energies

MDPI

Article

Research on Unstructured Text Data Mining and Fault Classification Based on RNN-LSTM with Malfunction Inspection Report

Daqian Wei [1], Bo Wang [1,*], Gang Lin [1], Dichen Liu [1], Zhaoyang Dong [2], Hesen Liu [3] and Yilu Liu [3]

1 School of Electrical Engineering, Wuhan University, Wuhan 430072, China; weidq.whu@gmail.com (D.W.); glin@whu.edu.cn (G.L.); dcliu@whu.edu.cn (D.L.)
2 School of Electrical Engineering and Telecommunications, University of NSW, Sydney 2052, Australia; zydong@ieee.org
3 Department of Electrical Engineering and Computer Science, The University of Tennessee, Knoxville, TN 37996, USA; liuhesen@gmail.com (H.L.); liu@utk.edu (Y.L.)
* Correspondence: whwdwb@whu.edu.cn; Tel.: +86-159-7297-6215; Fax: +86-27-6877-2299

Academic Editor: Shuhui Li
Received: 23 December 2016; Accepted: 8 March 2017; Published: 21 March 2017

Abstract: This paper documents the condition-based maintenance (CBM) of power transformers, the analysis of which relies on two basic data groups: structured (e.g., numeric and categorical) and unstructured (e.g., natural language text narratives) which accounts for 80% of data required. However, unstructured data comprised of malfunction inspection reports, as recorded by operation and maintenance of the power grid, constitutes an abundant untapped source of power insights. This paper proposes a method for malfunction inspection report processing by deep learning, which combines the text data mining–oriented recurrent neural networks (RNN) with long short-term memory (LSTM). In this paper, the effectiveness of the RNN-LSTM network for modeling inspection data is established with a straightforward training strategy in which we replicate targets at each sequence step. Then, the corresponding fault labels are given in datasets, in order to calculate the accuracy of fault classification by comparison with the original data labels and output samples. Experimental results can reflect how key parameters may be selected in the configuration of the key variables to achieve optimal results. The accuracy of the fault recognition demonstrates that the method we proposed can provide a more effective way for grid inspection personnel to deal with unstructured data.

Keywords: deep learning; recurrent neural network (RNN); natural language processing (NLP); long short-term memory (LSTM); unstructured data; malfunction inspection report

1. Introduction

With the higher requirements of the economy and safety of the power grid, online condition-based maintenance (CBM) of power transformers without power outages is an inevitable trend for equipment maintenance mode [1,2]. The transformer CBM analysis relies on two basic groups: structured (e.g., numeric and categorical) and unstructured (e.g., natural language text narratives). Using structured data analysis, researchers have proposed a variety of transformer fault diagnosis algorithms such as the Bayesian method [3–5], evidence reasoning method [6], grey target theory method [7], support vector machine (SVM) method [8–10], artificial neural network method [11,12], extension theory method [13], etc. These algorithms have achieved good results in engineering practice. However, for more unstructured data found in practice [14–17] (i.e., the malfunction inspection report), traditional artificial

means of massive original document annotation and classification are not only time-consuming, but are also unable to achieve the desired results. For this reason, it has not been possible to adapt development of network information needs to the grid. Therefore, compared with structured data processing, it is more comprehensive for grid inspection personnel to effectively identify the massive unstructured text in the inspection malfunction report.

Deep learning achieved great success in speech recognition, natural language processing (NLP), machine vision, multimedia, and other fields in recent years. The most famous was the face recognition test set Labeled Faces in the Wild [18], in which the final recognition rate of a non-deep learning algorithm was 96.33% [19], whereas deep learning could reach 99.47% [20].

Deep learning also consistently exhibits an advantage in regards to NLP. After Mikolov et al. [21] presented language modeling using recurrent neural networks (RNNs) in 2010, he then proposed two novel model architectures (Continuous Bag-of-Words and Skip-gram) for computing continuous vector representations of words from very large data sets in 2013 [22]. However, the model is not designed to capture the fine-grained sentence structure. When using the back-propagation algorithm to learn the model parameters, RNN needs to expand into the parameter-sharing multi-layer feed-forward neural network with the length of historical information corresponding to the number of layers to expand. Many layers not only make the training speed become very slow, but the most critical problem is also the disappearance of the gradient and the explosion of the gradient [23–25].

Long short-term memory (LSTM) networks were developed in [26] to address the difficulty of capturing long-term memory in RNNs. It has been successfully applied to speech recognition, which achieves state-of-the-art performance [27,28]. In text analysis, LSTM-RNN treats a sentence as a sequence of words with internal structures, i.e., word dependencies. Tai et al. [29] introduce the Tree-LSTM, a generalization of LSTMs to tree-structured network topologies. Tree-LSTMs outperform all existing systems and strong LSTM baselines on two tasks: predicting the semantic relatedness of two sentences and sentiment classification. Li et al. [30] explored an important step toward this generation task: training an LSTM auto-encoder to preserve and reconstruct multi-sentence paragraphs. They introduced an LSTM model that hierarchically builds a paragraph embedding from that of sentences and words, and then decodes this embedding to reconstruct the original paragraph. Chanen [31] showed how to use ensembles of word2vec (word to vector) models to automatically find semantically similar terms within safety report corpora and how to use a combination of human expertise and these ensemble models to identify sets of similar terms with greater recall than either method alone. In Chanen's paper, an unsupervised method was shown for comparing several word2vec models trained on the same data in order to estimate reasonable ranges of vector sizes to induce individual word2vec models. This method is based on measuring inter-model agreement on common word2vec similar terms [31]. Palangi [32] developed a model that addresses sentence embedding, a hot topic in current NLP research, using RNN with LSTM cells.

In the above papers, the RNN-LSTM is continuously improved and developed in the process of deep learning applied to NLP. At present, RNN-LSTM has many applications in NLP, but it has not been applied to the unstructured data in the grid. Unstructured data accounts for about 80% of power grid enterprises, which contain important information about the operation and management of the grid. In the malfunction inspection report, the unstructured data processing method is urgently needed to analyze the effective information.

The primary objective of this paper is to provide insight on how to apply the principles of deep learning via NLP to the unstructured data analysis in grids based on RNN-LSTM. The remaining parts in the paper are organized as follows. In Sections 2 and 3, the description of the text data mining–oriented RNN and LSTM model are presented. In Section 4, the malfunction inspection report analysis method based on RNN-LSTM is proposed. Experimental results are provided to demonstrate the proposed method in Section 5. Conclusions are drawn in Section 6.

2. Text Data Mining–Oriented Recurrent Neural Network

2.1. Text Model Representation

For the vector form of voice and text in the mathematical model, each word represents a vector, and each dimension of the vector represents a single word. If the word appears in the text, it is set to 1; otherwise, it is set to 0. The number of vectors is equal to the dimension of the vocabulary words.

$$d_j = (w_{1,j}, w_{2,j}, ..., w_{t,j}) \tag{1}$$

2.2. Recurrent Neural Networks

In the past, voice text processing was usually a combination of a neural network and a hidden Markov model. Taking advantage of algorithms and computer hardware, the acoustics model established through deep forward-propagation networks has made considerable progress in recent years. Taking sound into account, text processing is an internal dynamic processing, and a RNN can be used as one of its candidate models. Dynamic means that the currently processed text vector is associated with the context of the content, and it cannot be an independent analysis of the current sample, but should be set before and after the memory unit of the text information for a comprehensive analysis of the semantic information. This approach applies a larger data state space and a more abundant model dynamic performance.

In a neural network, each neuron is a processing unit which is connected to the output of its node as the input. Before the output is issued, each neuron will first apply a nonlinear activation function. It is precisely because of this activation function that neural networks have the ability to model nonlinear relationships. However, the general neural model cannot clearly simulate the time relationship. All data points are composed of a fixed-length vector hypothesis. When there is a strong correlation with the input phasor, the model will greatly reduce the processing effect. Therefore, we introduce the recurrent neural network by given the ability of modeling with explicit time. The explicit time can not only into the output, but also in the next time step hidden layer, by adding across time points from the hidden layer and hidden layer feedback connection.

The traditional neural network has no middle layer of the cycle process. When the specified input $x_0, x_1, x_2, ..., x_t$, after the process of neurons there will be some corresponding output, $h_0, h_1, h_2, ..., h_t$. In each training, no information needs to transfer between the neurons. The difference between RNNs and traditional neural networks is that in every training for RNN, neurons need to transfer some information. In this training, the neurons need to use the role of the last neuron after the stated information, similar to the recursive function. The basic structure of the RNN is shown in Figure 1, and the expansion is shown in Figure 2, where A is the hidden layer; x_i is the input vector; h_i is the output of the hidden layer. As can be seen from Figure 2, the output of each hidden layer is input as an input vector to the next hidden layer. The output on the impact of the following can be considered in the next paragraph of the unstructured text. The algorithm of the model is analyzed in the Appendix A.

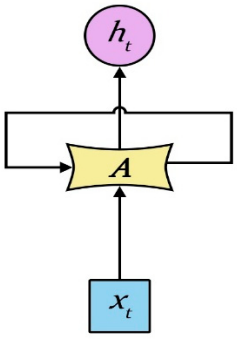


Figure 1. Recurrent neural network (RNN) basic structure.

