OPINION SPAM DETECTION

Suchismitha Vedala
svedala@uh.edu
UHID:1470929
Graduate Student
Department of Computer Science

Abstract— Online reviews are often the primary factor in a customer's decision to purchase a product or service, and are a valuable source of information that can be used to determine public opinion on these products or services. Because of their impact, manufacturers and retailers are highly concerned with customer feedback and reviews. Reliance on online reviews gives rise to the potential concern that wrongdoers may create false reviews to artificially promote or devalue products and services. This practice is known as Opinion (Review) Spam, where spammers manipulate and poison reviews (i.e., making fake, untruthful, or deceptive reviews) for profit or gain. Since not all online reviews are truthful and trustworthy, it is important to develop techniques for detecting review spam. By extracting meaningful features from the text using Natural Language Processing (NLP), it is possible to conduct review spam detection using various machine learning techniques. Additionally, reviewer information, apart from the text itself, can be used to aid in this process. In this project, we survey the prominent machine learning techniques that have been proposed to solve the problem of review spam detection and the performance of different approaches for classification and detection of review spam. This project is focused on implementing algorithms that efficiently classify an unseen review based on a given set of reviews. In most of this project, we focus on implementing Deep Learning concepts with Tensor Flow to classify an unseen review with maximum accuracy.

Index Terms— Review Spam, Classifier, Spam Detection, Machine Learning, Deep Learning, Tensor Flow,

1 INTRODUCTION

As the Internet continues to grow in both size and importance, the quantity and impact of online reviews continually increases. Reviews can influence people across a broad spectrum of industries, but are particularly important in the realm of e-commerce, where comments and reviews regarding products and services are often the most convenient, if not the only, way for a buyer to decide on whether to buy them. Online reviews may be generated for a variety of reasons. Often, to improve and enhance their businesses, online retailers and service providers may ask their customers to provide feedback about their experience with the products or services they have bought, and whether they were satisfied or not. Customers may also feel inclined to review a product or service if they had an exceptionally good or bad experience with it. While online reviews can be helpful, blind trust of these reviews is dangerous for both the seller and buyer. Many look at online reviews before placing any online order; however, the reviews may be poisoned or faked for profit or gain, thus any decision based on online reviews must be made cautiously. Furthermore, business owners might give incentives to whoever writes good reviews about their merchandise, or might pay someone to write bad reviews about their competitor's products or services. These fake reviews are considered review spam and can have a great impact in the online marketplace due to the importance of reviews.

Review spam can also negatively impact businesses due to loss in consumer trust. The issue is severe enough to have attracted the attention of mainstream media and governments. For example, the BBC and New York Times have reported that "fake reviews are becoming a common problem on the Web, and a photography company was recently subjected to hundreds of defamatory consumer reviews". In 2014, the Canadian Government issued a warning "encouraging consumers to be wary of fake online endorsements that give the impression that they have been made by ordinary consumers" and estimated that a third online reviews were fake. As review spam is a pervasive and damaging problem, developing methods to help businesses and consumers distinguish truthful reviews from fake ones is an important, but challenging problem.

- *Arjun Mukherjee is with the Department of Computer Science at University of Houston, Houston, TX 77004. E-mail: arjun@cs.uh.edu*
- *Suchismitha Vedala is with the Department of Computer Science at University of Houston, Houston, TX 77004. E-mail: svedala@uh.edu*

Manuscript received 11 Dec. 2016

2 Related Work

Spam, whose definitions usually center on the concept of unsolicited message, has been bothering Internet users for a long time. Email spam is one of the most prevalent types of spams that could be dated back to long ago. Web spam is another form of spams whose objective is to game the ranking algorithm of a search engine to get an undeserved high ranking. As social media gained its popularity in recent years, social network spam came along. One of the variants involves throwaway accounts created in batch to somehow bait regular users to clicks certain link for personal gain. Opinion spam is different from the above types of spams from various perspectives. One of the most prominent differences is that opinion spam is arguably the most "subtle" kind of spams. This is because it is not only completely ineffective, but also very harmful to the reputation of the target that it promotes, when it gets caught. Therefore, opinion spammers would generally try their best to disguise their opinion spams as genuine opinions. Carefully-written opinion spams have caused great challenges in manually identifying the spams and annotating the ground truth, which is in concert with the finding that human are poor judge of deception.

