Development of Fake Review Filter using Deep Learning (Final Project Report, CSE-5334)

Son Nguyen, Kanishka Tyagi*

Department of Electrical Engineering
The University of Texas at Arlginton, Texas, USA, 76019

Abstract

Consumer reviews are important tool today as part of decision making. however, the credibility of these reviews is never sure since businesses commit review fraud, creating fake reviews on their own. Therefore fake review detection has garnered significant attention. Yelp fake review filter is being deployed since 2005 [3] and have been a trade secret [4]. Owing from our experience with neural networks, we wanted to investigate about the use of these often called " black boxes " in the fake review detection.

1. Motivation and Objective

In our project, we'll be looking into the development of neural network based algorithms that can extract useful information from a vast dataset. We have taken a machine learning perspective to look into the yelp data challenge. Therefore our objective in this work is to develop fake review filter using neural nets based deep learning architecture

2. Problems tacked in the project :

For problems as big as Yelp Data challenge, the scalability is a big issue in itself. We have try to investigate on the following main problems in the challenge.

1) How do you handle this large dataset without getting out of memory problems .
2) Will there be any arithmetic overflow error that can creep in ?
3) How do you model the problem in a Deep Learning architecture.

Since we dealing the problem from a neural network perspective, there are certain problems that comes with it such as :

1) How will we label this huge amount of data ?
2) How will be tune the various parameters in a neural network ?
3) What about the time and speed complexity ?
4) What kind of classifier (linear, logistic or non linear) should i use ?
5) How deep should be the architecture ?

E-mail address: kanishka.tyagi@mavs.uta.edu

In the subsequent parts of the report we will describe the answers to above question in detail and try to come up with an deep architecture that can later be used to classify the fake/non fake reviews. We will do an extensive experimental study, to assert a few conjecture about the whole fake/non fake review filter theory.

Section 3 will discuss the design of methods where we will discuss various other methods and our reasons that lead to the use of deep learning architecture. Section 4 will detail out the implementation of our method in detail with all the required theoretical and mathematical background. Results and evaluation will be discussed in Section 5 and finally our final comments on the project will be discussed in Section 6.

3. Design of Methods

We did an extensive literature survey on where does the designing of fake review filter stands in the research community. The seminal work in [5] is a state of the art as it reported an accuracy of 90 %. Ott.et.al uses Amazon Mechanical Turk (AMT) to crowd-source anonymous online workers. Later [6] boosted the accuracy to 91.2 %. Using crowds sourcing techniques is a time consuming task which is a major shortcoming of there approach. The results achieved in these papers are encouraging since they are using only the linguistic features. This gives us a hint to investigate on the feature engineering and based on our experience with neural nets, we found that using representative learning is a good approach to go for. One good model for representative learning is the Deep Learning architecture. Following points substantiate our reasoning for why we decided to pick Deep learning architecture of neural nets as a good candidate .

a) A Quick literature survey revealed to us that labelling of dataset is a major issue. Researchers have used crowd sourcing or hand labelled features that are very time consuming and mostly perceptual in nature. Deep Learning involves a major unsupervised feature learning step which completely eliminates this problem and offers a cleaner approach with a good performance in less amount of supervised data.

b) Deep Learning have performed extremely well in other areas of natural language processing [6], spech recognition [7][8] as well as computer vision [9] [10]. The authors have themselves taken part in applying Deep learning models in MIREX .

c) The idea of depth, feature usability and data abstraction in a deep learning architecture is at the heart of feature extraction engineering. All these ideas are a key aspect of representative learning strategies.

d) Finally, we are motivated by the ingrained philosophy of deep architecture about the abstraction and invariance. More abstract representation comes from more depth in the architecture and it's generally invariant to most of the local changes in the input. This can also be seen as a direct result from Cover's theorem. [11]

4. Implementation of Methods

4.1) Auto Encoder - Review

We use a greedy layer-wise unsupervised pre-training for the SAE [12]. We not only use the smoothness prior but also the sparsity prior is used while designing the algorithm. Since we train the SAE using a self taught learning as in [13], we do not assume the same distribution for the unlabelled and labelled data. By doing so, we design an independent framework for feature extraction which can then be plugged into any classifier. This makes our algorithm robust and universal in nature.