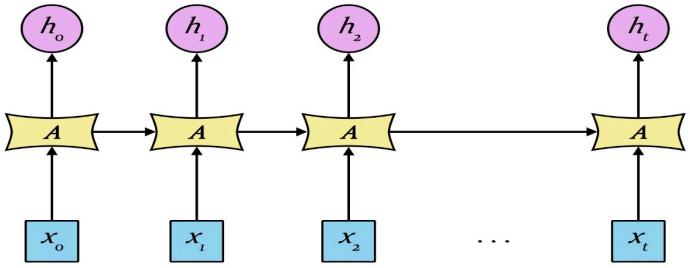


Figure 2. Recurrent neural network expansion.

3. Long Short-Term Memory Model

Although the RNN performs the transformation from the sentence to a vector in a principled manner, it is generally difficult to learn the long-term dependency within the sequence due to the vanishing gradients problem. The RNN has two limitations: first, the text analysis is in fact associated with the surrounding context, while the RNN only contacts the previous text, but not the following text; second, compared to the time step, RNN has more difficulties in the learning time correlation. A bidirectional LSTM (BLSTM) network can be used in the first problem, while the LSTM model can be used for the second. The RNN repeats the module as shown in Figure 3, which only contains one neuron. The LSTM model is an improvement of the traditional RNN model; based on the RNN model, the cellular control mechanism is added to solve the long-term dependence problem of the RNN and the gradient explosion problem caused by the long sequence. The model can make the RNN model memorize long-term information by designing a special structure cell. In addition, through the design of three kinds of "gate" structures, the forget gate layer, the input gate layer, and the output gate layer, it can selectively increase and remove the information through the cell structure when controlling information through the cell. These three "gates" act on the cell to form the hidden layer of the LSTM, also known as the block. The LSTM repeat module is shown in Figure 4, which contains four neurons.

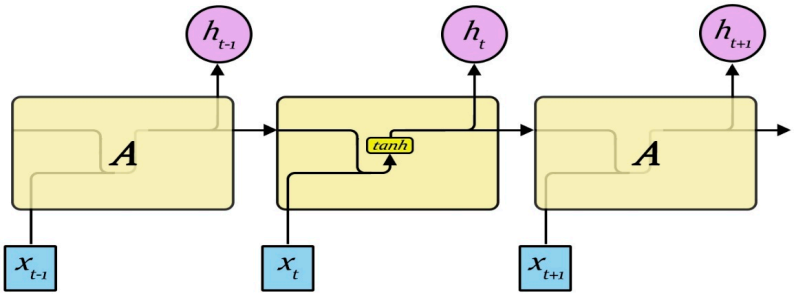


Figure 3. The repeating module in a standard RNN contains a single layer.

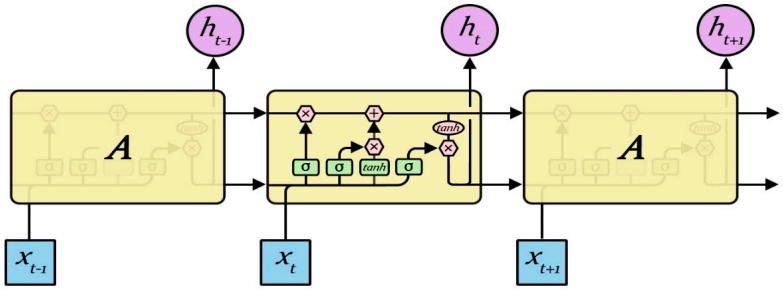


Figure 4. The repeating module in a long short-term memory (LSTM) contains four interacting layers.

3.1. Core Neuron

LSTM is used to control the transmission of information, it is usually expressed by *sigmoid* function. The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It is very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed of a *sigmoid* neural net layer and a pointwise multiplication operation. The state of the LSTM core neuron is shown in Figure 5.

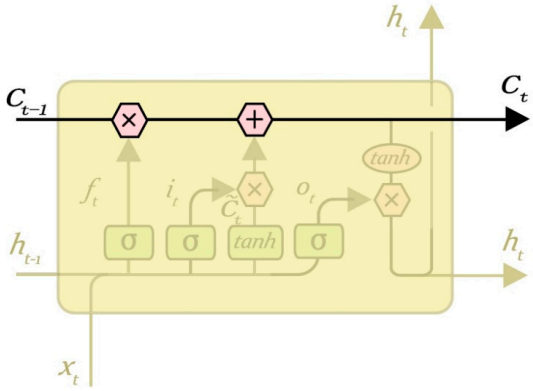


Figure 5. Neuron state transfer.

3.2. Forget Gate Layer

The role of the gate layer is to determine the upper layer of input information which may be discarded, and it is used to control the hidden layer nodes stored in the last moment of historical information. The forget gate computes a value between 0 and 1 according to the state of the hidden layer at the previous time and the input of the current time node, and acts on the state of the cell at the previous time to determine what information needs to be retained and discarded. The value "1" represents "completely keep", while "0" represents "completely get rid of information". The output of the hidden layer cell (historical information) can be selectively processed by the processing of the forget gate.

The forget gate layer is shown in Figure 6, when the input is h_{t-1} and the output is x_t:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2}$$

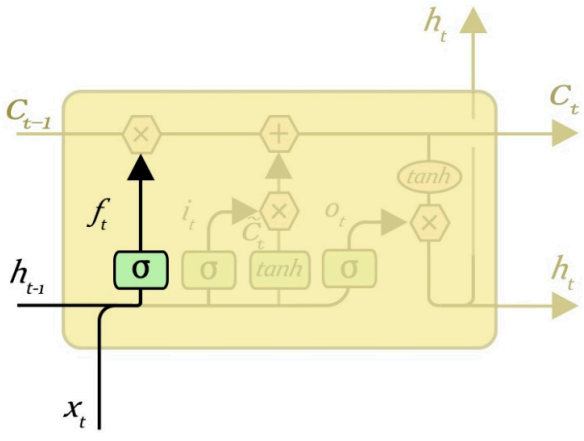


Figure 6. Forget gate layer.

3.3. Input Gate Layer

The output gate layer is used to control the input of the cell state of the hidden gate layer. It can input the information through a number of operations to determine what needs to be retained to update the current cell state. First the input gate layer is established through a *sigmoid* function to determine which information should be updated. The output of the input gate layer is a value between 0 and 1 of the *sigmoid* output, and then it acts on the input information to determine whether to update the corresponding value of the cell state, where 1 indicates that the information is allowed to pass, and the corresponding value needs to be updated, and 0 indicates that it may not be allowed as the corresponding values do not need to be updated. It can be seen that the input gate layer can remove some unnecessary information. Then a *tanh* layer can be established by adding the candidate state of the neuron phase phasor, and the two jointly calculate the updated value. The input gate layer is shown in Figure 7.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$

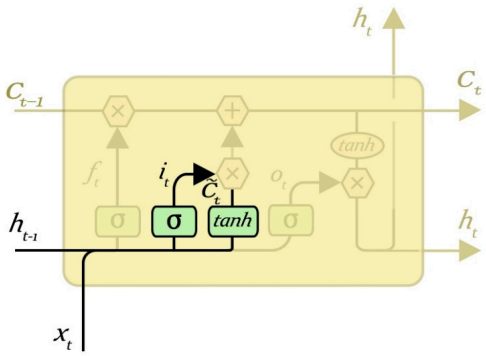


Figure 7. Input gate layer.

3.4. Update the State of Neurons

It is time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, and now we just need to actually do it. Multiply the old state by f_t, forgetting the information we decided to forget earlier; then add $i_t \times \widetilde{C}_t$. These are the new candidate values, scaled by how much we decided to update each state value. In the case of the language model, this is where we would actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps. The neurons' state update process is shown in Figure 8.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{5}$$

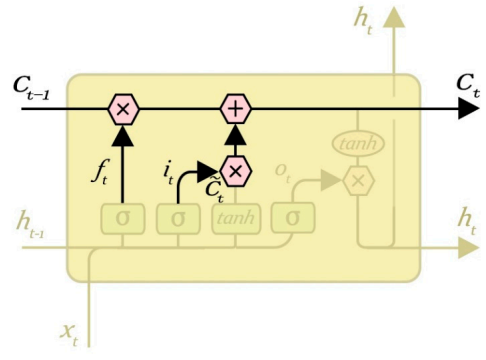


Figure 8. Neuron status update.

3.5. Output Gate Layer

The output gate layer is used to control the output of the current hidden layer node, and to determine whether to output to the next hidden layer or output layer. Through the output of the control, we can determine which information needs to be output. The value of its state is "0" or "1". The value "1" represents a need to output, and "0" represents that it does not require output. Output control information on the current state of the cell for some sort of value can be found after the final output value.

Determine the output of the neurons as (6) and (7), and the output gate layer is shown in Figure 9.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t * tanh(C_t) \tag{7}$$

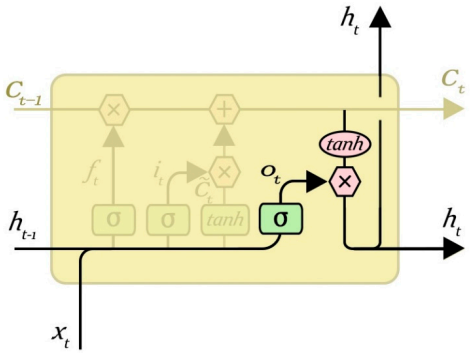


Figure 9. Output gate layer.

