Survey of Business Perception Based on Sentiment Analysis through Deep Neuronal Networks for Natural Language Processing

Ing. Mónica Patricia Pineda Vargas and Ing. Miguel Alexander Chitiva Díaz
Ing. Brayan S. Reyes Daza, Ing. PhD. Octavio J. Salcedo Parra,
Universidad Distrital Francisco José de Caldas, Universidad Nacional de Colombia, Bogotá D.C., Colombia

Abstract: In recent years, the machine-learning field, deep neural networks has been an important topic of research, used in several disciplines such as pattern recognition, information retrieval, classification and natural language processing. Is in the last that this paper it's going to be our principal topic, in this branch exist an specific task that in literature is called Sentiment Analysis were the principal function is to detect if an opinion is positive or negative. The paper shows how use this subset of the machine learning knowledge and use it for give us an insight in the question: what is the perception in a business or a product by means of the opinion of the consumers in social networks?

Key Words Deep Learning, Natural Language Processing, Neural Networks, Machine Learning

INTRODUCTION

In a globalized world the market competition has been increased and become stronger, the technology sector it is one with greater growing behavior in the last years, by that this economy sector has been gaining a lot of dynamism; thereby the companies became forced to launch new products in a shorter period. Therefore, nowadays it's a necessity make a profile of the market, analyzing the social perception of the company and their products with the objective of improve the business policies and by that way gain a bigger segment of the market. The social networks have an incommensurate potential, because could bring us a psychological profile of the users; that information can help or even replace classical market analysis such as ethnography, statistics, polling etc. That potential remains undeveloped, impeding the access of this knowledge to the CEOs to overcome weak commercial strategies an enhance the positive ones, this kind of knowledge could be give an important competitive advantage over another companies.

A common application of Natural Language Processing is a Sentiment Analysis, whose goal is to extract the sentiments and emotional content in text. This have many useful such as business intelligence, where we know about consumer reactions to a specific product detecting in online comments. Extract this kind of knowledge is possible with the use of machine learning techniques, such as presented by [1], [2], [3], these techniques are based in the Bag-of-Words paradigm, this paradigm works rating every word in a language; this rating varies between positive and negative but have a drawback that don't take in account the context of the phrase in which the word appears. For example employing a Naive Bayes classifier such as the implementation shown in [19] using the word "bad" in a different context we would obtain the same negative result i.e. "it's bad" return a probability of 0.8 to be a negative sentiment, but if we change the context given the phrase "it's not bad" remains with the same probability although we know that it's a positive sentence, if we use a highly positive context like "it's not bad at all" the probability of be a negative sentence remains.

This behavior can be explained because the word "bad" have a high negative weight in the English language with comparison with the rest. Another similar perspective that attempt to preserve the semantical information of the phrases is the Bag-of-n-grams this technique treat several words as only one, then for the combination of words only have a unique score; but this improvement have the drawback that only preserve an small amount of the semantical information, we can try a longer combinations let's say 5 words (Bag-of-5-gram), but then we need to store near to the 5 permutation scores of the entire English language, and this perspective don't improve in a substantial way the results.

This paper is organized as follows, in the section 2, we show the related work in this topic, in the section 3 the mathematical background behind the Word2Vec, in the section 4 we explain the experimentation framework that we used and the results derived from the experimentation and in the section 5, we discuss the conclusions and the future work.

