test set. The subsequent training sets correspond to later periods, adjacent records in them overlap by 1 hour and are given in a chronological order. The test set contains records that come from period later than the last training set, its records are non-overlapping and given in random order.

A label is given for each record in the training set while for the test set such label is missing – it is the goal of the competition to forecast those values. The label is a categorical variable that can be either *normal* or *warning*. Value *warning* indicates that a total seismic energy measured within 8-hour period after the time covered by the record exceeded the warning threshold of $50,000$ Joules. For each record in the test set, the numeric predictions are to be made about those (hidden) values.

Additionally, there is an extra „meta-data" set that describes main working sites included in the training and test sets. It contains information such as the height of the main working site or the latest geological assessment. We note that the training and test sets are highly unbalanced, with respect to both the *main working site* attribute (Figure 1) and the labels (Table I).

B. Evaluation

The competition score is defined as an *area under the ROC curve*. It is calculated based on predictions of label values, made for all $3,860$ test set records. Each prediction is a number, where a higher value denotes that the true label value is more likely to be *warning*.

The contestants submit their predictions during the competition. However, before the competition is concluded, the contestants know only the score computed over *preliminary test set* – a part of the whole test set that contains approximately 25% of the data. This subset is chosen randomly by the organizers and is the same for all the contestants. It is not revealed to the participants which of the test records belong to it. The contestants can select a single final solution, possibly guided by the scores obtained on the preliminary test set. The final score, however, is computed over the *final test set*, which consists of the remaining approximately 75% of the test data. This score is shown only after the end of the contest and is used to compute the final standings – the highest-scoring team is declared the winner.

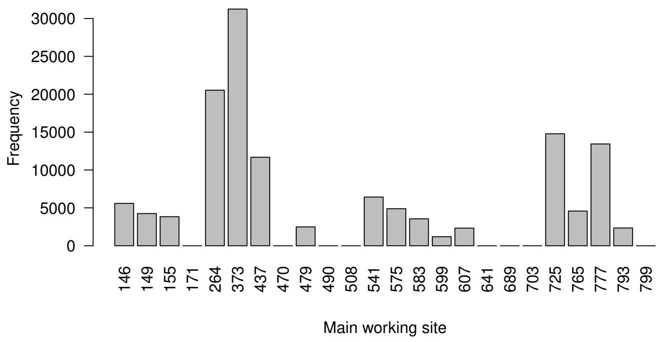

Fig. 1. The frequencies of different main working sites in the training sets. Some sites appear only in the test set.

TABLE I
Distribution of labels across the training sets

	tr. set 1	tr. set 2	tr. set 3	tr. set 4	tr. set 5
normal	78722	13137	13047	12744	12538
warning	1171	181	269	568	774

IV. Deep Recurrent Neural Network

A. Recurrent Neural Networks

Recurrent Neural Network (RNN) is a type of artificial neural network in which dependencies between nodes form a directed cycle. This allows the network to preserve a state between subsequent time steps. We focus on a simple RNN with a single, self-connected hidden layer.

RNNs process all elements from a sequence one-by-one, and the output at every time step depends on all previous inputs. This is a fundamental difference from feedforward networks, where the network's output depend only on the current element. It has an important theoretical implication: RNNs are capable of approximating arbitrary well any measurable sequence-to-sequence mapping [9].

Since RNNs contain loops, the standard backpropagation algorithm does not work. Instead, a *backpropagation through time* algorithm is used [20]. The idea behind this method is to unroll the network over N time steps, and copy the parameters N times. The RNN parameters are shared across all time steps, which makes them trainable and allows generalization.

Since the number of unrolled steps can be arbitrary, RNNs are particularly suited for modeling sequential data, where the length of the input is not fixed or can be very long. Recurrent nets have shown impressive results in many NLP tasks. One particularly successful variant of RNN is a recurrent network with LSTM cells, which we describe below.