One of the earliest researches on opinion spam is Jindal and Liu. Jindal and Liu (2008) built a logistic regression classifier with review feedback features, title and content characteristics and rating related features. Other researchers (Li et al. 2011; Ott et al. 2011; Feng, Banerjee, and Choi 2012) focused solely on the textual features, for instance, unigrams and bigrams. Mukherjee et al. (2013) further boosted the performance by appending users' behavioral features. Network-based approaches are exploited in (Wang et al. 2011; Akoglu, Chandy, and Faloutsos 2013; Li et al. 2014b) using various relational classifiers or graph propagation algorithms. Besides, with only a small portion of labeled reviews, researchers pointed out that using Positive-Unlabeled Learning (PU learning) (Li et al. 2014a; Ren, Ji, and Zhang 2014) outperforms traditional supervised learning

3 Methodology and Implementation

3.1 Data Set

The goal of this project was to determine whether a new unseen review is fake or not fake. In this regard, there was a need for a dataset containing various reviews and the class whether it is filtered or not. Different datasets were available to the researchers such as the Amazon reviews, Movie reviews etc., each of which had different categories such as reviewContent, reviewId, rating, flags etc. Additionally, texts that are written in a more informal style are more suitable for our project. Our project experimental results are based on a real-life Yelp data set.

Despite opinion spam being prevalent, there are only few which correctly classify the reviews and implement the algorithms. The Yelp filter set has been prevalent for long now and is quite popular in usage across many researchers. We download the Yelp filtered review dataset used in [Mukherjee et al., ICWSM 2013; Kc and Mukherjee WWW 2016]. The Yelp Dataset consists of two databases: yelp_hotel and yelp_res each with 5 tables; {authorfeatures, hotel, review, reviewer, review_features} and {authorfeatures, restaurant, review, reviewer, review_features}. We consider the yelp_res(restaurant) database for your project.

For our project implementation, we need the review sentences which are filtered as spam. The review table consists of various columns {date, reviewID, reviewerID, reviewContent, rating, usefulCount, coolCount, funnyCount, flagged,restaurantID}. The flagged field consists of 1, 0, Y, N, YR, NR. For our project, we consider those reviews which are flagged as Y and N, where the reviews tagged as Y mean 'Spam' and the reviews tagged as N means 'Not spam'

3.2 Methodology

In the process of opinion spam detection, we implement Various algorithms including many methodologies on the data set to classify an unseen example.

a) **Bag of words:**
Bag of Words (BoW) is an algorithm that counts how many times a word appears in an article/sentence/document. Those word counts allow us to compare them and gauge their similarities for applications like search, document classification and as one of the features in machine learning. BoW is a method for preparing text for input in a deep-learning net. This specific strategy involves tokenization, counting and normalization known as the Bag of Words or "Bag of n-grams" representation. In a bag of words approach, individual or small groups of words from the text are used as features. These features are called n-grams and are made by selecting n contiguous words from a given sequence, i.e.,

selecting one, two or three contiguous words from a text. These are denoted as a unigram, bigram, and trigram (n = 1, 2 and 3) respectively. These features are used by Jindal et al. Li et al. and Fei et al. However, Fei et al. observed that using n-gram features alone proved inadequate for supervised learning when learners were trained using synthetic fake reviews, since the features being created were not present in real-world fake reviews. For bag of words, we have removed the punctuations, spaces so that only words are considered.