RELATED WORK

In recent years, many works has been done in the field of Sentiment analysis. This field started since the beginning of the century and it was intended for binary classification, which assigns opinions or reviews to bipolar classes such as positive or negative, i.e. the opinions was assigned into two classes such as positive or negative. Nowadays, one of the most used models are neural networks, as a work presented by Mikolov in [4] using Recurrent Neural Network language model (RNNLMs). Other works implements a classical feedforward neural network language models such as [5], [6], [7], [8], [9], that unlike (RNNLMs) this not maintain a hidden-layer of neurons with recurrent connections to their own previous values. A characteristic of neural network language models is their representation of words as high dimensional real valued vectors, i.e. the words are converted via a learned lookuptable into real valued vectors, which are used as the inputs to a neural network. The first proposals of distributed word representations, was including by [11], [12] in the 90's and most recently, studied in the context of feed-forward networks by Bengio in [5] though a linear projection layer and a nonlinear hidden layer, used to learn jointly the word vector representation and a statistical language model. Later in RNNLMs by Mikolov, demonstrated outstanding performance in terms of word-prediction. Today, Le and Mikolov proposed in [13] a Paragraph Vector, an unsupervised algorithm that learns fixed-length feature representations such as paragraphs, sentences and documents, achieving a new state of art results on sentiment analysis tasks. We apply these methods to perform a sentimental analysis based on some comments from twitter.

3 PRIOR KNOWLEDGE 3.1 Feed-Forward Neural Network

This was proposed by Bengio in [5] as a language model. It consist in an input layer, a hidden layer with an output layer interconnected by modifiable weights, represented by links between layers as in the Figure 1. Furthermore exists, a single bias unit that is connected to each unit other than the input units. These networks compute a series of transformations between their input and their output and the activities of the neurons in each layer are a nonlinear function of the activities in the layer below. The input units represent a feature vector components and signals emitted by output units will be discriminant functions used for classification. In [5], consist in a *N* previous words that are encoded through 1-of-*V* coding (*V* is the size of vocabulary).

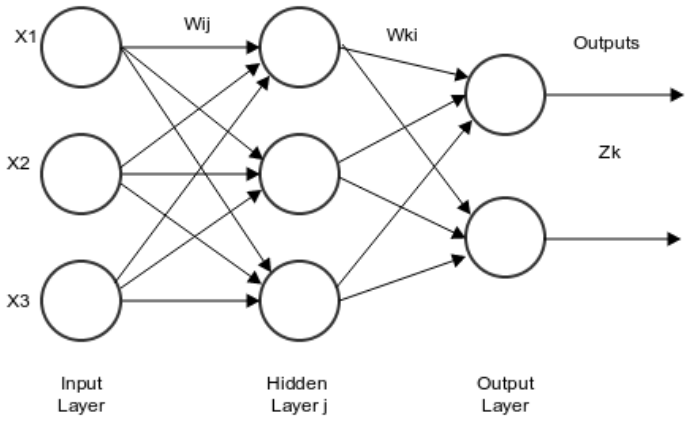

Figure 1. Feed Forward Neural Network

Each hidden unit realize the weighted sum of its inputs to form its (scalar) net activation, i.e. the net transference is the inner product of the inputs with the weights at the hidden unit, as can be seen in Equation 1.

$$z = b + \sum_{i=1} x_i w_{ji} = b + W^T x \qquad (1)$$

Where w_{ji} denotes the input to hidden layer weights at the hidden unit *j* and *b* is the bias.

$$\phi(z) = \frac{1}{1+e^{-z}} \qquad (2)$$

3.1.1 Activation Function

The activation function of a neuron defines the output of that neuron given an input or set of inputs. The most common activation function used is the logistic function, also known as the sigmoid function.

3.2 Backpropagation

Is one of the most general methods for supervised learning of multilayer neural networks, proposed by Rumelhart, Hinton and Williams in [14]. This algorithm is based on delta rule and it looks for the minimum of the error function in weight space using the method of gradient descent. The method calculates the gradient of a loss function with respects to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, thus be able to minimize the loss function.