4.2) Notation and Symbols

Without the loss of generality, we restrict our self to a three later fully connected MLP with Linear output activation function. A typical topological architecture of a forward fully-connected MLP is shown in Figure 1. Given a dataset with N_v training patterns $\{\boldsymbol{x_p}, \boldsymbol{t_p}\}$where $\boldsymbol{x_p}$ denotes the $\boldsymbol{N}$ dimensional p^{th} input vector and $\boldsymbol{t_p}$ denotes the corresponding $\boldsymbol{M}$ dimensional desired output vector. In Figure 1, the three layers in MLP namely, input, hidden and output layer consist of $\boldsymbol{N}, \boldsymbol{N_h}$ and M respectively.

In input layer, $x_p = [x_p(1), x_p(2) \ldots, x_p(N+1)]^T$, denotes the input vector of p^{th} pattern. Threshold in the hidden latyer are handled by letting $x_p(N+1) = 1$.

In hidden layer, the k^{th} hidden unit is linked to i^{th} input unit by the hidden weight, $w_{hi}(i,k)$, where $i=1,\ldots,N+1$ and $k=1,\ldots,N_h$. $w_{hi}(N+1,k)$ represents the threshold of k^{th} hidden unit linking to constant input $x_p(N+1)$. Thus, for k^{th} hidden unit, the net function $n_p(k)$ is

$$n_p(k) = \sum_{i=1}^{N+1} w_{hi}(i,k) x_p(i) \tag{1}$$

and the activation $O_p(k)$ is denoted as

$$O_p(k) = f\left(n_p(k)\right) \tag{2}$$

Where, $f(.)$ denotes the nonlinear hidden layer activation function, generally being taken as sigmoid function.

$$f\left(n_p(k)\right) = \frac{1}{1+e^{-n_p(k)}} \tag{3}$$

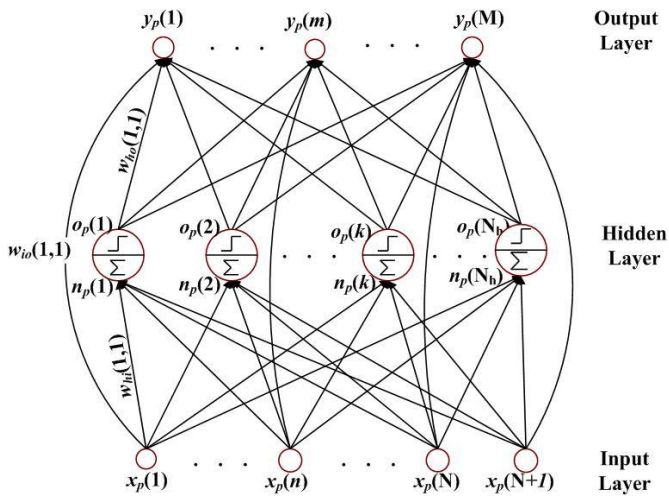

Figure 1. The architecture of a fully connected feedforward MLP

In the output layer, $w_{oh}(m,k)$ denotes the output weight connecting m^{th} output unit to k^{th} hidden unit . here $m=1,\ldots,M$. For p^{th} training pattern the output vector $\mathbf{y_p}$ is summarized as,

$$\boldsymbol{y_p} = \boldsymbol{W_{oh}} \cdot \boldsymbol{O_p} \tag{4}$$

For batch mode training, the typical error function E is *Mean Squared Error* (MSE) which is defined as

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{m=1}^{M} \left[t_p(m) - y_p(m)\right]^2 \tag{5}$$

$$= \frac{1}{N_v} \sum_{p=1}^{N_v} (\boldsymbol{t_p} - \boldsymbol{y_p})(\boldsymbol{t_p} - \boldsymbol{y_p})^T \tag{6}$$

Here $t_p(m)$ denotes the m^{th} element of the p^{th} desired output vector. Recall that $\boldsymbol{t_p}$ and $\boldsymbol{y_p}$ are the row vector. The general main purpose of any MLP training is to minimize the MSE based in the give n training set [14] [15]. Thus we can view a typical MLP training as a weight optimization problem. [16]-[18].

Overall, Figure 2 shows our final network . The yelp data is in *json* format. we parse it to create a dictionary of 100000 patterns. this is an unsupervised data which is then feed into the deep learner to create the features the features are then classifier by the linear or non linear classifier.