4. Malfunction Inspection Report Analysis Method Based on RNN-LSTM

To learn a good semantic representation of the input sentence, our objective is to make the embedding vectors for sentences of similar meanings as close as possible, and to make sentences of different meanings as far apart as possible. This is challenging in practice since it is hard to collect a large amount of manually labeled data that give the semantic similarity signal between different sentences. Nevertheless, a widely used commercial web search engine is able to log massive amounts of data with some limited user feedback signals. For example, given a particular query, the click-through information about the user-clicked document among many candidates is usually recorded and can be used as a weak (binary) supervision signal to indicate the semantic similarity between two sentences (on the query side and the document side). We try to explain how to leverage such a weak supervision signal to learn a sentence embedding vector that achieves the aforementioned training objective. The above objective to make sentences with similar meaning as close as possible is similar to machine translation tasks where two sentences belong to two different languages with similar meanings, and we want to make their semantic representation as close as possible.

In this paper, the algorithm, parameter setting and experimental environment are introduced as follows:

Input: training samples and test sample (including various types of reports and their labels). The number of truncated words: maxlen = 200. Minimum number of words: min_count = 5. The parameters of the model: Dropout = 0.5, Dense = 1. Activation function type: *Sofamax*, *Relu*, *tanh* and *sigmoid*. Number of LSTM units: 32; 64; 128; 259; 512. Output: classification results and fault recognition accuracy of test samples. Initialization: set all parameters of the model to small random numbers.

Experimental operating environment: operating system: Windows 10; RAM: 32 G; CPU: XeonE5 8-core; graphics: NVIDIA 980M; program computing environment: Theano and Keras.

The process of the training method for RNN-LSTM is presented in Appendix B.

5. Experimental Verification Based on Malfunction Inspection Report

The full learning formula for all the model parameters was elaborated on in the previous section. The training method was described in Section 4. In this section, we will take the power grid malfunction inspection report of a regional power grid in the China Southern Power Grid as the object of analysis. Through RNN-LSTM processing, we can use machine learning to classify and analyze the unstructured data in different situations.

5.1. Database

The description of the corpus is as follows. The malfunction inspection report records come in from the grid personnel by the inspection of the power grid equipment, lines, and protection devices during daily maintenance. The accumulation of fault-by-statement constitutes the main body of the report. Among them, the information in the malfunction inspection report is mainly composed of six main information bodies such as "DeviceInfo", "TripInfo", "Faultinfo", "DigitalStatus", "DigitalEvent", "SettingValue", and other common information. The TripInfo information body can contain multiple optional FaultInfo information. The FaultInfo information body indicates the current and voltage of the action, and it can clearly reflect and display the fault condition and the operation process through the report. The content source of DeviceInfo information can be a fixed value or a configuration file. The information of Faultinfo, DigitalStatus, DigitalEvent, and SettingValue can be different according to the type of protection or manufacturers. Faultinfo can be used as the auxiliary information of a single action message or as a fault parameter of the whole action group. The contents of each message are as follows:

(1) DeviceInfo: Description information portion of the recording apparatus.
(2) TripInfo: Partially records protection action events during the failure process.
(3) FaultInfo: Records the fault current, fault voltage, fault phase, fault distance and other information in the process of recording the fault.
(4) DigialStatus: Records the signal before the device into the self-test signal status.
(5) DigitalEvent: Records the change of events such as the self-test signal during the process of fault protection; all the switches are sorted according to the action time, and the action time and return time are recorded at the same time.
(6) SettingValue: Records the actual value of the device setting at the fault time.

According to the dynamic fault records of the power system, we divide all the faults into the following five categories and give the corresponding labels after each record: mechanical fault; electrical fault; secondary equipment fault; fault caused by external environment; fault caused by human factors.

We have selected the malfunction inspection report of the China Southern Power Grid for nearly 10 years as the data set of this paper. In the used data set, the specific types of faults and causes of faults, and the percentage of their statistics, are shown in Table 1. A single sample data size range is between 21 kb to 523 kb. Training samples and test samples were randomly selected to ensure the versatility of the model test.

In the semantic analysis, we also analyze the data sets used. In this paper, nine categories were selected to cover the semantic relations among most pairs of entities, and they make no overlap between them. However, there are some very similar relationships that can cause difficulties in recognition tasks, such as Entity-Origin (EO), Entity-Destination (ED), and Content-Container (CC), often appearing in one sample at the same time. Similarly, there are Component-Whole (CW) and Member-Collection (MC). Nine types of relationship profiles and examples are as follows:

(1) Cause-Effect (CE): Those ***cancers*** were caused by radiation ***exposures***.
(2) Instrument-Agency (IA): ***Phone operator***.

(3) Product-Producer (PP): A ***factory*** manufactures ***suits***.
(4) Content-Container (CC): A ***bottle*** full of ***honey*** was weighted.
(5) Entity-Origin (EO): ***Letters*** from foreign ***countries***.
(6) Entity-Destination (ED): The ***boy*** went to ***bed***.
(7) Component-Whole (CW): My ***apartment*** has a large ***kitchen***.
(8) Member-Collection (MC): There are many ***trees*** in the ***forest***.
(9) Message-Topic (MT): The ***lecture*** was about ***semantics***.

Table 1. Different fault type statistics in the dataset.

Fault Category	Fault Reason	Amount	Percentage	
Mechanical Fault	Turbine cooler fault	2736	25.6%	29.2%
	Switch mechanism oil leakage	312	2.9%	
	Monitor computer fault	39	0.3%	
	Low air pressure of equipment	44	0.4%	
Electrical Fault	Circuit breakers, disconnectors fault	3375	31.3%	43.5%
	Capacitor fault	502	4.7%	
	Unbalanced voltage and current	429	4.0%	
	Arrester fault	87	0.8%	
	Voltage and current transformer fault	126	1.2%	
	Battery fault	55	0.6%	
	Insulators blew	97	0.9%	
Fault caused by human factors	Wire facilities or referrals stolen	432	4.0%	4.0%
Fault caused by external environment	Lines and trees are too close	1991	18.6%	18.6%
Secondary equipment fault	Electromagnetic locking fault	55	0.6%	4.7%
	Remote control fault	445	4.1%	
Test sample	3217		30%	
Training sample	7508		70%	
Total	10725		100%	

The specific distribution of the number of samples in each category is shown in Table 2.

Table 2. Statistical distribution of relationship categories in samples.

Relation	Sample Quantity	Proportion of Sample
Cause-Effect (CE)	1331	12.4%
Instrument-Agency (IA)	1253	11.7%
Product-Producer (PP)	1137	10.6%
Content-Container (CC)	974	9.1%
Entity-Origin (EO)	948	8.8%
Entity-Destination (ED)	923	8.6%
Component-Whole (CW)	895	8.4%
Member-Collection (MC)	732	6.8%
Message-Topic (MT)	660	6.2%
Other	1872	17.4%
Total	10725	100%

5.2. Result and Analysis

Based on the RNN-LSTM training with massive single-fault samples, the test set is imported for fault-type accuracy testing. In this paper, three variables were selected. By comparing the fault recognition results under three different variables and the fault report, we obtained the fault recognition accuracy. When the other two variables are fixed, the test samples are verified by using different

traverse times. These three variables are: number of LSTM units, type of activation unit, batch size. Batch size is the size of each batch processing data and unique training methods of deep learning. The proper adjustment of Batch size not only can reduce the weight adjustment times to prevent over-fitting, also speeding up training.

5.2.1. Fault Recognition Accuracy and Number of LSTM Units

This experiment takes the activation unit and batch size at a constant rate, while the number of LSTM units gradually increases, and improves the number of traversals in the same LSTM units' number conditions. The number of LSTM training samples is 10,000, and the number of test samples is 3000; the activation unit is a *sigmoid*; the batch size is 20. The relationship between the accuracy rate and the number of LSTM units is shown in Table 3, and its trend is shown in Figure 10.

Table 3. Accuracy of fault recognition under different LSTM units.

Epoch (Traversal Times)	Number of LSTM Units				
	32	**64**	**128**	**256**	**512**
1	0.36045	0.38296	0.38847	0.41947	0.34154
5	0.41832	0.43441	0.46547	0.47669	0.38457
10	0.42052	0.48952	0.48952	0.50712	0.38457
15	0.47633	0.49903	0.50058	0.53964	0.39541
20	0.47633	0.50567	0.50856	0.58585	0.37854
30	0.49817	0.53811	0.53585	0.58684	0.36845
50	0.52576	0.53811	0.54273	0.61058	0.39574

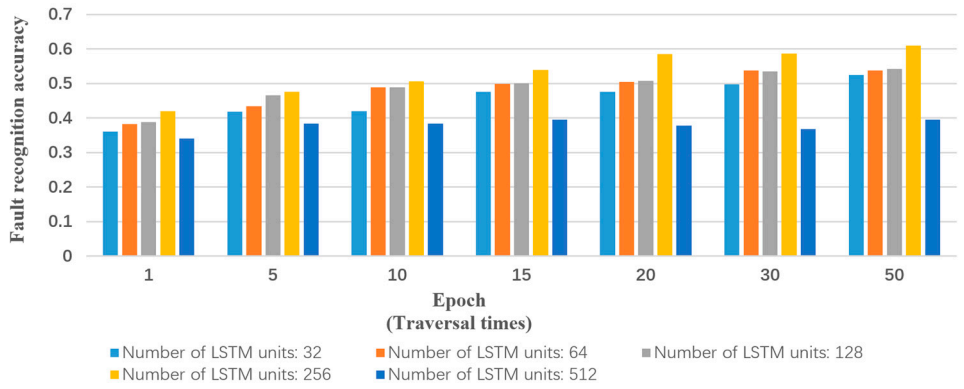


Figure 10. Accuracy of fault recognition under different LSTM units.

