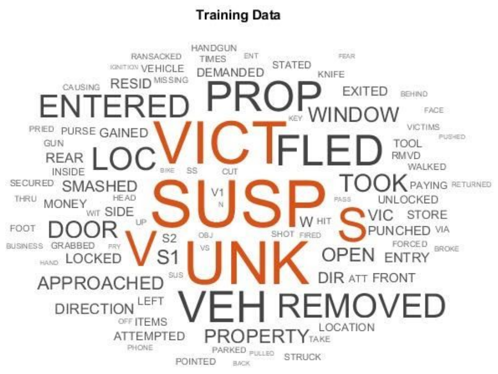
Crime Classification Application



Matthew Jackson
Computer Science
IUPUI
Thorntown, IN
Matajack@iu.edu

Giorgio Giudice
Computer Science
IUPUI
Indianapolis, IN
giudiceg@iu.edu

Nicolas Medina
Computer Science
IUPUI
Greenwood, IN
nlmedina@iu.edu

Dimitrije Prosevski
Computer Science
IUPUI
Brownsburg, IN
dprosev@iu.edu

Ujjaval Patel
Computer Science
IUPUI
Indianapolis, IN
ujpatel@iu.edu

Atul Singh
Computer Science
IUPUI
Fishers, IN
atsingh@iu.edu

Dhruv Patel
Computer Science
IUPUI
Indianapolis, IN
pateldhm@iu.edu

Jonathan Young
Computer Science
IUPUI
Indianapolis, IN
youngjon@iu.edu

***Abstract*—Classifying crimes based on police reports is a crucial part of our justice system. With this paper, we show our research exploring different applications of classifiers. This paper shows the evaluation of these models. We will also show our findings after testing a fully developed Naive Bayes Classifier and a Deep Learning Long-Term Short Memory Neural Network. The results from the two models gave us an opportunity to experience the weaknesses and strengths of these two radically different approaches.**

Keywords—crime, classification, modeling, data processing

I. Introduction

The 911 phone system program implemented in the United States handles 500,000 calls daily or 183 million yearly. This number is only increasing as access to wireless phones becomes readily available. An increased magnitude of these calls has the potential to exacerbate an already overwhelmed system. A significant amount of calls occur because of misdials or just hang up calls that are both intentional and on accident. The general populace also has a tendency to misdiagnose the severity of their problems and often call the emergency line for non-emergency purposes. Whether it be intentional calls, non-emergency 911 calls, prank calls, or exaggerated calls we want to build a system that is able to interpret the type of crime voiced in these calls and ultimately judge which avenue to direct them. This will not only direct the people's calls to the correct authorities but alleviate the situation of 911 responders on the front line. Our project attempts to implement this idea by first classifying the crimes of potential emergency situations. Ideally, our hope is that anytime a person makes a phone call and explains their situation, their reason for calling is automatically interpreted and placed in the correct category of the 911 responder's system.

II. Training and Testing

<u>**Data Pre-Processing and Testing**</u>

One of the most important aspects before creating any models is data preprocessing. If the quality of data is inferior then no model will be able to aptly gauge and produce accurate results. The libraries that were used to split the dataset and adapt it to be analyzed were TM and text2vec. The TM library is a text mining package that is used to transform incoming unstructured text into data fit for analysis or for use in machine learning algorithms. The main structure utilized in TM is a Corpus which is a collection of text documents. With the narrative data inserted into a Corpus, the data can now be modified using the tm_map function. With this function, the narrative dataset can be prepared for use in models by removing numeric characters, punctuation, and whitespace and changing the text to lowercase to have the narrative data containing only lowercase characters. With the dataset transformed, the data is now entered into an instance of the DocumentTermMatrix class. The DocumentTermMatrix class allows data to be analyzed through methods in the class. One such method used to analyze the narrative dataset is the findFreqTerms() function. By using the findFreqTerms() function, the narrative data can return terms that occur a set amount of times, further transforming the narrative dataset into a dataset that can be analyzed and entered in models.

The text2vec library is similar in use to the TM library as its another text mining package that provides a framework for text analysis. First, the create_vocabulary() function is used with the narrative dataset, already tokenized, as the parameter. This allows the narrative dataset to be revised through text2vec functions. Using the prune_vocabulary() function, the narrative dataset sheds any terms that are on the extremes in terms of frequency. With the data now modified, it is entered into the vocab_vectorizer() function, preparing it to be transformed into a vector space. At this stage, the data was made into a document-term matrix using create_dtm, making it ready to use for text analysis.