3.2.2 Algorithms

There are several algorithms that can be used for classification techniques. In the following section, the advantages and disadvantages of the naive Bayes classifier, logistic regression, the decision support vector machine and Bayesian-based logistic regression algorithms are briefly described.

a. Naïve Bayes

This algorithm has been widely used because of its simplicity. For naive Bayes conditional independent assumptions, the algorithm gathers the needed information quicker than other discriminative algorithms such as logistic regression, which leads to the use of less training data. Naive Bayes outperforms in real applications and as such, this algorithm is the best choice in cases where fast, easy and reliable classifier is needed. The primary disadvantage of this classifier is that it is not able to understand interactions between criteria.

b. Support Vector Machine

In the over fitting cases, the support vector machine (SVM) classifier performs with high accuracy and very strong theoretical guarantees, even when the classes involved are not linearly distinguishable. This algorithm is highly recommended for use in text classification, as its input vectors are highly dimensional. The disadvantage of this classifier is that it is memory intensive and too complicated to explain to others with limited knowledge thereof.

Overall, it has been stated by several researchers, as well as in practice, that the support vector machine classifier is successful in classifying the input data into related classes and can even be used for solving regression problems. Modern support vector machines differ from earlier algorithms in three ways, that is, in terms of optimal hyper-plane, kernel, and soft margins

c. **Logistic Regression**

Logistic regression is a regression model where the dependent variable (DV) is categorical. Logistic regression was developed by statistician David Cox in 1958. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It is also called a qualitative response/discrete choice model in the terminology of economics. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Thus, it treats the same set of problems as probit regression using similar techniques, with the latter using a cumulative normal distribution curve instead.

4 PROPOSED IDEA

4.1 DEEP LEARNING AND CNN

Deep learning models have achieved remarkable results in computer vision (Krizhevsky et al., 2012) and speech recognition (Graves et al., 2013) in recent years. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models (Bengio et al., 2003; Yih et al., 2011; Mikolov et al., 2013) and performing composition over the learned word vectors for classification (Collobert et al., 2011). Word vectors, wherein words are projected from a sparse, 1-of-V encoding (here V is the vocabulary size) onto a lower dimensional vector space via a hidden layer, are essentially feature extractors that encode semantic features of words in their dimensions. In such dense representations, semantically close words are likewise close—in Euclidean or cosine distance—in the lower dimensional vector space. Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features. Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing, search query retrieval sentence modeling and other traditional NLP tasks.

4.2. TENSORFLOW

TensorFlow is constructed around the basic idea of building and manipulating a computational graph, representing symbolically the numerical operations to be performed. This allows TensorFlow to take advantage of both CPUs and GPUs right now from Linux 64-bit platforms such as Mac OS X, as well as mobile platforms such as Android or iOS. Another strength of this new package is its visual TensorBoard module that allows a lot of information about how the algorithm is running to be monitored and displayed. Being able to measure and display the behavior of algorithms is extremely important in the process of creating better models. I have a feeling that currently many models are refined through a little blind process, through trial and error, with the obvious waste of resources and, above all, time.

TensorFlow was originally developed by the Google Brain Team, with the purpose of conducting Machine Learning and deep neural networks research, but the system is general enough to be applied in a wide variety of other Machine Learning problems. Google released its TensorFlow engine under an open source license (Apache 2.0). TensorFlow can be used by developers and researchers who want to incorporate Machine Learning in their projects and products, in the same way that Google is doing internally with different commercial products like Gmail, Google Photos, Search, voice recognition, etc.

4.3 Softmax Classifier

The SVM is one of two commonly seen classifiers. The other popular choice is the **Softmax classifier**, which has a different loss function. The Softmax classifier is its generalization of binary logistic regression to multiple classes. Unlike the SVM which treats the outputs $f(x_i,W)$,as (uncalibrated and possibly difficult to interpret) scores for each class, the Softmax classifier gives a slightly more intuitive output (normalized class probabilities) and also has a probabilistic interpretation that we will describe shortly. In the Softmax classifier, the function mapping $f(x_i;W)=Wx_i$ stays unchanged, but we now interpret these scores as the un normalized log probabilities for each class and replace the *hinge loss* with a **cross-entropy loss.** The full cross-entropy loss that involves the softmax function might look scary if you're seeing it for the first time but it is relatively easy to motivate.

