Learned BloomFilter

Final Report

Sai Sahana Bhargavi Byrapu, Jagruthi Ashwath
George Mason University
sbyrapu@gmu.edu, jashwath@gmu.edu

Abstract

A Traditional Bloom Filter neither consider distribution of keys(Malicious) nor how they differ from non-keys(non-malicious). Learned model helps to define and understand the distribution of data and overall advances the performance of the Traditional Bloom Filter and also To minimize index space and the number of false positives.Bloom Filter has zero False negative rate and specific False Positive rate for the known data. Learned model ensures specific False Positive Rate for realistic queries as well.we followed the two approaches where in first approach Bloom Filter is implemented as a Classification problem and using Model Hashes in second approach and further compare the results with traditional bloom filter.

1 Introduction

In real world examples the data distribution is not known and hence advanced solutions are needed to be designed and hence with the help of Machine learning Techniques it enables us to implement a model that learns the data distribution and enhances the functionalities of Traditional Index structures and Optimizes them in terms of latency and memory with reduced costs. we implement a model that predicts the data distribution and enhances the traditional bloom filter with improved accuracy and with advanced features . Learned Bloom Filter understand the distribution of data and ensures specific False Positive Rate for realistic queries as well assumption that future queries are retrieved from the same distribution. A Bloom Filter is a data structure that tells if the key exists in the set or not.

Usually Traditional Bloom Filter's do not consider data distribution and how they differ from malicious (keys)and non-Malicious data(Non-keys). The Learned Model provides further performance enhancement by analysing the data distribution and predicting the right output with minimal discrepancy.In classification approach,the CNN model is used as binary probabilistic classifier to distinguish keys and non-keys based on threshold values and Bloom filters with model-hashes a approach that uses a procedure to maximize collisions between malicious and non-malicious urls while limiting collisions among malicious and non-malicious URLs. The classification approach is also tested on CIFAR-10 dataset an Image Classification Dataset and compared with Traditional Bloom Filter.

Section 2 describes the Approaches of the Model's and its implementation Section 3 demonstrates the results calculated for classification Approach and comparison with traditional Bloom Filter. Section 4 describes the results computed for Hash Model and comparison it with traditional Bloom Filter. Section 5 describes the results computed for CIFAR-10 and comparison it with traditional Bloom Filter. Section 6 describes the Environment Setup . Section 7 Provides the Conclusion. Section 8 States the Future Work. Section 9 contains the Metadata. Section 10 contains the References used for the project.

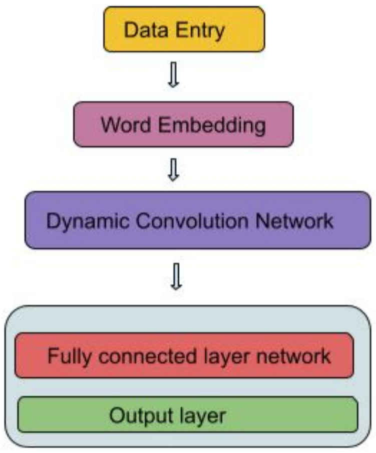
2 Approach

To minimize index space and the number of false positives, there are two potential ways to replace existence indexes with learned models.The two Approaches are Bloom Filters as classification problem and Bloom Filters with Model Hashes .

Data Entry: Data set used is a list of Malicious and Non-malcious URLS where each record consists of an URL and which consists of type such as phishing,benign,defacement. The benign type records are encoded to 1(Malicious) and the rest as 0(non-malicious) and used for further Approaches .

2.1 Bloom filters as Classification problem:

In order to build a build Binary probabilistic classifier that predicts if a query 'x' is a key (malicious) or non-key (non - malicious),we have used a Conventional neural network Model (CNN).



Word Embedding: data is pre-processed using word embedding -a language modeling technique that converts words into real-number vectors and represent in several dimensions.we utilized CBOW(Continuous Bag of Words) architecture of Word2Vec to extract the vocabulary length for the model and encoded the strings into integers as per vocabulary size and further, to maintain the uniform length, each encoded document is padded with zeros equivalent to maximum length of the vector. Embedding layer with respective vocabulary size , dimension 8 and maximum length of the vector is added to the model. CNN Model consisting of the following components:

Dynamic Convolution Network(DCNN):: This layer extracts features from the input data.The output of the network's top layer is fed into the network's next layer during training. The URL is entered into the input layer and converted to a vector expression suitable for embedding in the embedding layer.Based on the input length and the current convolutional layer, the DCNN can alter parameters and extract features over a wider range.A conservative CNN configuration is used with 32 filters (parallel fields for processing words) and a kernel size of 8 with a rectified linear ('relu') activation function. This is followed by a pooling layer that reduces the output of the convolutional layer by half. The data is then loaded into the Fully Connected layer for further rounds.