The error is a scalar function of the weights such that the network outputs match the desired outputs, it is minimized. To reduce this measure of error, the weights are adjusted. The training error is defined by the equation 3:

$$E(w) = \frac{1}{2}\sum_{n\epsilon training}(t^n - y^n)^2 \qquad (3)$$

Where y is the network output and t is the target output. As the backpropagation is based on gradient descendent, the weights are updated in a direction that will reduce the error:

$$\Delta w = -\varepsilon \frac{\partial E}{\partial \omega} = -\varepsilon \sum_n x_i^n y^n (1 - y^n)(t^n - y^n) \qquad (4)$$

Where ε is the learning rate which indicates the variation in weights. To actualize the weights used a recursive algorithm, starting with the output neurons and working backwards to the input layer, adjusting weights for the m iteration it as follow:

$$w(\boldsymbol{m} + \mathbf{1}) = w(\boldsymbol{m}) + \boldsymbol{\Delta w}(m) \qquad (5)$$

Finally, the error regarding weights is:

$$\frac{\partial E}{\partial z_j} = \frac{\partial y_i \partial E}{\partial z_j \partial y_j} = y_j(1 - y_i)\frac{\partial E}{\partial y_j} \qquad (6)$$

$$\frac{\partial E}{\partial y_j} = \sum_j \frac{\partial z_j \partial E}{\partial z_j \partial y_i} = \sum_j \boldsymbol{w}_{ij}\frac{\partial E}{\partial z_j} \qquad (7)$$

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial z_i \partial E}{\partial z_j \partial \omega_{ij}} = y_i \frac{\partial E}{\partial z_j} \qquad (8)$$

3.3 The Skip-gram and Continuous Bag-of-Words (CBOW) Models

Distributed representations of words was proposed by Rumelhart in [16] and have become extreme successful. The advantage of this model is that the representations of similar words are close in the vector space. They showed that in distributed representation of words are many types of similarities among words that can be expressed as a linear translations. For example, vector operations ”king”-”man”+”woman”= ”queen” [17].

Two models for learning word representation proposed in 2013 by Mikolov [15] was the Continuous Bag-of-words (CBOW) and Skip-gram models. In CBOW the goal is to predict one target word given the surrounding words. Similarly, the training objective of the Skip-gram model is predict a context, a window of word given a single word. This two models are based on artificial neural networks [5], [7].

3.4 CBOW

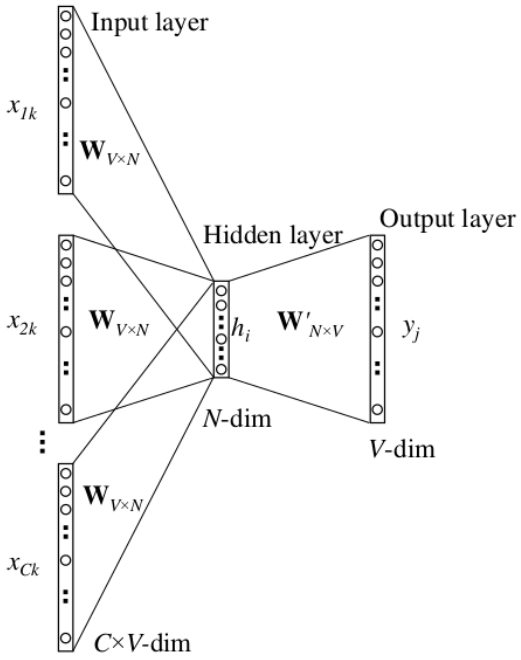

Figure 2: Continuous Bag-of-words architecture

The Figure 2, shows the architecture of this models, the input of CBOW are vectors with one-hot encoded vector, i.e, *V* is the vocabulary vector and the input is a vector with size *V*; the values of input represents the times to appears each word of the context in the vocabulary. The weights between the input and the output layer are represented by a VxN matrix ***W***. Each row of ***W*** is a vector v with dimension *N*, that represents the associated word of the input layer. Given a context, then

$$h = \frac{1}{C}\boldsymbol{W}(x_1 + x_2 + \cdots + x_C) \quad (9)$$

$$h = \frac{1}{C}(v_{\omega 1} + v_{\omega 2} + \cdots + v_{\omega C}) \quad (10)$$

where $\omega_1 + \omega_2 + \cdots + \omega_3$ are the words in the context and v_ω is the input vector of a word ω. The hidden layer to the output layer have the weight matrix $\boldsymbol{W'} = \{\omega'_{ij}\}$ with size . Using this weights, the score $u_j$can be calculated for each word in the vocabulary.