As can be observed from Table 3 and Figure 10, when the number of units in the LSTM remains constant, with the increase of the number of traversals, the higher the fault recognition accuracy rate is. When the number of LSTM units is the same, the greater the number of LSTM units, and the better the performances; however, a significant decline in the accuracy rate emerges when the number of LSTM units stays at 512. The reason for the decrease of the accuracy rate is that, as the required data volume increases, if more than 512 LSTM units are needed, the parameters need to be adjusted and optimized.

To further analyze the data, the receiver operating characteristic (ROC) curve system is added to the results. Due to the different performances by different numbers of LSTM units, we repeated experiments under different epoch conditions and chose three worth analyzing: 64, 128, 256. The area under the curve (AUC) reflects the ability of the recognition algorithm to correctly distinguish two types of targets. The larger the AUC is, the better the performance of the algorithm is. False negative (FN), false positive (FP), true negative (TN), and true positive (TP) are important parameters in the ROC curve. Specificity is defined as the true negative rate (TNR), and sensitivity is defined as the true positive rate (TPR). In the following experiment, the threshold was set to 0.5. The test was positive if

the accuracy of the fault recognition under different activation units was higher than the threshold value. As can be seen from Table 4 and Figure 11, in the proposed algorithm, the performance of the algorithm tends to be better in a certain interval with the increase of the number of LSTM units.

Table 4. Area under the curve (AUC) analysis of receiver operating characteristic (ROC) curves under different LSTM unit numbers.

Number of LSTM Units	AUC	Standard Error	Lower Bound (95%)	Upper Bound (95%)
64	0.6994	0.0785	0.5456	0.8533
128	0.7724	0.0666	0.6419	0.9030
256	0.8319	0.0588	0.7165	0.9473

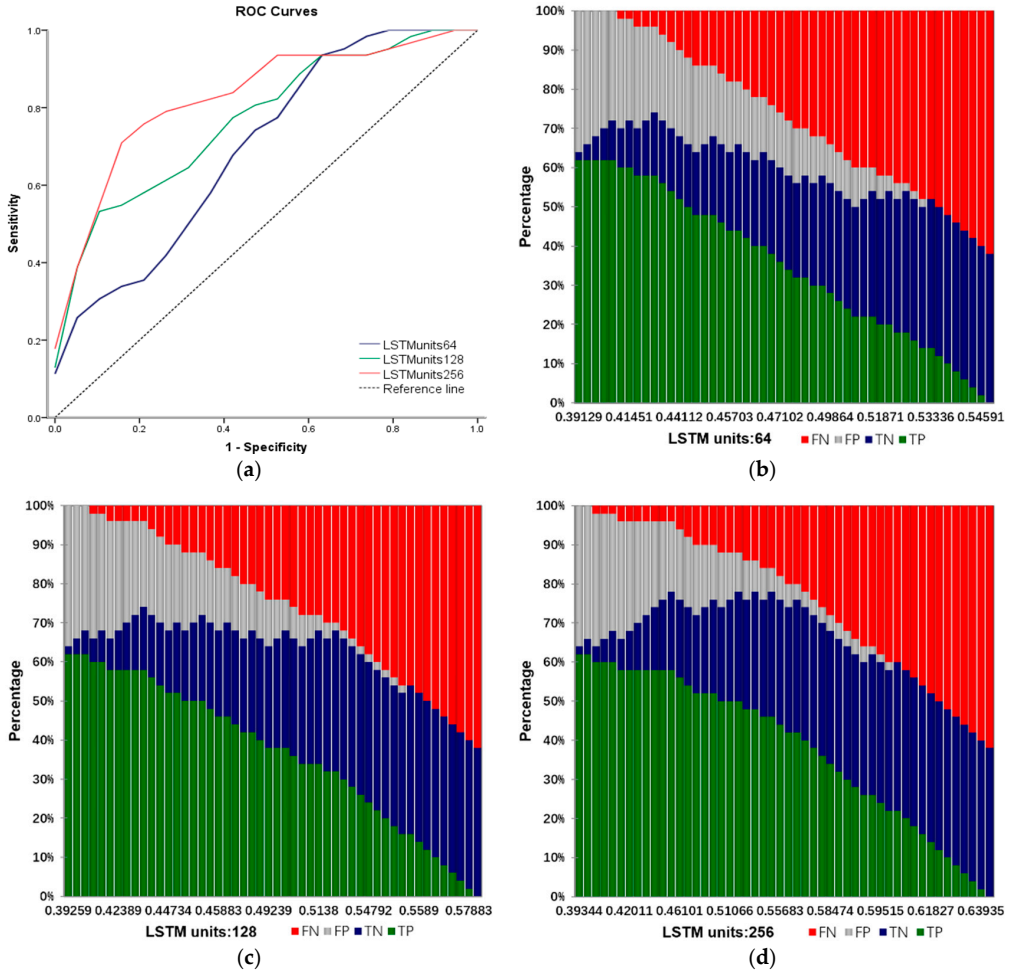


Figure 11. (**a**) Comparison of ROC curves under different LSTM unit numbers. The percentage of FN/FP/TN/TP under (**b**) 64 LSTM units; (**c**) 128 LSTM units; and (**d**) 256 LSTM units. FN/FP/TN/TP: false negative/false positive/true negative /true positive.

5.2.2. Fault Recognition Accuracy and Activation Unit Type

This experiment keeps the number of LSTM units and the batch size at a constant rate, while selecting four different activation units, and improving the number of traversals in the same activation unit conditions. The number of LSTM training samples is 10,000, and the number of test samples is 3000; the number of LSTM units is 128; the batch size is 20. The relationship between the accuracy rate and the different activation units is shown in Table 5, and its trend is shown in Figure 12.

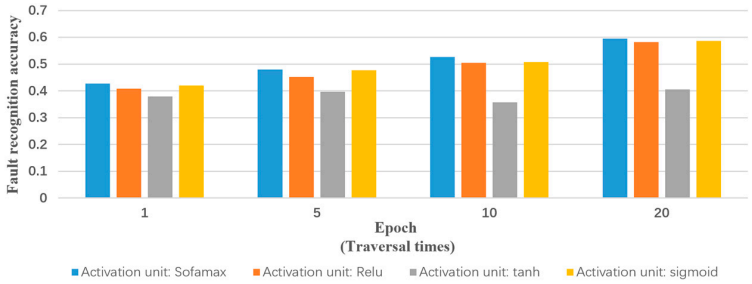


Figure 12. Accuracy of fault recognition under different activation units.

As can be observed from Table 5 and Figure 12, under the same activation unit condition, with the increase of the number of traversals, the fault recognition accuracy is higher. With the same number of traversals, the use of *Sofamax* and the *sigmoid* activation unit will obtain a better accuracy, the *Relu*'s performance followed. It can be seen that the greater the number of traversals, the *Relu* and *sigmoid* performances become closer, but the results obtained using the *tanh* change are not obvious. Thus, in the choice of activation function, *Sofamax* and *sigmoid* are more suitable for text processing.

Table 5. Accuracy of fault recognition under different activation units.

Epoch (Traversal Times)	Activation Unit	Activation Unit	Activation Unit	Activation Unit
	Sofamax	*Relu*	*tanh*	*Sigmoid*
1	0.42797	0.40868	0.37974	0.41947
5	0.48042	0.45259	0.39684	0.47669
10	0.52669	0.50478	0.35741	0.50712
20	0.59572	0.58163	0.40587	0.58585

We also selected the above four activation functions for ROC analysis by repeated experiments under different epoch conditions. In the following experiment, threshold was set to 0.5. The test was positive if the accuracy of the fault recognition under different activation units was higher than the threshold value. As can be seen from Table 6 and Figures 13 and 14, *Sofamax* and *sigmoid* activation performed the best. It also confirms the above conclusions.

Table 6. AUC analysis of ROC curves under different activation units.

Activation Unit	AUC	Standard Error	Lower Bound (95%)	Upper Bound (95%)
Sofamax	0.8889	0.0477	0.7953	0.9824
Relu	0.7743	0.0692	0.6386	0.9099
tanh	0.6267	0.0817	0.4665	0.7868
sigmoid	0.7916	0.0667	0.6607	0.9225

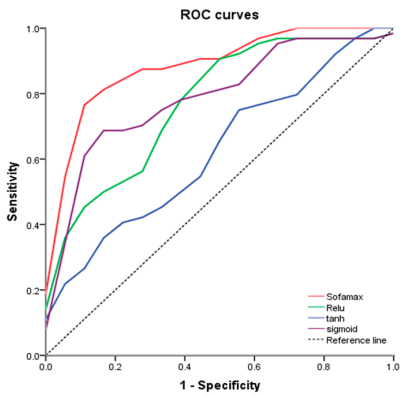


Figure 13. Comparison of ROC curves under different activation units.