5 EXPERIMENTS AND RESULTS

Two experiments were conducted on our project and the accuracy was measured across these.

5.1 Experiment 1 – Implementing Naïve Bayes and SVM with Bag of Words

Naive Bayes (NB) and Support Vector Machine (SVM) models are often used as baselines for other methods in text categorization and sentiment analysis research. However, their performance varies significantly depending on which variant, features and datasets are used. Generally, SVM is regarded to perform better when compared to Naïve Bayes, however high accuracy is not achieved with either. To design an efficient algorithm to our opinion spam detection, we implement a SVM over the Naïve Bayes model. An SVM is built over the NB log-count of bag of words as features values to get the maximum accuracy.

Table 1: NBSVM Performance

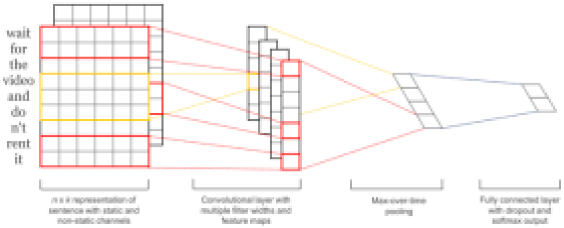
Classifier	Uni-gram	Bi-gram	Tri-gram
Accuracy	87.948%	90.6637%	90.995%

5.2 Experiment 2: Implementing Convoluted Neural Networks in Tensor Flow

- We consider the set of reviews which are not spam as positive reviews while the set of reviews which are spam as negative reviews.
- We do not implement any pre trained word2vec or doc2vec, instead we clean the strings and learn embedding form the scratch.
- We build a vocabulary index and map each word to a vector. Thus our whole data files consists of sentences which are converted to a vector of integers.

5.2.1 Model

Figure 1:


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

The model architecture, shown in figure 1, is a slight variant of the CNN architecture of Collobert et al. (2011). Let xi ∈ R k be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n (padded where necessary) is represented as

$x_{1:n} = x_1 \oplus x_2 \oplus . . . \oplus x_n$,

where ⊕ is the concatenation operator.

In general, let $x_{i:i+j}$ refer to the concatenation of words x_i , $x_{i+1}, . . . , x_{i+j}$. A convolution operation involves a filter $w \in R^{hk}$, which is applied to a window of h words to produce a new feature. For example, a feature ci is generated from a window of words

$x_{i:i+h-1}$ by $c_i = f(w \cdot x_{i:i+h-1} + b)$.

Here b ∈ R is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence {x1:h, x2:h+1, . . . , xn−h+1:n} to produce a feature map

$c = [c_1, c_2, . . . , c_{n-h+1}]$, (3) with $c \in R^{n-h+1}$.

We then apply a max-overtime pooling operation (Collobert et al., 2011) over the feature map and take the maximum value cˆ = max{c} as the feature corresponding to this filter. The idea is to capture the most important feature—one with the highest value—for each feature map. This pooling scheme naturally deals with variable sentence lengths. We have described the process by which one feature is extracted from one filter. The model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over labels.

5.2.2 IMPLEMENTATION

1. **Defining the network**
   - We define the embedding followed by convolution and the max pooling layers and the softmax regression layer.
   - The embedding layer is essentially a 2D table consists of the vector of words.
   - While initializing the tensor, we must realise that the tensor tries to perform on GPU. In absence of one, we specify the tensor runs on the CPU.
   - The *allow_soft_placement* setting allows TensorFlow to fall back on a device with a certain operation implemented when the preferred device doesn’t exist.
   - For example, if our code places an operation on a GPU and we run the code on a machine without GPU, not using allow_soft_placement would result in an error.
   - If *log_device_placement* is set, TensorFlow log on which devices (CPU or GPU) it places operations.
   - Our project implements only single channel so we manually give the size
   - Next we build our convolutional layers followed by max-pooling. Since we use filters of different sizes, each convolution produces tensors of different shapes we need to iterate through them. We create a layer for each of them, and then merge the results into one big feature vector.