To learn more about the data itself we examined the distribution of the crime type. Figure 1 below visualizes our distribution. We notice several interesting features. Aggravated Assault class contained the highest length of characters of the crime narrative while Theft has the least amount. From this, we can gather that there seems to be more text for crimes like Agg, BTFV, and Burglary. In comparison to that of Theft, Robbery, and GTA.

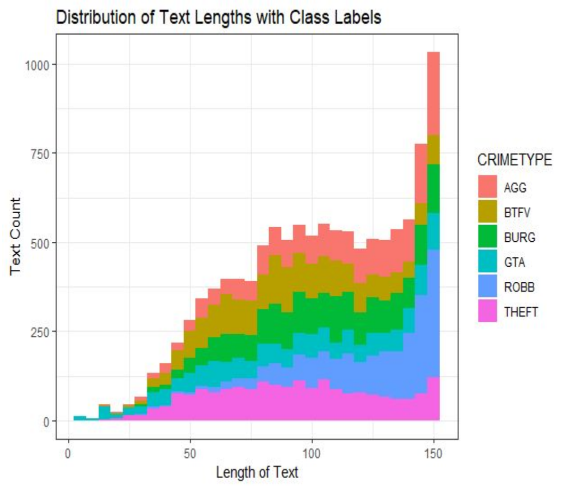


Figure 1. Distribution of Text Lengths with Class Labels

Cross-validation

Using cross-validation on the program's models is important in determining the accuracy of the models. The train_test split and k-fold approach are a couple of common techniques used for cross-validation. K-fold is one of the better methods in achieving cross-validation as it results in a less biased model. However, for our project, we utilized the train_test split approach to ensure our models did not under- or over-fit the data. The train_test split approach is done by randomly splitting the complete data into training and test sets. The models use the training set to train and then are tested on the test set.

Procedure

For each of the models that were created, we followed a similar consistent procedure. Once we had our dataset, we split our data into two-halves to compose the training and testing set. The next step involved learning about and developing multiple models in order to build it for classification. The models were used to test a plethora of inputs to determine the correctness of the result.

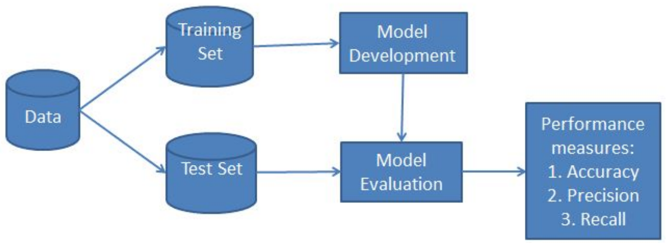


Figure 2. Classification Procedure

III. Models

Generalized Linear Model:

GLM is the generalized linear model. The class of GLM includes models such as Poisson regression and linear regression. Every GLM consists of three components. These components of a GLM are the linear predictor, link function, and probability distribution. In the Poisson model, the linear predictor involves some variable that explains a parameter within the model. While the link function is the function that associates the linear predictor with the parameter in the model. And the probability distribution is what displays the internal variable y. For the linear regression model, there is only a link function since the internal variable for the probability distribution is the same as the linear predictor [4]. When using the logit function as the link function combined with a binomial distribution this creates a logistic regression model: $\ln(p/(1-p))=b_0+ b_1x$. This model predicts the probability of an outcome with only two possible endings. Here is an example figure:

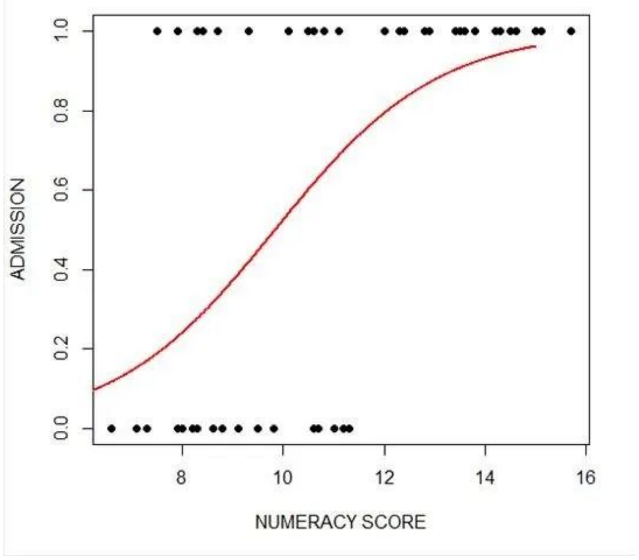


Figure 3. Logistic Regression Plot

This allows our model to connect our predictors to our target and this function is run repeatedly until there is no significant change to the model. The goal is for it to maximize a given likelihood function: $L_n(\theta)=L_n(\theta;y)=f_n(y;\theta)$

This is done by attempting to find the parameters that maximize this likelihood function. This is accomplished through the following example: $\theta = \arg_{\theta\in\Theta}\max L_n(\theta;y)$. However, these models are not a good fit for our project.

Random Forest Classification:

The Random Forest model can be used for two purposes; RF for Classification and RF for Regression, but for the context of our research we are using the random forest for classification of crime. Thus, Random forest is another classification algorithm just like GLM and GBM models, but with few more modifications which makes the random forest a more accurate and efficient algorithm.

Like the name random forest, the algorithm is a set of many uncorrelated decision trees. Being an ensemble learning algorithm, RF takes a random subset of the training data set and assigns it to decision trees. As shown in figure 4, each tree works as an individual model and generates a class of prediction. Here the prediction is either 0 or 1. The resulting prediction with the most votes becomes the entire model's final prediction. As the figure shows six subtrees are predicting 1, hence 1 is the entire model's final prediction.

In our case, the data we are using here contains crimes and a detailed description explaining crimes. The RF models will take 70% of the original dataset and will randomly break it into multiple subsets. Now, using these subsets it generates multiple instances of decision trees. Each instance of the binary tree will generate a prediction class and a committee of these decision trees will determine the result based on the most predicted result. After finalizing the prediction class, the remaining 30% of the test data set is used to check the accuracy of the model based on the capacity of prediction crimes right. The error rate of the RF mode is lower than all other classification models. The reason why the random forest is so accurate is that many decision trees together protect each other from individual errors. In some cases, some decision trees predict totally wrong crimes but on the other side many more will predict the right crimes, and this reduces the overall error rate to almost zero, predicting the right crimes almost every time. The efficiency of the RF models also depends on the correlation among the generated decision trees. The lower the correlation, the higher the efficiency. Basically, Random forest works on an idea of 'the wisdom of crowds.' In the context of computer science, it means that a very large number of relatively uncorrelated trees working as a committee will outperform any other kind of individual model.

Now, as we know every algorithm or model comes with advantages and disadvantages. Let us first look at the advantages of using RF and later we will discuss the disadvantages of RF as a classifier.

Advantages of Random Forest as Classification Model:

- One of the most accurate learning algorithms and predicts a highly accurate result for real-life datasets.
- Efficient with large datasets.
- Works with thousands of input variables without any variable deletion.
- Provide an estimation of important variables in classification.

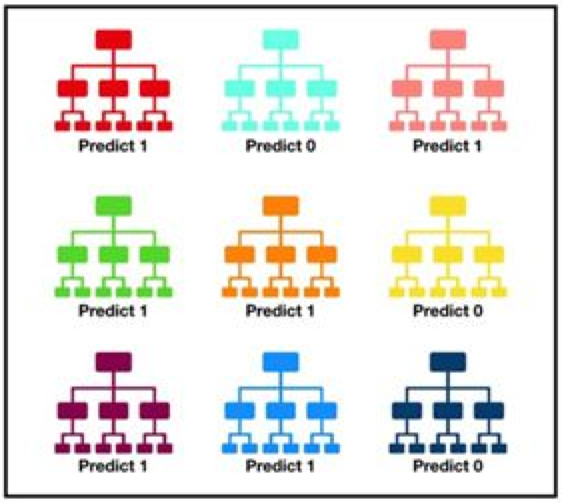


Figure 4. Classification Procedure

- Efficient in estimating data, thus works accurately even though a large portion of data is missing.
- Unbiased estimation of generalization error through forest building processes.