$$u_j = \boldsymbol{v'}^T_{\omega j} * \boldsymbol{h} \quad (11)$$

Where $\boldsymbol{v'}_{\omega j}$ is the *j*-thcolumn for the weights matrix **W'.** Using soft-max, we can obtain the posterior distribution of words.

$$p(\omega_j|\omega_I) = y_i = \frac{\exp(\boldsymbol{v'}^T_{\omega O}\boldsymbol{v}_{\omega I})}{\sum_{j'=1}^{V}\exp(\boldsymbol{v'}^T_{\omega' j}\boldsymbol{v}_{\omega I})} \quad (12)$$

Where y_j is the output of the *j*-th node in the output layer. $\boldsymbol{v}_\omega$ is the input vector and $\boldsymbol{v'}_\omega$ is the output vector of the word ω.

The loss function is *E*

$$E = -\log p(\omega_O|\omega_{I,1}, \ldots, \omega_{I,C}) \quad (13)$$

$$= -u_{j*} + log\sum_{j'=1}^{v}\exp(u_{j'}) \quad (14)$$

$$= -v'^T_{wO} * h + log\sum_{j'=1}^{v}\exp(v'^T_{wj} * h) \quad (15)$$

the goal is minimize $E.j_*$ is the index of the actual output word in the output layer. The prediction error e_j of the output layer is given by the derivative of *E*

$$e_j = \frac{\partial E}{\partial z_j} = y_i - t_j \qquad (16)$$

where t_j is 1 only when the *j*-th node is the actual output word, in other case is 0. To obtain the gradient on the hidden to output layer weights is necessary to take the derivative of loss function on ω_{ij}

$$\frac{\partial E}{\partial \omega'_{ij}} = \frac{\partial E\, u_j}{\partial u_j \partial \omega'_{ij}} = e_j * h_i \qquad (17)$$

The weight updating equation for hidden to output weights is calculated using stochastic gradient descendent

$$\omega'^{(new)}_{ij} = \omega'^{(old)}_{ij} - \in * e_j * h_i \qquad (18)$$

$$v'^{(new)}_{w_j} = v'^{(old)}_{w_j} - \in * e_j * h, for\ j = [1, \dots, V] \quad (19)$$

Where ∈ is the learning rate and h_i is the i-th node in the hidden layer. We obtain the update of the input to hidden layer wights as:

$$v'^{(new)}_{w_{I,c}} = v'^{(old)}_{w_{I,c}} - \frac{1}{C} * \in * EH, for\ c = [1, \dots, C] \qquad (19)$$

Where

$$EH = \frac{\partial E}{\partial h_i} = \sum_{j=1}^{v} e_j * \omega'_{ij} \qquad (19)$$

Where h_i is the output of the i-th node in the hidden layer and EH is the sum of the output vectors for all words in the vocabulary, weighted by the prediction error e_j.

3.5 Hierarchical Soft-max

It was first introduced by Morin and Bengio [18] in the context of neural network language models. It uses a Huffman tree representation of the output layer based on word frequencies with the *V* words as it leaves and for each node, represents the relative probabilities of its child nodes. Each step in the search process from root to the target word is a normalization. The final probability for finding the target word is the continuous multiplication of probabilities in each search step.