B. Long Short-Term Memory

One important problem with training RNNs is the *vanishing gradient*, which can occur when values smaller than 1.0 are multiplied at each time step during the backpropagation through time. For some activation functions, the maximal value of the derivative is small. For example, the derivative of commonly used sigmoid function is never bigger than 0.25. As a result, after N time steps the gradient is multiplied by a value less than or equal to 0.25^N, which quickly becomes very small as N increases. While using some activation functions (eg. ReLU [17]) can reduce the likelihood of vanishing gradients, there is a special architecture designed to address this problem: Long Short-Term Memory (LSTM).

The LSTM is better at storing and accessing information than standard RNN [11]. The LSTM block consists of a self-connected memory cell and 3 gates named: input, output and forget. The gates control the access to the cell and can be interpreted as "read", "write" and "reset" operations in the standard computer's memory. The network learns to control the gates and decides to update and/or use the value at any given time step. Since all the components are built from differentiable functions, the gradients can be computed for the whole system and it is possible to train it end-to-end using backpropagation. There are several variants of LSTM that slightly differ in connectivity structure and activation functions. Below we describe the definitions of the input, output and forget gates that we used.

Let $h_t \in \mathbb{R}^n$ be a hidden state, $c_t \in \mathbb{R}^n$ be a vector of memory cells of the network and let x_t be the input at the time step t. Let W_i, W_f, W_u, W_o be matrices and b_i, b_f, b_u, b_o the respective bias terms. We define LSTM as a transformation that takes 3 inputs (h_{t-1}, c_{t-1}, x_t) and produces 2 outputs (h_t and c_t). In all equations below $\odot$ is element-wise multiplication. We assume also that $\oplus$ is an operation that aggregates h_{t-1} and x_t. We used plain sum, but concatenation of vectors is also commonly used.

The *forget gate* which decides how much of the information should be removed from the cell is defined as:

$$f_t = \text{sigm}(W_f * [h_{t-1} \oplus x_t] + b_f) \tag{1}$$

The *input modulation* gate value i_t and the cell update u_t are defined as:

$$\begin{aligned} i_t &= \text{sigm}(W_i * [h_{t-1} \oplus x_t] + b_i) \\ u_t &= \tanh(W_u * [h_{t-1} \oplus x_t] + b_u) \end{aligned} \tag{2}$$

Intuitively, input modulation decides how much of the u_t should be added to the memory at step t. For example, if x_t can be ignored, i_t will be close to 0. Knowing the values above, the new cell value c_t is computed as:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{3}$$

The last step is to compute h_t, the output passed to the next LSTM's time step. It is controlled by the *output gate* o_t:

$$\begin{aligned} o_t &= \text{sigm}(W_o * [h_{t-1} \oplus x_t] + b_o) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{4}$$

The LSTM networks have been successfully applied to real-world problems, including language modeling [18], handwriting [7] or speech [6] recognition, and machine translation [19].

V. Model

A. Preprocessing

Recall from Section II that most of the solutions to the previous challenge depend heavily on feature engineering. Such approach, while effective in practice, makes the model less generalizable as the feature engineering steps depend on the problem at hand. Our goal was to create a model that learns everything from the raw data and does not rely on the domain knowledge. To this end, we limit our preprocessing only to the following two operations:

- **Data normalization**, in regards to *mean* and *standard deviation*. This is a standard Machine Learning procedure, and as such it should be applicable to almost any problem. The normalization makes easier both optimization of the loss function and the regularization, because all feature values are at the same scale.
- **Upsampling positive examples**. As presented in Table I, the ratio of positive to negative examples is highly skewed. To make it more balanced, we sample with repetition from the set of positive examples and add them to the training set. We experimented with different upsampling ratios and achieved best results for increasing the number of positives by $10 - 20$ times.