- One of the most accurate learning algorithms and predicts a highly accurate result for real-life datasets**.**

Disadvantages of Random Forest as Classification Model:

- The complexity of RF models, they are much harder and time-consuming than decision trees.
- It requires more memory and takes a lot of computation cost training a large number of decision trees for large datasets.
- Take more time to predict results than any other algorithms. Hence, Time-consuming RF models are challenging for some applications.

Naive Bayes Classification:

When it comes to text classification, the multinomial Naive Bayes classifier is one of the most widely used classifiers and faster relative to other classification algorithms. A Naive Bayes classifier is a probabilistic machine learning model based on Bayes' theorem with an assumption of independence amongst predictors. The requirement for a Naive Bayes classifier is to have an existing set of examples for each category (class/label) that we wish to classify words (features) into. For example, if a crime includes a word "car" our classifier will predict that out of all five possibilities (BTFV, Robbery, Grand Theft Auto, Theft, Aggravated Assault, BURG) that the crime is most likely to be "Grand Theft Auto" because it was trained that way in the training set.

Let us explain the theoretical steps; the First step of the process is to compute the Term-Document Matrix (TDM) that consists of a list of word frequencies in the set of narratives . TDM is a sparse rectangular matrix of "n" words and "m" narratives. Each entry has its "i" and "j" position in the matrix, where "i" represents the frequency of the word in the "j" narrative. Once frequencies are computed, the next step is to calculate occurrences (percentage of time each word appears in all the narratives). The third step, to recall, Bayes theorem formula is:

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}$$

which translates to posterior = (likelihood x prior)/evidence. The "Naive" Bayes classifier makes an assumption that the probability of each word is independent of each other. Using the example with "car" and "Grand Theft Auto" (GTA) again, we can write the formula as following

$$P(GTA \mid car) = \frac{P(car \mid GTA)\,P(GTA)}{P(car)} .$$

The formal decision rule, and the last step, is

$$y = \text{argmax } p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k),\ k \in \{\text{all classes}\}.$$

which means pick the most probable hypothesis. So, in this case, assuming that the word "car" determines "GTA" means that y_{GTA} will have the highest probability.

Now that we are clear with the theory behind the process, let us explain the coding process. First off, we imported three libraries from sklearn: CountVectorizer, TfidTransformer, MultinomialNB. CountVectorizer creates the TDM. Where the 2D vector columns are the frequencies of the words, and rows are individual narratives. TfidfTransformer, where "Tfidf" stands for term-frequency times inverse document-frequency. TfidTransformer is a commonly used term weighting scheme whose goal is to scale down the impact of words that appear very frequently in the given narrative but are in fact less informative than the features that occur in the training set, MultinomialNB stands for multinomial Naive Bayes function that takes integers as parameters.

Therefore, to classify a certain narrative successfully we have to train the model first. Training is executed in the following order: take the Narrative column of the dataset, call CountVectorizer function on it to transform it into TDM, then call TfidfTransformer function on TDM to scale down the impact of an uninformative word to get TDM_{tfidf}, and finally we call the MultinomialNB and pass our TDM_{tfidf} as a parameter to fit it onto the CrimeType column. Now that the model is trained, we can predict the crime type of any narrative by simply calling the .predict("someNarrative") method.

Naive Bayes has many advantages but there are some disadvantages as well. The foremost benefit of using this algorithm is that they are fast and easy to implement. When the assumptions of features in a class being independent holds true the classifier performs better than other models such as logistic regression which needs more training data. Performance for categorical input variables, as in the case of our project, works much better as compared to that of numerical variables. But, this also leads to the disadvantages as well. Our training data can not hold true for all categorical variables. If the training data set has not observed the feature, the models will assign a 0 probability and will be unable to make the correct predictions. Another big factor to consider also is that the requirement of predictions to be independent which is contrary to real life, where it is almost impossible to get a set of predictors that are independent, hindering the performance of the classification model.

IV. Neural Networks

When we talk about Neural Networks in Computer Science, we are really referring to Artificial Neural Networks. The underlying idea is to simulate a neural network that normally operates in human brains. Since the 80's many different approaches and technologies have been employed to enhance the ability of computerized "deep learning".