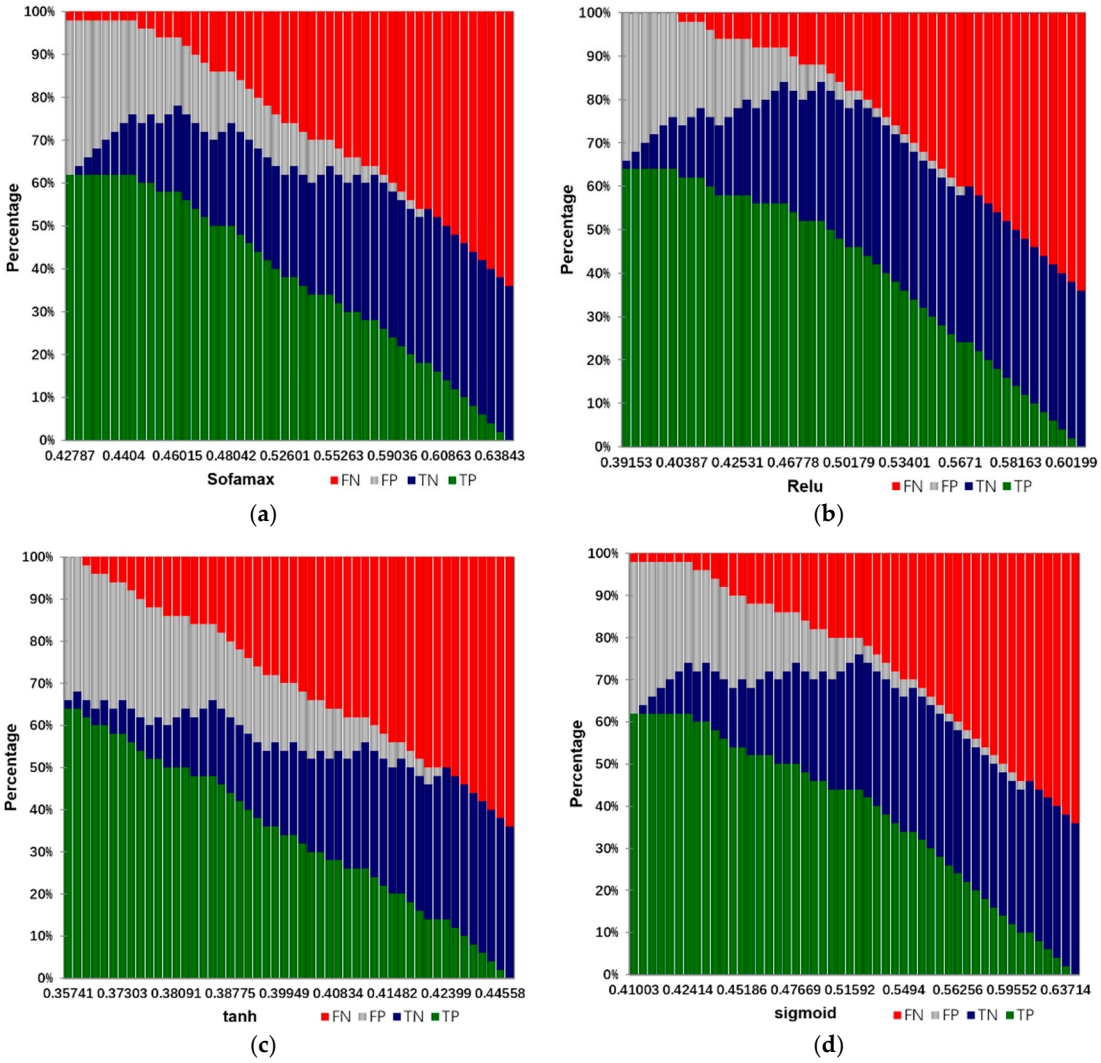


Figure 14. The percentage of FN/FP/TN/TP under the activation unit of (**a**) *Sofamax*; (**b**) *Relu*; (**c**) *tanh*; and (**d**) *sigmoid*.

5.2.3. Accuracy of Fault Recognition and Batch Size

This experiment keeps the number of LSTM units and the activation unit at a constant rate, while the single-batch processing data size gradually increases, and improves the number of traversals in the same batch size condition. The number of LSTM training samples is 10,000, and the number of test samples is 3000; the number of LSTM units is 128; and the activation unit is *sigmoid*. The relationship between the accuracy rate and the different batch sizes is shown in Table 7, and its trend is shown in Figure 15.

Table 7. Accuracy of fault recognition under different batch sizes.

Epoch (Traversal Times)	Batch Size: 10	Batch Size: 20	Batch Size: 50
1	0.31189	0.47369	0.31947
5	0.39451	0.49687	0.37669
10	0.50321	0.51476	0.40712
15	0.50147	0.55684	0.43964
20	0.53697	0.59548	0.48585
50	0.55876	0.64587	0.51058

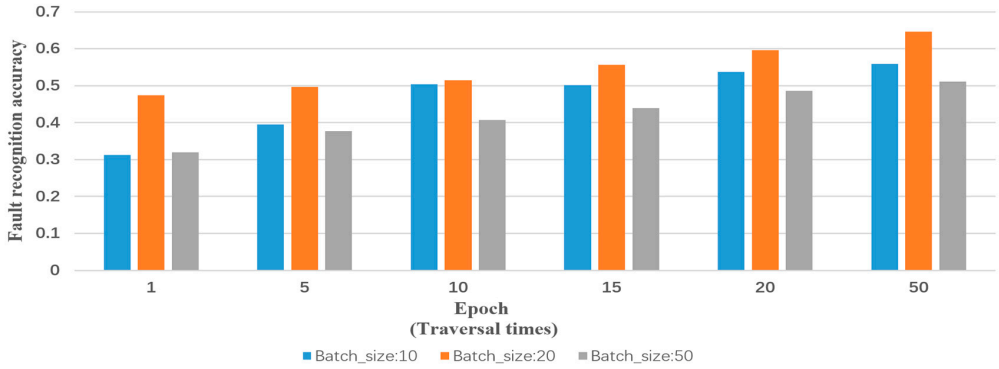


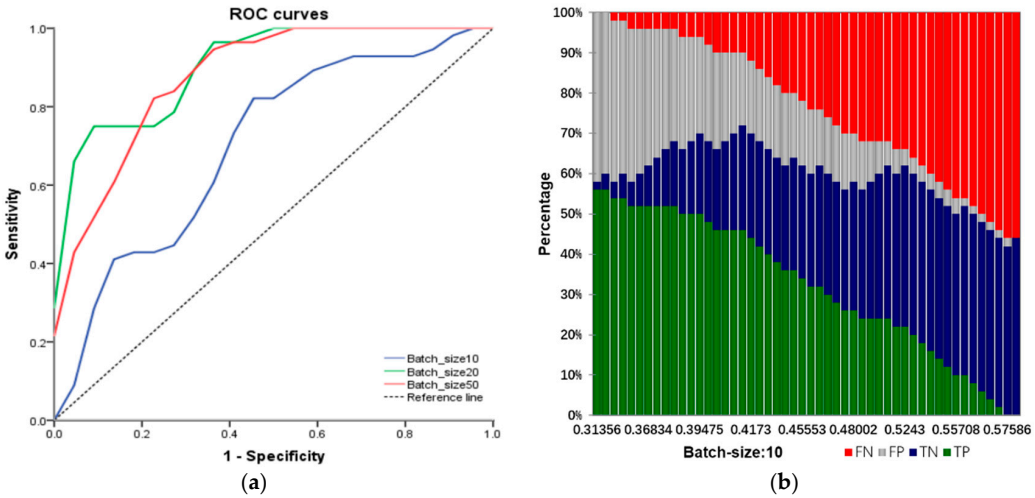
Figure 15. Accuracy of fault recognition under different batch size.

As can be observed from Table 7 and Figure 15, under the same batch size conditions, with the increase of the number of traversals, the fault recognition accuracy is higher. With the same number of traversals, when the batch size was valued at 20, the accuracy was higher than with the other two sizes. When the batch size was valued 10, the accuracy rate increased with the increase of the number of traversals, but the lack of continuous improvement was an under-fitting state. When the batch size was valued at 50, the accuracy rate was significantly decreased compared to the previous two sizes, as too much data in each batch processing caused an over-fitting phenomenon.

We also selected the above three batch sizes for ROC analysis by repeated experiments under different epoch conditions. In the following experiment, the threshold was set to 0.48. The test was positive if the accuracy of the fault recognition under different batch sizes was higher than the threshold value. As can be seen from Table 8 and Figure 16, when the batch size was valued at 20, it performed best. However, when the batch size was valued at 50, the overall ROC curve tended to be smoother. It should be noted that, for different datasets showing different characteristics, the batch size should not have a fixed range of selection. As the inspection report requires a certain word length that can express the corresponding characteristics, the best value for the batch size is 20.

Table 8. AUC analysis of ROC curves under different batch sizes.