2. **Dropout Layer**
   To reduce overfitting, we will apply dropout before the readout layer. We create a placeholder for the probability that a neuron's output is kept during dropout. This allows us to turn dropout on during training, and turn it off during testing. TensorFlow's tf.nn.dropout op automatically handles scaling neuron outputs in addition to masking them, so dropout just works without any additional scaling. In our project we initialise the dropout probability to 0.5

3. **L2 regularisation**
   The avoid overfitting the data, we use L2 regularisation. However in our project, the l2 regularisation is initialised to 0.

4. **Loss and Accuracy**

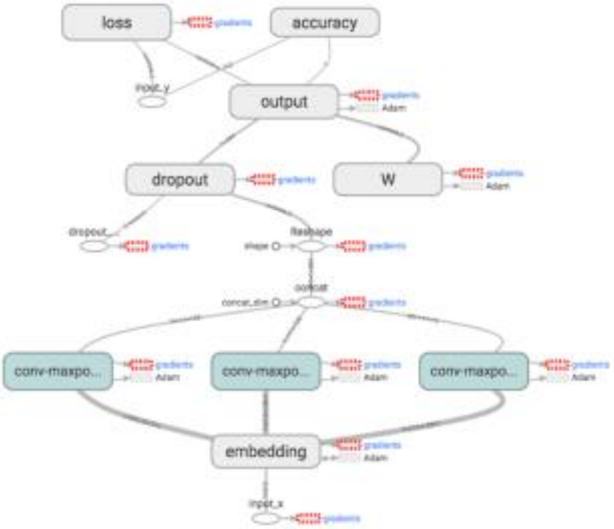
- The *tf.nn.softmax_cross_entropy_with_logits* is a convenience function that calculates the cross-entropy loss for each class, given our scores and the correct input labels.
- We then take the mean of the losses. We could also use the sum, but that makes it harder to compare the loss across different batch sizes and train/dev data.
- To minimise the loss of the network we use the Adam optimiser.
- We also define an expression for the accuracy, which is a useful quantity to keep track of during training and testing.

5. Summaries and Checkpoints

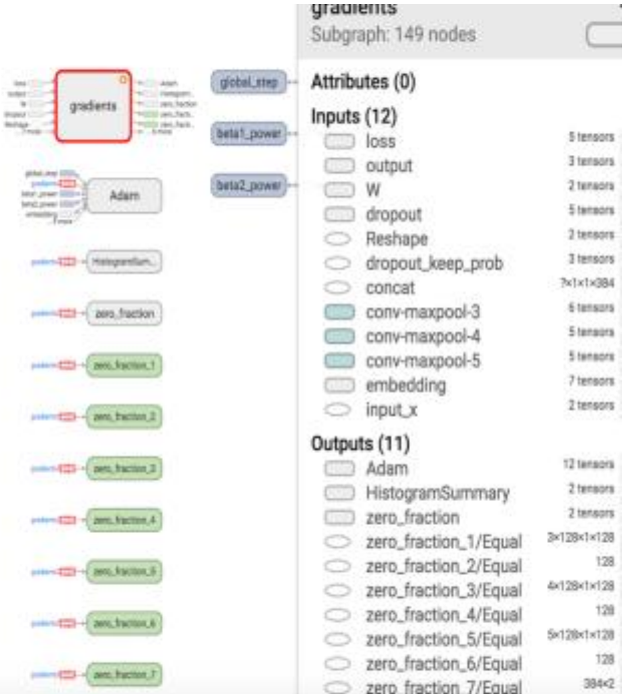
- TensorFlow has a concept of a summaries, which allow you to keep track of and visualize various quantities during training and evaluation. For example, you probably want to keep track of how your loss and accuracy evolve over time. You can also keep track of more complex quantities, such as histograms of layer activations. Summaries are serialized objects, and they are written to disk using a SummaryWriter.
- We implement checkpoints in tensor flow which will help us to restore the training at any point later. Checkpoints can be used to continue training at a later point, or to pick the best parameters setting using early stopping. Checkpoints are created using a Saver object.