There are a few approaches we have evaluated. One of them was employing a Feedforward Neural Network, which is probably the simplest one when it comes to describing its architecture. In fact, while in other types of neural networks the data "bounces" back and forth among two nodes, this kind of NN does not have any kind of backpropagation. Another option we had was using a Convolutional Neural Network. That kind of architecture employs several layers that have multiple filters each. This would have been a great choice if we had to process images since those layers are very proficient in recognizing patterns in pictures. However, in our case, we needed a slimmer and more flexible architecture since our data set was text-based.

<u>Self-Organizing Feature Map</u>

A SOFM is a Neural Network that can be trained utilizing unsupervised learning to create typically a two-dimensional map (low-dimensional). This is a result of the discretized representation of the input space of the training samples, which will become our map, and is therefore considered to be a variation to dimensionality reduction. Furthermore, SOFM branches from the generic ANN as they do not implement error-correction to learn; rather, they take a more competitive approach. They do this by having nodes competing for the right to respond to a subset of the given data. This is a form derived from Hebbian Theory, which is why SOFM is often associated with the synaptic responses of the brain.

The goal of this application regarding SOFM, unlike most alternatives, is to eliminate the biases created by supervised models. These problems can arise in areas such as overusing certain keywords in association with one factor on the training set to create a network that does not accurately represent the true relationship for its use in the indication of the crime. This kind of error is less frequent in SOFM. Ergo, we do not require error-correction like other more prevalently used network types.

<u>Long Short-Term Memory (LSTM) Network</u>

This is why we tried to classify our data set using a deep learning long short-term memory (LSTM) network which is a type of recurrent neural network. The difference between this architecture and the standard neural network architecture is that "Recurrent networks…. have an internal state that can represent context information. … [they] keep information about past inputs for an amount of time that is not fixed a priori, but rather depends on its weights and on the input data"[11].

In our attempt, we have decided to implement one using MATLAB. First, we checked the class distribution to see if there was a crime that did not have as many narratives as the others. Please see Figure 5.

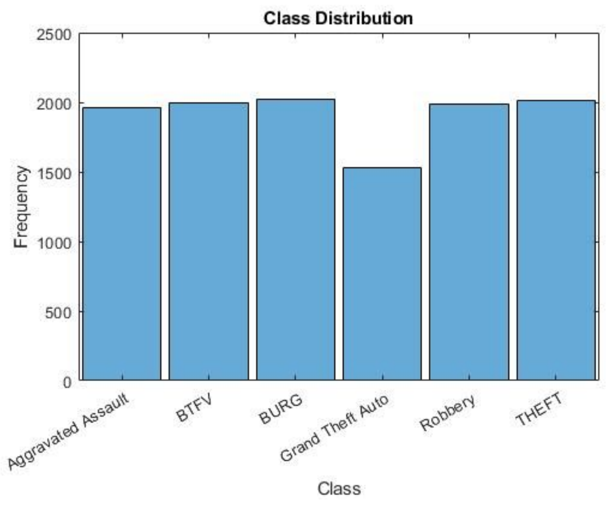
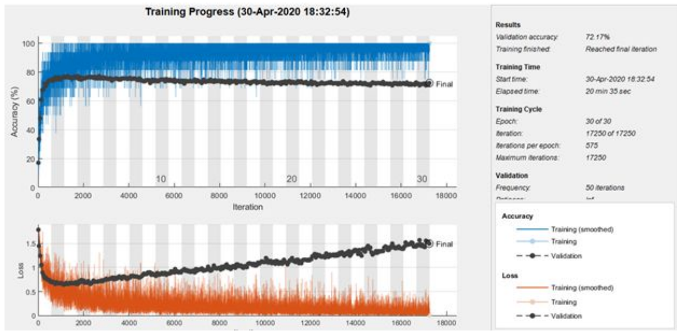


Figure 5. Class Distribution for crimes and their frequency

The distribution is quite even except for "Grand Theft Auto" which is slightly lower than the others.

We then proceeded with the training (figure 6).



Results
Validation accuracy: 72.17%
Training finished: Reached final iteration
Training Time
Start time: 30-Apr-2020 18:32:54
Elapsed time: 20 min 35 sec
Training Cycle
Epoch: 30 of 30
Iteration: 17250 of 17250
Iterations per epoch: 575
Maximum iterations: 17250
Validation
Frequency: 50 iterations

Figure 6. Training of LSTM Network

The Validation Accuracy was 72.17%, and we chose to have 30 epochs with 50 iterations. Considering that we had over 11000 entries, the training process only took 20 minutes and 35 seconds.