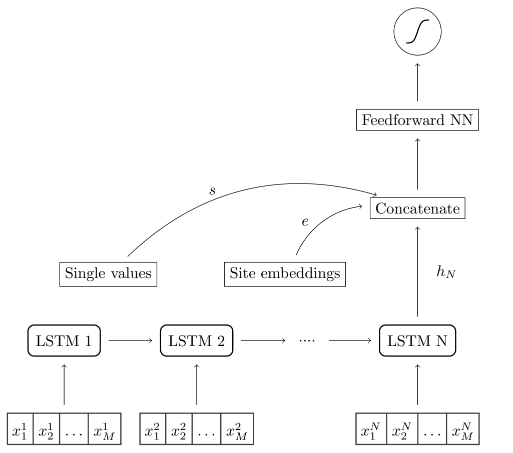

Fig. 2. Overview of the architecture. A core component is a single-layer LSTM unrolled for $N = 24$ time steps that processes $M = 22$ per-hour measurements. The ith per-hour measurement is marked as x^i. After processing N time steps, the last hidden state $h_N \in \mathbb{R}^{50}$ of the LSTM encodes information about all per-hour measurements. Then, h_N is concatenated with vector $s \in \mathbb{R}^{12}$ of per-record characteristics and the vector $e \in \mathbb{R}^{10}$ representing the working site id embedding.

B. Architecture

The overview of the architecture is presented in Fig. 2. We use a single-layer LSTM model that is processing 24 hourly aggregated measurements. At every time step, the hidden state of the LSTM ($h_i \in \mathbb{R}^{50}$) is connected to the previous state h_{i-1} and the normalized measurement values from the ith hour (x^i in the picture). After processing the whole sequence, the network's final hidden state h_N encodes all measurements in the order in which they appeared.

The vector h_N is concatenated with 2 other vectors: s and e. The vector s contains 12 per-record characteristics described in Section III-A. The vector e is an 10-dimensional embedding of the working site id. The values of the embedding vectors are initialized randomly and learned from the training data.

On top of the concatenation layer we build a standard supervised classifier (2-layer feedforward network in this case). We apply sigmoid on the network's output to ensure the predicted value is in the range $[0, 1]$ and can be interpreted as the probability of the *warning* label. The Binary Cross Entropy loss is used as the cost function.

TABLE II
WORKING SITES CHARACTERISTICS

site id	region name	bed name	assessment	mapped to
146	Partia F	416	a	N/A
149	Partia F	418	b	N/A
155	Partia H	502	b	N/A
171	Partia F	409	a	146
264	Z	405/2	b	N/A
373	G-1	707/2	b	N/A
437	G-1	712/1-2	b	N/A
470	Z	405/2	c	264
..	.....	....	.....	....
777	9	504	b	N/A
793	0	405	b	N/A
799	9	504	a	777

C. Training

We initialize all model's parameters by sampling uniformly from $[-0.1, 0.1]$. The optimization of the loss function is done using Adam algorithm [14] with a learning rate of 0.0005 and ε parameter equal to 10^{-10}. The training is run for 5 full passes (epochs) over the training data. After each epoch, the learning rate is multiplied by 0.63 and the training set is randomly shuffled.

We apply standard l_2 regularization of the weights with $\lambda = 0.01$. To avoid *exploding gradient* problem, the gradients are clipped globally to the value of 1. The model was implemented in Torch [2] and trained using a single GPU.

D. Model selection

Model selection was a significant challenge in the AAIA'16 competition. Recall from Section III-A that the time periods in the training data are overlapping. As a result, the standard cross-validation on a random split of the data tends to be over-optimistic. Also, there is a significant concept drift between the 5 provided training sets. The k-th training set was collected in a time period right after the set $(k-1)$th. We also know that the last training set was collected before the test set.

To address the problem of overlapping periods and to make the local evaluation as close to the final one as we can, we decided to use 5-fold cross-validation, with one training file being one fold. The average of the AUC scores was the final score we assigned to the model. We completely ignored the leaderboard score, as it proved to be very misleading in the past for this type of data [13].

E. Dealing with unknown sites

As described in Section V-B, our architecture computes embedding vectors for every working site id. However, recall from Fig. 1 that some of those identifiers exist only in the test data and not in the training data. We used the following method to fix this problem: we looked at the working site metadata and manually mapped 8 missing ids to existing ids that share similar characteristics. An example is presented in Table II which contains a subset of the metadata file. The id 171 is mapped to 146 because the region name and geological assessment is the same for those two sites. Similarly, id 799 is mapped to 777 because of the same region and bed names.