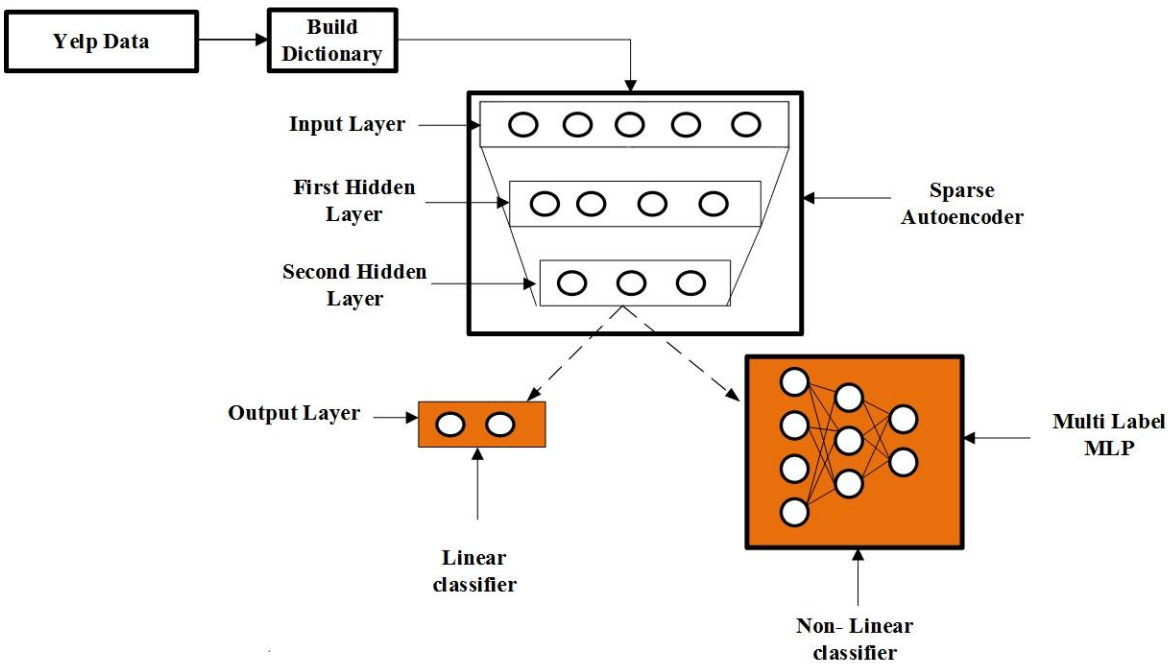

Figure 2. The proposed filtering algorithm

4.3) Learning Algorithms:

We now describe the building blocks of our algorithm in detail.

4.3.1) Gradient descent methods

The idea of gradient decent optimization dates back to Cauchy [19]. One of the earliest training algorithms is gradient descent back propagation algorithm. The origin comes from the linear approximation of the error function

$$E(w + \Delta w) \approx E(w) + \Delta w^T \frac{\partial E}{\partial w} = E(w) + \Delta w^T g \quad (7)$$

Where g is the gradient vector of E(w) evaluated at current w. Therefore a typical weight updating strategy will be

$$\Delta w = -z.g \quad (8)$$

Here z is the learning factor set to a small positive constant or determined by an adaptive or optimal method [20]-[22] The back propagation (BP) algorithms are essentially gradient descent methods [23]-[25]. In any standard BP the hidden weights (W_{hi}) are updated as:

$$(W_{hi})_{new} = (W_{hi})_{old} + z.G_{hi} \tag{9}$$

In (9), z is the learning factor which presents step length of updated hidden weights and $G_{hi}$is an N_h *by* $(N+1)$ negative Jacobian matrix of hidden weights which represents the direction of learning and is calculated as

$$G_{hi} = \frac{1}{N_v}\sum_{p=1}^{N_v} \delta_p(x_p)^T \tag{10}$$

Here $\delta_p = [\delta_p(1), \delta_p(2) \dots, \delta_p(N_h)]^T$ and the k^{th} element of it , $\delta_p(k)$ is formulated as following equation

$$\delta_p(k) = f'\left(n_p(k)\right)\sum_{i=1}^{M} \delta_{po}(i) w_{oh}(i,k) \tag{11}$$

Where

$$\delta_{po}(i) = 2(t_p(i) - y_p(i)) \tag{12}$$

In gradient descent algorithm, the output weights are updated as

$$(W_o)_{new} = (W_o)_{old} + z_1.G_o \tag{13}$$

Where $W_o = [W_{oi}: W_{oh}]$, z_1 is the learning factor for output weight changes, $G_o$is the gradient matrix of output weights and is given as

$$G_o = \frac{1}{N_v}\sum_{p=1}^{N_v} \delta_{po}(\overline{x_p})^T \tag{14}$$

Where $\overline{x_p} = [x_p: O_p]$.From (5) and (8), we can find that the learning factors, z and z_1 have direct effects on the convergence rate of training MLP.