5.2.3 VISUALISING THE NETWORK

The results of the project can be visualised on the tensor board with the logdir of the summaries. The tensor board visualization is hosted on port 6006.
The CNN of our project can be visualised in the following figures. This is a neural network with many neurons.



.



gradients
Subgraph: 149 nodes

Attributes (0)

Inputs (12)

loss	5 tensors
output	3 tensors
W	2 tensors
dropout	5 tensors
Reshape	2 tensors
dropout_keep_prob	3 tensors
concat	?×1×1×384
conv-maxpool-3	6 tensors
conv-maxpool-4	5 tensors
conv-maxpool-5	5 tensors
embedding	7 tensors
input_x	2 tensors

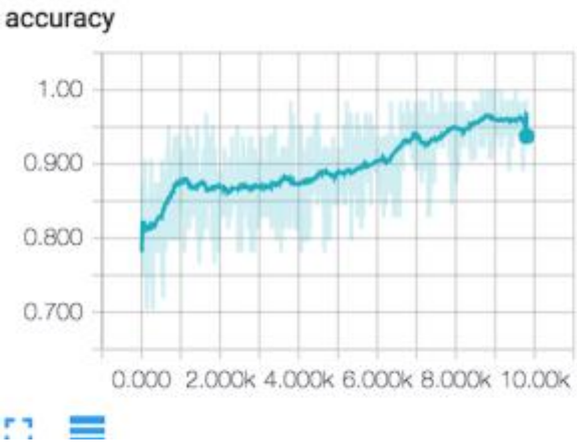
Outputs (11)

Adam	12 tensors
HistogramSummary	2 tensors
zero_fraction	2 tensors
zero_fraction_1/Equal	3×128×1×128
zero_fraction_2/Equal	128
zero_fraction_3/Equal	4×128×1×128
zero_fraction_4/Equal	128
zero_fraction_5/Equal	5×128×1×128
zero_fraction_6/Equal	128
zero_fraction_7/Equal	384×2

5.2.3.2 GRAPHS:

OUTPUT:

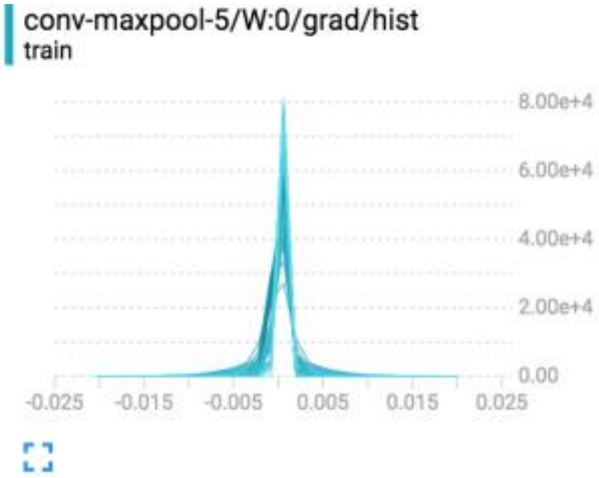
The graph is drawn with the output on the y-axis and the number of epochs on x-axis.