The above approach can be automated by joining (in SQL sense) the training data with the metadata on working site id attribute and then embedding different categorical variables (region/bed name, etc). This would remove the need of manual mapping of the missing ids and potentially improve the quality as well. However, we did not manage to try this approach during the competition.

F. Ensembling

From the begin of the competition, our main design decision was to create a competitive solution that consist of only one model trained from the raw data. However, the practice of machine learning competitions shows that ensembling of many different models is an easy way of improving the final score. We decided to do a simple rank average ensembling with a logistic regression model. More precisely, we use two models: RNN (described in section V) and LR (logistic regression) to evaluate records from the test set in the following way.

Let $\text{rank}_X(\text{record})$ denote the rank of the prediction given to the record by model X, among all predictions by model X on the test set. Then, the final prediction is computed as follows:

$$pred(record) = \frac{rank_{RNN}(record) + rank_{LR}(record)}{2}$$

By employing this technique we moved our solution one place up on the leaderboard.

VI. Conclusion

In this paper we presented a solution to AAIA'16 data mining challenge based on a Recurrent Neural Network with LSTM cells. It achieved a competitive score of 0.934 and the 5th place in the competition.

Compared to other methods (see Section II), our solution does not rely heavily on many hand-crafted features. Instead, it learns feature representation from the raw sensor data with a minimal feature engineering. It is a similar method to the one that we used in the previous IJCRS'15 competition, where our model achieved the 6th place. Top performance in both competitions suggests that our approach is versatile and can be successfully applied to different multivariate time series problems.

References

[1] Marc Boullé. Prediction of methane outbreak in coal mines from historical sensor data under distribution drift. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 439–451. Springer, 2015.

[2] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[3] C Allin Cornell. Engineering seismic risk analysis. *Bulletin of the Seismological Society of America*, 58(5):1583–1606, 1968.

[4] Long-jun Dong, Xi-bing Li, and PENG Kang. Prediction of rockburst classification using random forest. *Transactions of Nonferrous Metals Society of China*, 23(2):472–477, 2013.

[5] Longjun Dong, Xibing Li, and Gongnan Xie. Nonlinear methodologies for identifying seismic event and nuclear explosion using random forest, support vector machine, and naive bayes classification. In *Abstract and Applied Analysis*, volume 2014. Hindawi Publishing Corporation, 2014.

[6] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[7] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

[8] Marek Grzegorowski and Sebastian Stawicki. Window-based feature engineering for prediction of methane threats in coal mines. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 452–463. Springer, 2015.

[9] Barbara Hammer. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1):107–123, 2000.

[10] David R Hanson, Thomas L Vandergrift, Matthew J DeMarco, and Kanaan Hanna. Advanced techniques in site characterization and mining hazard detection for the underground coal industry. *International journal of coal geology*, 50(1):275–301, 2002.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Andrzej Janusz, Marek Sikora, Łukasz Wróbel, and Dominik Ślęzak. Predicting Dangerous Seismic Events: AAIA16 Data Mining Challenge. In *Proceedings of FedCSIS 2016*. IEEE, 2016. In print September 2016.

[13] Andrzej Janusz, Marek Sikora, Łukasz Wróbel, Sebastian Stawicki, Marek Grzegorowski, Piotr Wojtas, and Dominik Ślęzak. Mining data from coal mines: Ijcrsž201915 data challenge. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 429–438. Springer, 2015.

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Petre Lameski, Eftim Zdravevski, Riste Mingov, and Andrea Kulakov. Svm parameter tuning with grid search and its impact on reduction of model over-fitting. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 464–474. Springer, 2015.

[16] A McGarr. Some applications of seismic source mechanism studies to assessing underground hazard. In *Rockbursts and Seismicity in Mines.*, pages 199–208, 1984.

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[18] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, 2012.

[19] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[20] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

[21] Adam Zagorecki. Prediction of methane outbreaks in coal mines from multivariate time series using random forest. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 494–500. Springer, 2015.

[22] Adam Zagorecki. A versatile approach to classification of multivariate time series data. In *2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015, Lódz, Poland, September 13-16, 2015*, pages 407–410, 2015.