Batch Size	AUC	Standard Error	Lower Bound (95%)	Upper Bound (95%)
10	0.6996	0.0773	0.5481	0.8512
20	0.9107	0.0386	0.8349	0.9865
50	0.8814	0.0477	0.7878	0.9751



(**a**) (**b**)

Figure 16. *Cont.*

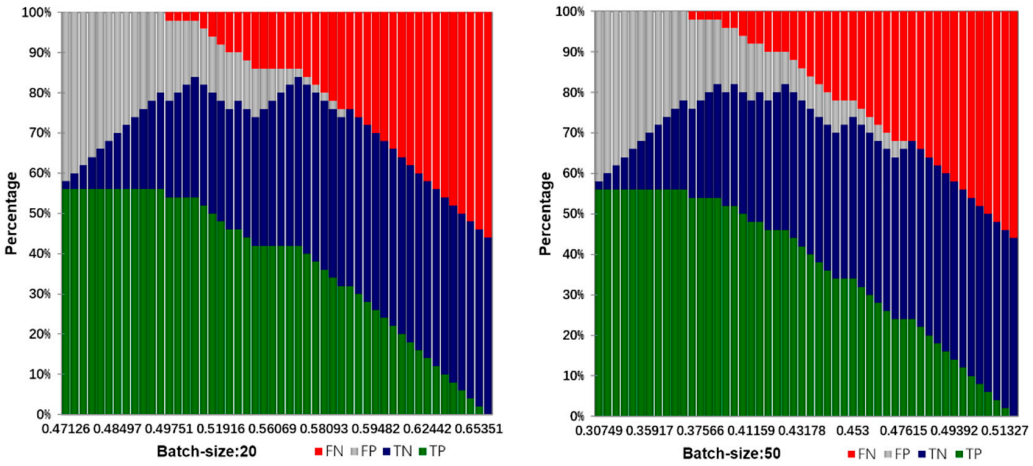


Figure 16. (**a**) Comparison of ROC curves under different batch sizes. The percentage of FN/FP/TN/TP under batch size (**b**) 10; (**c**) 20; and (**d**) 50.

In this paper, three key parameters are selected as experimental variables, and the experimental results can be reflected: the unstructured data processing method based on RNN-LSTM proposed in this paper can achieve the current research level in machine learning when it is applied to malfunction inspection reports. This means that this method can provide a more effective way for grid inspectors to deal with unstructured text.

6. Conclusions

How to efficiently handle large numbers of unstructured text data such as malfunction inspection reports is a long-standing problem faced by operation engineers. This paper proposes a deep learning method for malfunction inspection report processing by using the text data mining–oriented RNN-LSTM. An effective training strategy for an effective RNN-LSTM network for modeling inspection data is presented. From the obtained results, and an effectiveness analysis, it was demonstrated that RNN-LSTM, especially with target replication, can successfully classify diagnoses of labeled malfunction inspection reports given unstructured data. It is emphasized that the experiment via different variables can be reflected in the configuration of key variables in how we should select parameters to achieve optimal results, including the selection of the maximum LSTM unit number, the processing capacity of the activation unit and the prevention of over-fitting. In this paper, we used fault labels without timestamps, but we are obtaining timestamped diagnoses, which will enable us to train models to perform early fault diagnosis by predicting future conditions in a larger inspection data set.

Acknowledgments: The authors are grateful for the projects supported by the National Natural Science Foundation of China No. 51477121 and the China Southern Power Grid No. GZ2014-2-0049.

Author Contributions: Daqian Wei, Gang Lin and Bo Wang designed the main parts of the study, including RNN-LSTM modeling and the implementation of the algorithms, the neural network training process and the experiments. Daqian Wei, Gang Lin and Hesen Liu mainly contributed to the writing of the paper. Yilu Liu was responsible for guidance, a number of key suggestions, and manuscript editing. Zhaoyang Dong and Dichen Liu were also responsible for some constructive suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The Detailed Description of Recurrent Neural Network

Appendix A.1. Forward Propagation

$$u = W_{xh} \times x \tag{A1}$$

$$h_t = tanh(W_{hh} \times h_{t-1} + u) = tanh(z_t + u) \tag{A2}$$

$$u' = W_{hy} \times h \tag{A3}$$

$$y = u' \tag{A4}$$

where u is the input of the hidden layer, h is the output of the hidden layer, u' is the input of the output layer, y is the input of the input layer, and because the output layer is linear, $y = u'$. Specific calculation of each neuron is as follows:

$$u_i = \sum_{j=1}^{V} W_{xh}(i,j) \times x_j \tag{A5}$$

$$z_i^t = \sum_{j=1}^{H} W_{hh}(i,j) \times h_j^{t-1} \tag{A6}$$

$$h_i = tanh\left(u_i + z_i^t\right) \tag{A7}$$

$$u_i{}' = \sum_{j=1}^{H} W_{hy}(i,j) \times h_j \tag{A8}$$

$$y_i = u_i{}' \tag{A9}$$

Appendix A.2. Loss Function

There are two kinds of loss function: quadratic error function and cross-entropy error function. Assuming that t is the true value of the training sample, y is the output of the neural network, and a training sample is (x, t).

(1) Quadratic error function

$$E(t,y) = \frac{1}{2}(t-y)^2 \tag{A10}$$

(2) Cross-entropy error function

$$E(t,y) = -[t\ln(y) + (1-t)\ln(1-y)] \tag{A11}$$

When the activation function does not adopt through the output layer of the neural network, the quadratic error function should be used, which can give relatively fast gradient parameter estimation. If the output layer uses the *sigmoid* activation function, we use the cross-entropy error function to estimate the parameters. As long as the *sigmoid* activation function is selected, the cross-entropy error function can be used to eliminate the *sigmoid* function at the time of derivation, which can speed up the gradient descent.

(3) Partial derivation of error to the output of the output layer

(a) Quadratic error function

$$\frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t-y)^2 = y - t \tag{A12}$$

(b) Cross-entropy error function

$$\begin{array}{l} \frac{\partial E}{\partial y} = \frac{\partial}{\partial y}(-[t\ln(y) + (1-t)\ln(1-y)]) = \frac{-t}{y} + \frac{1-t}{1-y} \\ = \frac{-t(1-y)+y(1-t)}{y(1-y)} = \frac{y-t}{y(1-y)} \end{array} \tag{A13}$$

(4) Partial derivation of error to the input of the output layer

(a) Linear output layer

If the output layer is without an activation function, the different cost functions are solved separately: $\frac{\partial E}{\partial u'} = \frac{\partial E}{\partial y}$.

(b) *Sigmoid* output layer

Calculate the derivative of the error on input to the output layer, z denotes the input to the output layer, $z = u'$, $y = sigmoid(z)$. In the neural network, the error of the output layer is defined as the derivative of the loss function to the input layer, denoted by δ^L.

Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$, where the partial derivative of the *sigmoid* function is: $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x))$.

$$\delta^L = \frac{\partial E}{\partial z} = \frac{\partial E}{\partial y} * \frac{\partial y}{\partial z} = \frac{y-t}{y(1-y)}\frac{\partial y}{\partial z} = \frac{y-t}{y(1-y)}\sigma(z)(1-\sigma(z)) = \frac{y-t}{y(1-y)}y(1-y) = y-t \quad \text{(A14)}$$

Appendix A.3. Back Propagation

(1) Partial derivation of the error to the input of the output layer.

$$\frac{\partial E}{\partial y_i} = \frac{\partial}{\partial y_i}\frac{1}{2}(t_i - y_i)^2 = y_i - t_i \quad \text{(A15)}$$

Matrix obtained:

$$\frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t-y)^2 = y-t \quad \text{(A16)}$$

(2) Partial derivation of the error to the output of the output layer

If the output layer does not use the activation function, then ${u_i}' = y_i$

$$\frac{\partial E}{\partial {u_i}'} = \frac{\partial E}{\partial y_i} = y_i - t_i \quad \text{(A17)}$$

Matrix obtained:

$$\frac{\partial E}{\partial u'} = \frac{\partial E}{\partial y} = y-t \quad \text{(A18)}$$

(3) Partial derivative of the error to W_{hy}

$$\begin{aligned} &\tfrac{\partial E}{\partial W_{hy}(i,j)} = \tfrac{\partial E}{\partial {u_i}'} * \tfrac{\partial {u_i}'}{\partial W_{hy}(i,j)} \\ &= (y_i - t_i) * \tfrac{\partial}{\partial W_{hy}(i,j)}\left(\sum_{j=1}^{H} W_{hy}(i,j) * h_j\right) \\ &= (y_i - t_i) * h_j \end{aligned} \quad \text{(A19)}$$

Matrix obtained:

$$\frac{\partial E}{\partial W_{hy}} = \frac{\partial E}{\partial y} * H^T \quad \text{(A20)}$$

(4) Partial derivation of the error to the hidden layer output

Since h is affected by the following two Equations (A21) and (A22), when calculating the partial derivative of the loss function to the hidden layer output, it is necessary to compute the partial derivatives of h with respect to both formulas.

$$h_i^{t+1} = tanh\left(u_i^t + z_i^t\right), z_i^t = \sum_{j=1}^{H} W_{hh}(i,j) * h_j^{t-1}, u_i = \sum_{j=1}^{V} W_{xh}(i,j) * x_j \quad \text{(A21)}$$

$$u_i{}' = \sum_{j=1}^{H} W_{hh}(i,j) * h_j \tag{A22}$$

The matrix of Equation (A21): $u_i^{t+1} = u_i^t + z_i^t$, $u^{t+1} = W_{hh} * h^t + W_{hx} * x$, d_{next} is defined as the partial derivative of the error to the hidden layer input at the next moment.