Each word ω can be reached by an appropriate path from the root of the tree. Then the hierarchical softmax defines the probability of a word being the output word $p(\omega_O|\omega_I)$as follows:

$$p(\omega|\omega_I) = \prod_{j=1}^{L(\omega)-1} \sigma\left[\left|n(\omega, j+1) = ch\left(n(\omega, j)\right)\right|\right] * {v'_{n(\omega,j)}}^T h \qquad \mathbf{(22)}$$

Where $n(\omega, j)$ is the j-th node on the path from the root to word ω. $L(\omega)$ is the length of the path. So, $n(\omega, 1)$ is the root and $n(\omega, L(\omega))$ is $\omega.ch(n)$ is a left child of n and $[|x|]$ is 1 if x is True or -1 in otherwise. $\sigma(x) = \frac{1}{}(1 + \exp(-x)), v'_{n(\omega,j)}$ is the vector representation for the inner node $n(\omega, j)$ and **h** is $\frac{\mathbf{1}}{\boldsymbol{C}}\sum_{\boldsymbol{c=1}}^{\boldsymbol{C}} \boldsymbol{v}_{\boldsymbol{\omega_c}}$ for the CBOW model.

3.6 Word2vec

Word2vec is an efficient implementation of the CBOW and the Skip Gram model for computing vector representations of words. It was developed by Google in 2013. This tool takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications. This means that the model learns to map each word into a low dimensional continuous vector space from their distributional properties observed in some raw text corpus.

4 EXPERIMENTS

The Figure 3 shows the implemented process to carried out the experiment. The text corpus was a Google News Dataset with 100 billion words in representation vectorial. This is our vocabulary.

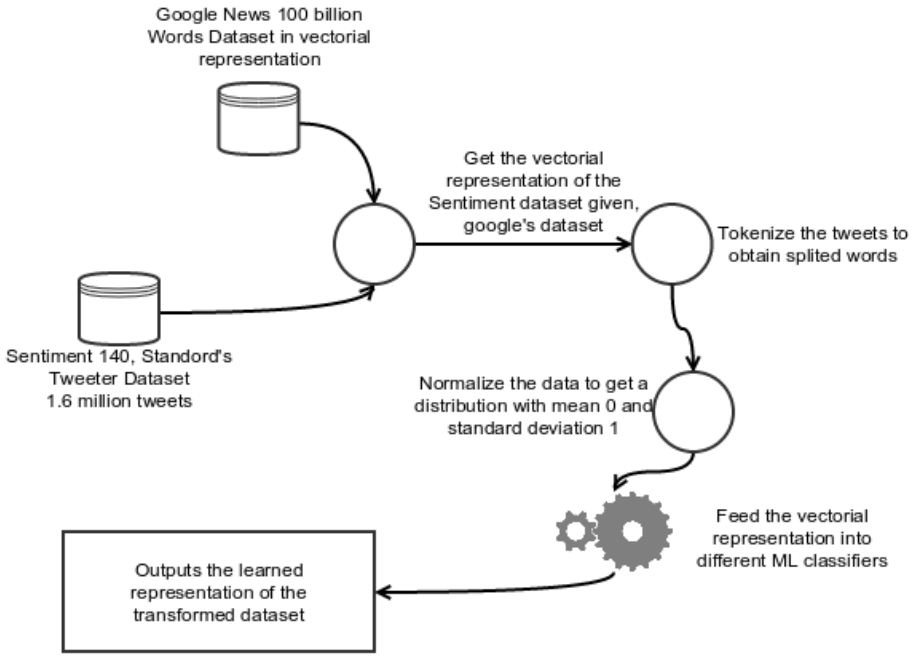

Figure 3. Experiment process

To train the model, we use Sentiment 140, a Stanford's Tweeter Dataset with 1'600.000 tweets, 800.000 positive and 800.000 negatives, that they have been classified by humans. The format of this database consists of 6 fields:

- Polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
- ID of the tweet
- Date of the tweet
- The query. If there is no query, then this value is NO QUERY.
- User that tweeted
- Text of the tweet

As the tweets have a symbols, links, etc., is necessary tokenize each tweet and then, obtain the vector representation of each one of them. As we mentioned above, the vector representation is for a word, for this we have to obtain the representation for a single word and then calculate the average for all words in the tweet. In this way, we obtain the vector representation for the full tweet.