Many researchers have suggested that each synaptic weight should be optimized separately with its own learning factor which is updated iteratively [31]-[34]. As mentioned, for many MLP training algorithms, the definitions of learning factor z and z_1 were either empirically assigned the fixed values or tuned based on some heuristic rules. When using an optimal learning factor, the gradient descent method is also known as steepest descent .

Although these methods can show fast convergence rates for some specific applications, they are suffered from lots of efforts to tune the proper learning factors empirically. The gradient decent method are typically slow convergence and using Newton's method require heavy storage and computationally heavy Hessian and inversion of hessian [23]. Hence in our investigation we use a non-gradient method which is a Gram-Schmidt implementation of the Orthogonal Least Square (OLS). Based on our experimental results, OLS proves to be an efficient learning process for MLP training

4.3.2) Output Weights Optimization with back propagation

Output Weights Optimization back propagation (OWO-BP) is a fully optimal second order learning factor method which convergence automatically [26][27]. As the output units have linear activations function, the OWO procedure can be realized by solving the linear equation [28] which result when gradients of E with respect to the output weights are set to zero. If W_o denotes the output weight, by taking the first derivative,

$\frac{\partial E}{\partial W_o} = 0$ leads to a set of linear equations given by:

$$\boldsymbol{C_a = W_o . R_a^T} \tag{15}$$

$$C_a = \frac{1}{N_v}\sum_{p=1}^{N_v} y_p \bar{x}_p^T \quad ; R_a = \frac{1}{N_v}\sum_{p=1}^{N_v} \bar{x}_p \bar{x}_p^T \tag{16}$$

Here, R_a is the autocorrelation matrix of values of $\overline{x_p}$ of (N+N_h+1) *by* (N+N_h+1) dimension. C_a is the correlation matrix of output vector y with $\overline{x_p}$ of (N+N_h+*1*) by M dimension. (14) is directly solved by using orthogonal least square (OLS) .

Since the equations are often ill conditioned, meaning that the determinant of R is close to 0, it is therefore often unsafe to use Gauss-Jordan elimination. The Singular value decomposition (SVD), LU decomposition and conjugate gradient are better. However (46) is most easily solved using orthogonal least squares (OLS) which is equivalent to using the QR decomposition .

As the training of output weights is equivalent to solving linear equations and the learning factor z is optimized by using second order Newton method, OWO-BP performs much better than gradient descent BP with much less computations than LM. However, one OLF are computed by OWO-BP for all hidden weights, it cannot guarantee the global optimal factor for the whole error surface. Hence, OWO-BP doesn't exhibit faster convergence than LM.

4.3.3) Learning Factor

In BP, the optimal learning factor (OLF) z is for all hidden weights which is optimized by Newton method as

$$z = -\frac{\partial E/\partial z}{\partial^2 E/\partial z^2} \tag{17}$$

Where $\partial E/\partial z$ and $\partial^2 E/\partial z^2$ are first order and second order derivative of error function with respect to learning factor of hidden weights.

$$\frac{\partial E}{\partial z} = \frac{-2}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\big(t_p(i) - y_p(i)\big) = \sum_{k=1}^{N_h} w_{oh}(i,k) f'\big(n_p(k)\big).\sum_{n=1}^{N+1} g(k,n) x_p(n) \tag{18}$$

And

$$\frac{\partial^2 E}{\partial z^2} = \frac{2}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[\frac{\partial y}{\partial z}\right]^2$$
$$= \frac{2}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[\frac{\partial y}{\partial z}\right]^2 \left[\sum_{k=1}^{N_h} w_{oh}(i,k) f'\big(n(k)\big)\sum_{n=1}^{N+1} g(k,n)\, x_p(n)\right]^2$$
(19)

5.) Results and Evaluation

We now describe the evaluation strategy employed to develop the review filter. Firstly, we took 100 000 reviews from 1.3 million, the reason we did not take the whole 1.3 million is that training will take time days or weeks, while we do not have that much time. After having 100 000 reviews, we then parse the reviews, remove stop words , perform lowers case, stemming, and take first 800 most frequent words to build a dictionary. Then the dictionary will be used to perform features extractor and transform the whole 100 000 reviews become training data. In a deep learner, this is represented as the unsupervised data of size 100000. The input dimension is 800 at first and second hidden layer has 400 and 200 units respectively. We take 300 iterations of training for each stack of auto-encoder and by the end of the training, we have 200 features with

100000 reviews. Now in the supervised step, we have hand labelled 10000 reviews and use it in the linear classifier. The results are as follows :

Table 1 : Results for the auto encoder based Fake filter review

Training time (in sec)	Training Classification Error (in %)	Testing Time	Testing Classification Error (in %)
3208	83	10	78

References

[1] Chapelle, Olivier, Bernhard Schölkopf, and Alexander Zien. "Semi-Supervised Learning."