$$\begin{aligned} \frac{\partial E}{\partial h_i} &= \sum_{k=1}^{V} \frac{\partial E}{\partial u_k{}'} * \frac{\partial u_k{}'}{\partial h_i} = \sum_{k=1}^{V} \frac{\partial E}{\partial y_k} * \frac{\partial u_k{}'}{\partial h_i} \\ &= \sum_{k=1}^{V} (y_k - t_k) * \frac{\partial u_k{}'}{\partial h_i} = \sum_{k=1}^{V} (y_k - t_k) * W_{hy}(k,i) \end{aligned} \tag{A23}$$

$$\begin{aligned} dh_{next} &= \frac{\partial E}{\partial h_i^t} = \sum_{j=1}^{H} \frac{\partial E}{\partial u_j^{t+1}} \frac{\partial u_j^{t+1}}{\partial h_i^t} \\ &= \sum_{j=1}^{H} \frac{\partial E}{\partial u_j^{t+1}} \frac{\partial}{\partial h_i^t} \left(\sum_{i=1}^{H} W_{hh}(j,i) * h_i^t + \sum_{i=1}^{V} W_{xh}(j,i) * x_i \right) \\ &= \sum_{j=1}^{H} \frac{\partial E}{\partial u_j^{t+1}} \left(\frac{\partial}{\partial h_i^t} \sum_{i=1}^{H} W_{hh}(j,i) * h_i^t + \frac{\partial}{\partial h_i^t} \sum_{i=1}^{V} W_{xh}(j,i) * x_i \right) \\ &= \sum_{j=1}^{H} \frac{\partial E}{\partial u_j^{t+1}} \left(\frac{\partial}{\partial h_i^t} \sum_{i=1}^{H} W_{hh}(j,i) * h_i^t + 0 \right) \\ &= \sum_{j=1}^{H} \frac{\partial E}{\partial u_j^{t+1}} (W_{hh}(j,i) + 0) = \sum_{j=1}^{H} \frac{\partial E}{\partial u_j^{t+1}} W_{hh}(j,i) \end{aligned} \tag{A24}$$

Matrix obtained:

$$dh_{next} = W_{hh}^T \frac{\partial E}{\partial u} \tag{A25}$$

$$\frac{\partial E}{\partial h} = W_{hy}^T * \frac{\partial E}{\partial u'} + dh_{next} \tag{A26}$$

$$\frac{\partial E}{\partial h} = W_{hy}^T * \frac{\partial E}{\partial y} + dh_{next} \tag{A27}$$

(5) Partial derivation of the loss function to the hidden layer input

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial u_i} = \frac{\partial E}{\partial h_i} \frac{\partial}{\partial u_i} (tanh(u_i)) = \frac{\partial E}{\partial h_i} \left(1 - \left(tanh(u_i)^2 \right) \right) \tag{A28}$$

Matrix obtained:

$$\frac{\partial E}{\partial u} = (1 - h \odot h) \frac{\partial E}{\partial h} \tag{A29}$$

(6) Partial derivation of the error to W_{xh}

$$\frac{\partial E}{\partial W_{xh}(i,j)} = \frac{\partial}{\partial u_i} \frac{\partial u_i}{\partial W_{xh}(i,j)} = \frac{\partial}{\partial u_i} \frac{\partial}{\partial W_{xh}(i,j)} \left(\sum_{j=1}^{V} W_{xh}(i,j) * x_j \right) = \frac{\partial}{\partial u_i} x_j \tag{A30}$$

As known $u_i = \sum_{j=1}^{V} W_{xh}(i,j) * x_j$, matrix obtained:

$$\frac{\partial E}{\partial W_{xh}} = \frac{\partial E}{\partial u} x^T \tag{A31}$$

(7) Partial derivation of the error to $W_{hh}(i,j)$

$$\begin{aligned}\frac{\partial E}{\partial W_{hh}(i,j)} &= \frac{\partial E}{\partial u_i}\frac{\partial u_i}{\partial W_{hh}(i,j)} \\ &= \frac{\partial}{\partial u_i^t}\frac{\partial}{\partial W_{hh}(i,j)}\left(\sum_{i=1}^{H} W_{hh}(j,i) * h_i^{t-1} + \sum_{i=1}^{V} W_{xh}(j,i) * x_i\right) \\ &= \frac{\partial}{\partial u_i^t}\left(\frac{\partial}{\partial W_{hh}(i,j)}\sum_{i=1}^{H} W_{hh}(j,i) * h_i^{t-1} + \frac{\partial}{\partial W_{hh}(i,j)}\sum_{i=1}^{V} W_{xh}(j,i) * x_i\right) \\ &= \frac{\partial}{\partial u_i^t}\left(\frac{\partial}{\partial W_{hh}(i,j)}\sum_{i=1}^{H} W_{hh}(j,i) * h_i^{t-1} + 0\right) = \frac{\partial}{\partial u_i^t} h_i^{t-1}\end{aligned} \tag{A32}$$

where $u_i^t = \sum_{i=1}^{H} W_{hh}(j,i) * h_i^{t-1} + \sum_{i=1}^{V} W_{xh}(j,i) * x_i$, matrix obtained:

$$\frac{\partial E}{\partial W_{hh}} = \frac{\partial E}{\partial u} * \left(h^{t-1}\right)^T \tag{A33}$$

By calculating the partial derivatives of all the parameter matrices with respect to the errors, the parameters can be updated according to the gradient descent of the partial derivatives.

$$\frac{\partial E}{\partial W_{hy}} = \frac{\partial E}{\partial y} * H^T \tag{A34}$$

$$\frac{\partial E}{\partial W_{xh}} = \frac{\partial E}{\partial u} x^T \tag{A35}$$

$$\frac{\partial E}{\partial W_{hh}} = \frac{\partial E}{\partial u} * \left(h^{t-1}\right)^T \tag{A36}$$

$$\frac{\partial E}{\partial u} = (1 - h \odot h)\frac{\partial E}{\partial h} \tag{A37}$$

$$\frac{\partial E}{\partial h} = W_{hy}^T * \frac{\partial E}{\partial y} + dh_{next} \tag{A38}$$

$$dh_{next} = W_{hh}^T \frac{\partial E}{\partial u} \tag{A39}$$

Appendix B. The Process of the Training Method for RNN-LSTM

The forward pass process of the training method is shown in Table A1.

Table A1. Forward pass process.

Process 1: Forward Pass
input units: y = current external input;
roll over: activation: $\hat{y} = y$; cell state: $\hat{s}_{c_j^v} = s_{c_j^v}$;
Loop over memory blocks, indexed j
Step 1a: input gates (1):
$net_{in_j} = \sum_m w_{in_j m}\hat{y}^m + \sum_{v=1}^{S_j} w_{in_j c_j^v}\hat{s}_{c_j^v}$; $y^{in_j} = f_{in_j}\left(net_{in_j}\right)$;
Step 1b: forget gate (2):
$net_{\varphi j} = \sum_m w_{\varphi_j m}\hat{y}^m + \sum_{v=1}^{S_j} w_{\varphi_j c_j^v}\hat{s}_{c_j}^v$; $y^{\varphi_j} = f_{\varphi_j}\left(net_{\varphi_j}\right)$;
Step 1c: the cell states (3):
Loop over the S_j cells in blocks j, index v
$\{net_{c_j^v} = \sum_m w_{c_j^v m}\hat{y}^m$; $s_{c_j^v} = y^{\varphi_j}\hat{s}_{c_j^v} + y^{in_j} g\left(net_{c_j^v}\right)\}$;
Step 2:
Output gate activation (4):
$net_{out_j} = \sum_m w_{out_j m}\hat{y}^m + \sum_{v=1}^{S_j} w_{out_j c_j^v} s_{c_j^v}$; $y^{out_j} = f_{out_j}\left(net_{out_j}\right)$;
Cell outputs (5):
Loop over the S_j cells in block j, indexed v
$\{y^{c_j^v} = y^{out_j} s_{c_j^v}\}$;
End loop over memory blocks
Output units (6): $net_k = \sum_m w_{km} y^m$; $y^k = f_k(net_k)$;

The partial derivatives process of the training method is shown in Table A2.

Table A2. Partial derivatives process.