The next step, as can be seen in the Figure 3, is to normalize where we move our dataset into a gaussian distribution with a mean of zero and standard deviation 1, meaning that values above the mean will be positive, and those below the mean will be negative.

Finally as shown in Figure 3, a classifier is used. For this experiment we used Logistic Regression, Support Vector Machine and Random Forest.

PARTIAL RESULTS TABLE

Algorithm	Accuracy
Logistic Regression	0.746 (+/- 0.009)
Support Vector Machine	0.748 (+/- 0.010)
Random Forest	0.672 (+/- 0.005)

5 CONCLUSIONS

The results of this naive experimental framework offers a performance far from the state of the art algorithms, but taking in account the lack of preprocessing over the twitter database such as removing the misspelling, removing urls, treat in a special way the emoticons that clearly express sentiments, and another techniques that can remove noise from the data we can introduce all of this enhancements to achieve better results in future work. Also provide another sources of knowledge such as topic related information, with the aim to offer a better semantical meaning, moreover the syntactical information isn't taking in account this kind of information can give to the model another perspective that helps to define more accurately the meaning not only of the window of the neural network but also about the entire context of the tweet phrase.

BIBLIOGRAPHY

1. Geetika Gautam, Divakar yadav, Sentiment Analysis of Twitter Data Using Machine Learning Approaches and Semantic Analysis, Department of Computer Science \& Engg. Jaypee Institute of Information technology Noida, India, 2014

2. Warih Maharani, Microblogging Sentiment Analysis with Lexical Based and Machine Learning Approaches, Faculty of Informatics Telkom Institute of Technology, Bandung Indonesia, 2013

3. P. Waila Marisha, V.K. Singh ,M.K. Singh, Evaluating Machine Learning and Unsupervised Semantic Orientation Approaches for Sentiment Analysis of Textual Reviews, Banaras Hindu University, Varanasi India, 2012

4. G. Zweig, T. Mikolov. Context Dependent Recurrent neural Network Language Model. Microsoft, Redmond, WA USA. 2012

5. Y. Bengio, R. Ducharme, Vincent, P., and C. Jauvin, "A neural probabilistic language model," Journal of Machine Learning Reseach, vol. 3, no. 6, 2003.

6. Holger Schwenk, "Continuous space language models," Computer Speech and Language, vol. 21, no. 3, pp. 492 – 518,2007.

7. Andriy Mnih and Geoffrey Hinton, "Three new graphical models for statistical language modelling," in Proceedings of the 24th International Conference on Machine Learning, 2007.

8. Hai-Son Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in ICASSP, 2011.

9. Le Hai Son, Alexandre Allauzen, and Francois Yvon, "Measuring the influence of long range dependencies with neural network language models," in Proceedings of the Workshop on the Future of Language Modeling for HLT (NAACL/HLT 2012), 2012.

10. Tomas Mikolov, Martin Karafiat, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In Proceedings of Interspeech 2010.

11. G.E. Hinton. Learning distributed representations of concepts. In Proceedings of the eighth annual conference of the cognitive science society,Amherst, MA. 1986.

12. J.L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. Machine learning, 7(2):195–225. 1991

13. Q. Le, T. Mikolov. 31 st International Conference on Machine Learning, Beijing, China, 2014.

14. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. In Parallel distributed processing: explorations in the microstructure of cognition, vol. 1, David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group (Eds.). MIT Press, Cambridge, MA, USA 318-362.

15. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

16. D.E. Rumelhart, G.E. Hinton, and R.J Williams. 1986. Learning representations by back-propagating errors. Nature, 323(6088):533–536.

17. Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In NAACL HLT

18. Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Proceedings of the international workshop on artificial intelligence and statistics, pages 246–252, 2005.

19. Natural Language Processing APIs and Python NLTK Demos, (March 23, 2015). Retrieved from http://text-processing.com/demo/sentiment/.