[2] Mukherjee, Arjun, et al. "Spotting opinion spammers using behavioral footprints." *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013.

[3] http://officialblog.yelp.com/2009/10/why-yelp-has-a-review-filter.html

[4] http://officialblog.yelp.com/2010/03/yelp-review-filter-explained.html

[5] Ott, Myle, et al. "Finding deceptive opinion spam by any stretch of the imagination." *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011.

[6] Bengio, Yoshua, et al. "A neural probabilistic language model." *The Journal of Machine Learning Research* 3 (2003): 1137-1155..

[7] Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." *Signal Processing Magazine, IEEE* 29.6 (2012): 82-97.

[8] Dahl, George E., et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1 (2012): 30-42.

[9] Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.

[10] Ciresan, Dan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.

[11] Haykin, Simon S., et al. *Neural networks and learning machines*. Vol. 3. Upper Saddle River: Pearson Education, 2009.

[12] Bengio, Yoshua, et al. "Greedy layer-wise training of deep networks." *Advances in neural information processing systems* 19 (2007): 153.

[13] R., Battle, A., Lee, H., Packer, B., and Ng, A.Y. Self-taught learning: Transfer learning from unlabelled data. In ICML, 2007.

[14] Vapnik., V.N., The Nature of Statistical learning theory. New York: Springer Verlag, 1995.

[15] Geman., S., Bienenstock., E, Doursat., R., "Neural Networks and the bias/variance dilemma," Neural Computation, vol 4, no1, pp.1-58,Jan 1992

[16] 5 Haykin., S, Adaptive Filter Theory, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

[17] Manry., M,T., Apollo., S,J., and Yu., Q., "Minimum mean Square estimation and Neural networks," Neurocomputing, vol 13.

[18] Barnard. E., "Optimization for training neural nets,"IEEE Trans, Neural Networks, vol 3, no.2, pp.232-240, 1992.

[19] Moller., M, " Efficient training of feed-forward neural networks," Ph.D. dissertation, Aarhus University, Denmark, 1997

[20] Maldonado.,F.J., Manry.,M.T., and Kim.,T.H., "Finding optimal neural network basis function subsets using the Schmidt procedure," Proceedings of *the International Joint Conference on Neural Networks*,(Portland, Oregon, July 20-24 ,2003),1, 444 – 449

[21] Magoulas., G,D., Vrahatis., M,N., and Androulakis., G,S., "Improving the convergence of the Backpropagation algorithm using learning adaption methods," Neural Computation, vol .11, no.8, pp. 1769-1796, 199.

[22] Nachtsheim., P,R., " A first order adaptive learning rate algorithm for back propagation networks," in Proc. Of IEEE Worlds Congress on Computational Intelligence' 94, 1994, pp.257-262.

[23] Werbos., P, " Beyond regression: New tools for predictions and analysis in the behavioral science," PhD. dissertation, Harvard University, Cambridge, Mass., 1974.

[24] Werbos., P, " Backpropagation: Past and future," in Proc. Of IEEE International Joint Conference on neural Networks'88, 1988, pp.343-353.

[25] Yu,X., Efe,M.,O., and Kaynak.,O., "A general backpropagation algorithm for feed forward neural networks learning," IEEE Trans. Neural Networks, vol.13, pp251-254, Jan. 2002

[26] Gong,B., "A Novel Learning Algorithm of Backpropagation Neural Network," IITA *International Conference on Control, Automation and System Engineering* 2009, 411-414.

[27] Maldonado.,F.J., Manry.,M.T., and Kim.,T.H., "Finding optimal neural network basis function subsets using the Schmidt procedure," Proceedings of *the International Joint Conference on Neural Networks*,(Portland, Oregon, July 20-24 ,2003),1, 444 – 449

[28] Manry., M.T. " Fast training of Neural network for remote sensing, " Remote Sensing Reviews,vol.9, pp. 77-96, 1994.