It is to be noted that there was a considerable loss halfway through the training. This was to be expected since the data was very repetitive and not diverse enough to avoid overfitting.

We then decided to test it and compare the result with the Naive Bayes Classifier. Please see the following spreadsheet to see the results of our test.

DESCRIPTION	NB	LSTMN
the man took my car	BTFV	GTA
the lady broke the windshield of the car and ran away	ROB	BTFV
the kid punched his father	AA	GTA
the 10 year old kid pointed the gun at me and took my money	ROB	AA
10 year old kid entered the house pointed the gun and took my money	ROB	AA
the man entered the shop and took the money	ROB	ROB
atul took my wallet	ROB	ROB
A girl broke guy’s heart	BTFV	GTA
The lady broke into my apartment	BURG	GTA
My mom ate my food	AA	GTA
his daughter stole her grandfather's watch	THEFT	THEFT
A man stole a Lambo	GTA	GTA
Jacob disposed of his wife	AA	GTA
Steve drove over his grandma	AA	AA
Marcia got her wallet stolen	GTA	GTA
Patrick drunk Marlie’s drink	AA	AA
Gabe was pushed into a wall	AA	BURG
George groped his coworker	ROB	AA
the woman slapped the kid	AA	AA
the man punched his grandpa	ROB	GTA
the kid headbutted his coach	AA	GTA
Matthew set my hair on fire	AA	AA
Luke hit me on my face	ROB	AA
Martina sneaked two apples out of the store	THEFT	THEFT
Giorgio stole pizza from the market	THEFT	ROB

V. Conclusion

After numerous testing, we arrived at the conclusion that Naive Bayes and Long Short Term Memory Network were most accurate in classifying the given narratives. The accuracy of Naive Bayes came out to 85% while the Long-Short Term Memory Network returned a validation accuracy of 72.17%. When tested with the description on the left column, the Naive Bayes has correctly classified the crime 60% of the times, outperforming the LSTM that was able to return a correct output only 44% of the times. We believe that with a larger data set both these architectures could potentially return more accurate output.

VI. References

[1] Sanjay.M. “Why and How to Cross Validate a Model?” Medium, Towards Data Science, 13 Nov. 2018, towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f.

[2] Feinerer, Ingo. Introduction to the Tm Package Text Mining in R. 12 Dec. 2019, cran.r-project.org/web/packages/tm/vignettes/tm.pdf.

[3] Selivanov, Dmitriy. Package ‘text2vec.’ 18 Feb. 2020, cran.r-project.org/web/packages/text2vec/text2vec.pdf.

[4] Generalized Linear Models Yuho Kida - https://towardsdatascience.com/generalized-linear-models-9cbf848bb8ab

[5] Chávez, Gustavo. “Implementing a Naive Bayes Classifier for Text Categorization in Five Steps.” Medium, Towards Data Science, 28 Feb. 2019, towardsdatascience.com/implementing-a-naive-bayes-classifier-for-text-categorization-in-five-steps-f9192cdd54c3.

[6] “Learn: Machine Learning in Python - Scikit-Learn 0.16.1 Documentation.” Scikit, scikit-learn.org/.

[7] Wyatt, Jeremy. “Kohonen Networks .” Kohonen Networks, University of Birmingham, 7 July 1997, www.cs.bham.ac.uk/~jlw/sem2a2/Web/Kohonen.htm

[8] Chakure, Afroz. “Random Forest Classification and Its Implementation.” Medium, 10 Jan. 2020, towardsdatascience.com/random-forest-classification-and-its-implementation-d5d840dbead0..

[9] Yiu, Tony. “Understanding Random Forest.” Medium, Towards Data Science, 12 June 2019, towardsdatascience.com/understanding-random-forest-58381e0602d2.

[10] Pros and cons of random forests. “Hands-On Machine Learning for Algorithmic Trading.” O’Reilly | Safari, 2019, www.oreilly.com/library/view/hands-on-machine-learning/9781789346411/e17de38e-421e-4577-afc3-efdd4e02a468.xhtml.

[11] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.