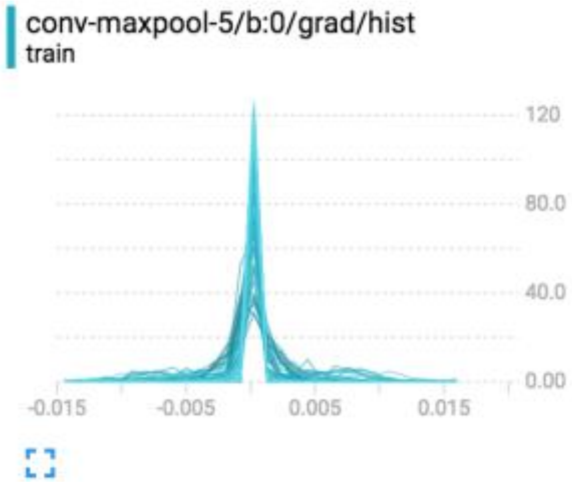
Histogram

The histogram for a conv-max pool layer can be visualized as follows:

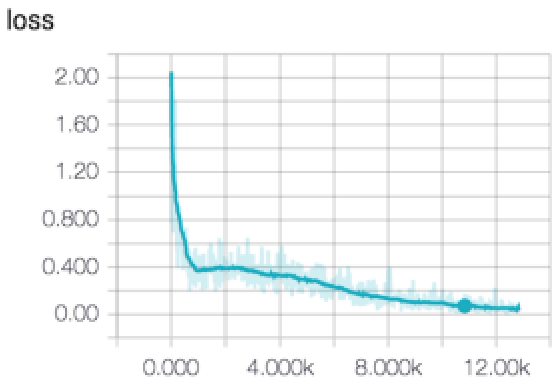
For weight vector:



For bias vector:

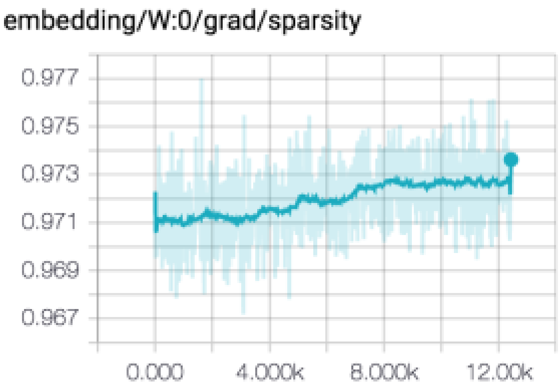


Loss :



Embedding

The embedding graph over epochs can be observed as follows:



Accuracy

Convoluted Neural Networks with TensorFlow generate maximum accuracy compared to other algorithms. The accuracy obtained for our project is ***0.98386*** which is around ***98.386%***

6. Conclusion and Future Work

In the present work we have described a series of experiments with convolutional neural networks implemented in tensor flow. Despite little tuning of hyperparameters, a simple CNN with one layer of convolution performs remarkably well. Our results observe that the Deep Learning concepts yield remarkable results.

However, the project can be made even more efficient by enabling the GPU in the embedding layer, thus enabling the code to run on a GPU. The filter sizes can be varied and tested. The number of epochs can be increased. The L2 regularization norm has been 0 in our code, constraints can be added to it to enable sufficient coding.

- **References**

[1] http://jorditorres.org/first-contact-with-tensorflow/

[2] file:///Users/suchivedala/Downloads/10534-46457-1-PB.pdf

[3] http://www.www2015.it/documents/proceedings/proceedings/p173.pdf

[4] https://www.cs.uic.edu/~liub/FBS/fake-reviews.html

[5] https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

[6] https://arxiv.org/pdf/1510.03820v4.pdf

[7] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[8] https://www.tensorflow.org/versions/master/how_tos/summaries_and_tensorboard/index.html#tensorboard-visualizing-learning

[9] https://journalofbigdata.springeropen.com/articles/10.1186/s40537-015-0029-9

[10] http://www.pyimagesearch.com/2016/09/12/softmax-classifiers-explained/

[11] https://www.researchgate.net/publication/303499094_Fake_Review_Detection_From_a_Product_Review_Using_Modified_Method_of_Iterative_Computation_Framework