Process 2: Partial Derivatives
Loop over memory blocks, index j
{Loop over the S_j cells in block j, indexed v
{**Cells**, $\left(dS_{cm}^{jv} := \frac{\partial s_{c_j^v}}{\partial w_{c_j^v m}}\right)$:
$dS_{cm}^{jv} = dS_{cm}^{jv} y^{\varphi j} + g'\left(net_{c_j^v}\right) y^{in_j} \hat{y}^m$;
Input gates, $dS_{in,m}^{jv} := \frac{\partial s_{c_j^v}}{\partial w_{in_j m}}$, $dS_{in,c_j^{v'}}^{jv} := \frac{\partial s_{c_j^v}}{\partial w_{in_j c_j^{v'}}}$;
$dS_{in,m}^{jv} = dS_{in,m}^{jv} y^{\varphi j} + g\left(net_{c_j^v}\right) f'_{in_j}\left(net_{in_j}\right) \hat{y}^m$;
Loop over peephole connections from all cells, indexed v'
$\{dS_{in,c_j^{v'}}^{jv} = dS_{in,c_j^{v'}}^{jv} y^{\varphi j} + g(net_{c_j^v}) f'_{in_j}(net_{in_j}) \hat{s}_c^{v'};\}$;
Forget gates, $(dS_{\varphi m}^{jv} := \frac{\partial s_{c_j^v}}{\partial w_{\varphi_j m}}, dS_{\varphi c_j^{v'}}^{jv} := \frac{\partial s_{c_j^v}}{\partial w_{\varphi_j c_j^{v'}}})$
$dS_{\varphi m}^{jv} = dS_{\varphi m}^{jv} y^{\varphi j} + \hat{s}_{c_j^v} f'_{\varphi j}\left(net_{\varphi j}\right) \hat{y}^m$;
Loop over peephole connections from all cells, indexed v'
$\{dS_{\varphi c_j^{v'}}^{jv} = dS_{\varphi c_j^{v'}}^{jv} y^{\varphi j} + \hat{s}_{c_j^v} f'_{\varphi j}\left(net_{\varphi j}\right) \hat{s}_c^{v'};\}\}$
End loops over cells and memory blocks

The backward pass process of the training method is shown in Table A3.

Table A3. Backward pass process.

Process 3: Backward Pass (If Error Injected)
Errors and δ_S:
Injected error: $e_k = t^k - y^k$;
δ_S **of output units:** $\delta_k = f'_k(net_k) e_k$;
Loop over memory blocks, indexed j
{δ_S **of output gates:**
$\delta_{out_j} = f'_{out_j}\left(net_{out_j}\right)\left(\sum_{v=1}^{S_j} s_{c_j^v} \sum_k w_{kc_j^v} \delta_k\right)$;
Internal state error:
Loop over the S_j cells in blocks j, indexed v
$\{e_{s_{c_j^v}} = y^{out_j}\left(\sum_k w_{kc_j^v} \delta_k\right);\}\}$
End loop over memory blocks

The weight updates process of the training method is shown in Table A4.

Table A4. Weight updates process.

Process 4: Weight Updates
Output units: $\Delta w_{km} = \alpha \delta_k y^m$;
Loop over memory blocks, indexed j
{**Output gates:**
$\Delta w_{out,m} = \alpha \delta_{out} \hat{y}^m$; $\Delta w_{out,c_j^v} = \alpha \delta_{out} s_{c_j^v}$;
Input gates:
$\Delta w_{in,\mathrm{m}} = \alpha \sum_{v=1}^{S_j} e_{s_{c_j^v}} dS_{in,\mathrm{m}}^{jv}$;
Loop over peephole connections from all cells, indexed v'
$\{\Delta w_{in,c_j^{v'}} = \alpha \sum_{v=1}^{S_j} e_{s_{c_j^v}} dS_{in,c_j^{v'}}^{jv};\}$
Forget gates:
$\Delta w_{\varphi m} = \alpha \sum_{v=1}^{S_j} e_{s_{c_j^v}} dS_{\varphi m}^{jv}$;
Loop over peephole connections from all cells, indexed v'
$\{\Delta w_{\varphi c_j^{v'}} = \alpha \sum_{v=1}^{S_j} e_{s_{c_j^v}} dS_{\varphi c_j^{v'}}^{jv};\}$
Cells:
Loop over the S_j cells in block j, indexed v
$\{\Delta w_{c_j^v m} = \alpha e_{s_{c_j^v}} dS_{cm}^{jv};\}\}$
End loop over memory blocks

References

1. Besnard, F.; Bertling, L. An approach for condition-based maintenance optimization applied to wind turbine blades. *IEEE Trans. Sustain. Energy* **2010**, *1*, 77–83. [CrossRef]
2. Ahmad, R.; Kamaruddin, S. An overview of time-based and condition-based maintenance in industrial application. *Comput. Ind. Eng.* **2012**, *63*, 135–149. [CrossRef]
3. Carita, A.J.Q.; Leita, L.C.; Junior, A.P.P.M.; Godoy, R.B.; Sauer, L. Bayesian networks applied to failure diagnosis in power transformer. *IEEE Lat. Am. Trans.* **2013**, *11*, 1075–1082.
4. Su, H.; Li, Q. A hybrid deterministic model based on rough set and fuzzy set and Bayesian optimal classifier. In Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC), Beijing, China, 30 August–1 September 2006; Volume 2, pp. 175–178.
5. Zheng, G.; Yongli, Z. Research of transformer fault diagnosis based on Bayesian network classifiers. In Proceedings of the International Conference Computer Design and Applications (ICCDA), Qinhuangdao, China, 25–27 June 2010; Volume 3, pp. 382–385.
6. Tang, W.H.; Spurgeon, K.; Wu, Q.H.; Richardson, Z.J. An evidential reasoning approach to transformer condition assessments. *IEEE Trans. Power Deliv.* **2004**, *19*, 1696–1703. [CrossRef]
7. Li, J.; Chen, X.; Wu, C. Power transformer state assessment based on grey target theory. In Proceedings of the International Conference Measuring Technology and Mechatronics Automation, Zhangjiajie, China, 11–12 April 2009; Volume 2, pp. 664–667.
8. Ma, H.; Ekanayake, C.; Saha, T.K. Power transformer fault diagnosis under measurement originated uncertainties. *IEEE Trans. Dielectr. Electr. Insul.* **2012**, *19*, 1982–1990. [CrossRef]
9. Shintemirov, A.; Tang, W.; Wu, Q.H. Power transformer fault classification based on dissolved gas analysis by implementing bootstrap and genetic programming. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2009**, *39*, 69–79. [CrossRef]
10. Koley, C.; Purkait, P.; Chakravorti, S. Wavelet-aided SVM tool for impulse fault identification in transformers. *IEEE Trans. Power Deliv.* **2006**, *21*, 1283–1290. [CrossRef]
11. Miranda, V.; Castro, A.R.G.; Lima, S. Diagnosing faults in power transformers with autoassociative neural networks and mean shift. *IEEE Trans. Power Deliv.* **2012**, *27*, 1350–1357. [CrossRef]
12. Wu, H.R.; Li, X.H.; Wu, D.N. RMP neural network based dissolved gas analyzer for fault diagnostic of oil-filled electrical equipment. *IEEE Trans. Dielectr. Electr. Insul.* **2011**, *18*, 495–498. [CrossRef]
13. Wang, M.H. A novel extension method for transformer fault diagnosis. *IEEE Trans. Power Deliv.* **2003**, *18*, 164–169. [CrossRef]
14. Reshmy, A.K.; Paulraj, D. An efficient unstructured big data analysis method for enhancing performance using machine learning algorithm. In Proceedings of the International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 19–20 March 2015; pp. 1–7.
15. Yuanhua, T.; Chaolin, Z.; Yici, M. Semantic presentation and fusion framework of unstructured data in smart cites. In Proceedings of the 10th IEEE Conference on Industrial Electronics and Applications (ICIEA 2015), Auckland, New Zealand, 5–17 June 2015; Volume 10, pp. 897–901.
16. Goth, G. Digging deeper into text mining: Academics and agencies look toward unstructured data. *IEEE Internet Comput.* **2012**, *16*, 7–9. [CrossRef]
17. Alifa, N.P.; Saiful, A.; Wikan, D.S. Public facilities recommendation system based on structured and unstructured data extraction from multi-channel data sources. In Proceedings of the International Conference on Data and Software Engineering (ICoDSE), Yogyakarta, Indonesia, 25–26 November 2015; pp. 185–190.
18. Huang, G.B.; Mattar, M.; Berg, T.; Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In Proceedings of the Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition, Marseille, France, 12–18 October 2008.
19. Chen, D.; Cao, X.; Wen, F.; Sun, J. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013; pp. 3025–3032.
20. Sun, Y.; Wang, X.; Tang, X. Deeply learned face representations are sparse, selective, and robust. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 2892–2900.

21. Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; Khudanpur, S. Recurrent neural network based language Model. In Proceedings of the Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010; pp. 1045–1048.
22. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. 2013. Available online: arxiv.org/abs/1301.3781 (accessed on 7 Sepember 2013).
23. Hochreiter, S. Untersuchungen zu Dynamischen Neuronalen Netzen. Master's Thesis, Technische Universität München, Munich, Germany, 1991.
24. Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. Gradient Flow in recurrent nets: The difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*; Kremer, S.C., Kolen, J.F., Eds.; IEEE Press: Piscataway, NJ, USA, 2001.
25. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef] [PubMed]
26. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
27. Graves, A.; Mohamed, A.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 26–31 May 2013.
28. Sak, H.; Senior, A.; Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), Singapore, 14–18 September 2014.
29. Tai, K.S.; Socher, R.; Manning, C.D. Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks. 2015. Available online: arxiv.org/abs/1503.00075 (accessed on 30 May 2015).
30. Li, J.; Luong, M.T.; Jurafsky, D. A Hierarchical Neural Autoencoder for Paragraphs and Documents. 2015. Available online: arxiv.org/abs/1506.01057 (accessed on 6 June 2015).
31. Chanen, A. Deep learning for extracting word-levesl meaning from safety report narratives. In Proceedings of the Integrated Communications Navigation and Surveillance (ICNS), Herndon, VA, USA, 19–21 April 2016; pp. 5D2-1–5D2-15.
32. Palangi, H.; Deng, L.; Shen, Y. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 694–707. [CrossRef]