Fully Connected Layer: In this layer data is flattened ,dense and passed on to the output layer. i.e., the 2D output from the CNN part of the model is flattened to one long 2D vector to represent the 'features' extracted by the CNN. The back-end of the model is a standard Multilayer Perceptron layer to interpret the CNN features. The output layer uses a sigmoid activation function to output a value between 0 and 1 for the malicious and non-malicious urls. We use a binary cross entropy loss function as the problem we are learning is a binary classification problem. The efficient Adam implementation of stochastic gradient descent is used and we keep track of accuracy in addition to loss during training. The model is trained for 1 epochs, or 1 pass through the training data. To build Binary probabilistic classifier that predicts if a query 'x' is a key or non-key. For Strings, we used convolutional neural network(CNN), distribution can be represented as, D = (xi,yi = 1)|xi K (xi,yi = 0)|xi U . Where K is a set of keys (malicious) ,U is a set of non-keys(non-malicious).We Choose a Threshold value 'T' such that the key above that value is classified as exists.To preserve zero false negative rate.,we consider K = x K|f (x) < to be the set of false negatives from f and create a Bloom filter for this subset of keys.When we run our existence index as in diagram , if f (x) , the key is believed to exist; otherwise we check the overflow Bloom filter.To obtain desired FPR(False Positive Rate) , we represent the FPR for model as shown in the formula , U is a held-out set of non-keys.Further, we tune threshold 'T' to achieve False positive rate on U.

2.2 Bloom filters with Model-Hashes

This approach is to follow discretization procedure with the idea to maximize collisions among keys(Malicious) and among non-keys (Non-Malicious URls)while minimizing collisions between keys and non-keys.
Prediction model: Piecewise linear regression with scikit-learn predictors embedded with the bucketization using KBinsDiscretizer(number of bins=2) and then a linear model is then fitted on each bucket is used to train f(x) maps key values with range of [0,1] where x is a key belongs to set K.
Insertion of keys: hash function d =[f (x) m] computes the index of the particular predicted value and that respective index in the bitmap M marked as 1,M[[mf (x)]]=1.Here, for the given dataset, x belongs to set of keys.
False Positive Rate: consider the non-keys in the validation set and record the location M[f (x)m] in the bitmap having value of 1 and percentage is computed as FPRm = M[f (x)m]/| U |.here, x non-key belongs to the set of non-malicious urls U.

3 Results (Classification)

Total records: 651191 Labels: 0,1 m :325000
Non-malicious urls(0):428103 Malicious urls(1) :223088
Training records :455833 Testing records :196358

metric	classification	Traditional Bloom filter
Training accuracy	0.97	0.97
Testing accuracy	0.96	0.49
False negative rate	0.001	0.56
False positive rate	0.003	0.34
Memory	4.34 GB	1.23GB
. Query Latency	0.257 sec	0.261s

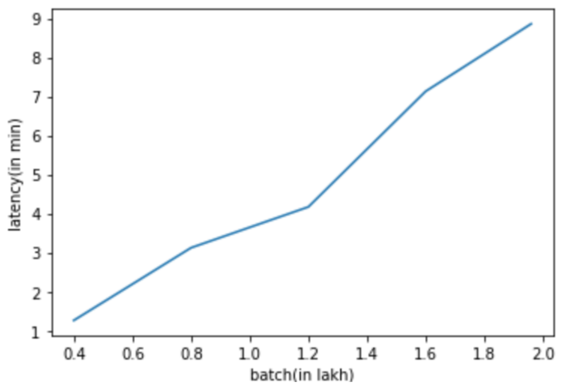


Figure 1: classification

Figure 2: A graph of latency vs Batch size of different input sizes

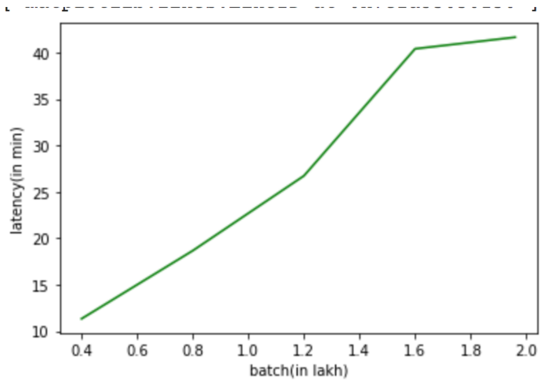


Figure 3: Hash Model

Figure 4: A graph of latency vs Batch size of different

4 Results(Hash Model approach)

Total records: 651191 Labels: 0,1 m :325000
Non-malicious urls(0):428103 Malicious urls(1) :223088
Training records :455833 Testing records :196358

metric	Hash Model	Traditional BF
False positive rate	0.38	0.33
Memory	0.22 GB	1.23 GB
. Query Latency	0.95 sec	0.257 sec

5 Results (CIFAR-10)

It is an Image Classification Dataset consists of 60000 32x32 colour images with 10 classes. We are Splitting the data into 50000 training images and 10000 test images.The labels are Positive and Negative and we have compared it with the

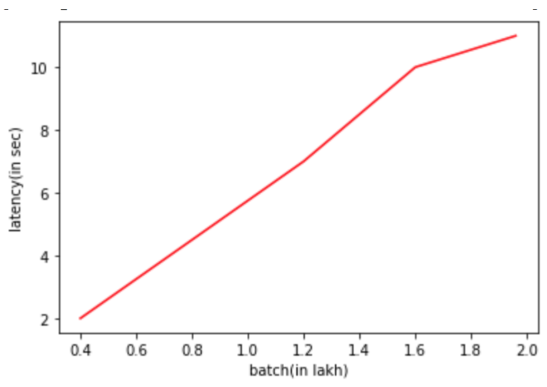


Figure 5: Traditional

Traditional Bloom Filter .

Total records: 60,000
Labels: Positive(0-4) Negative(5-9)
Training records:50,000 Testing records :10,000

metric	classification	Traditional BF
Training accuracy	94	53.2
Testing accuracy	83.4	47.45
False negative rate	0.13	0.18
False positive rate	0.099	0.15
Memory	5.35 GB	2.78GB
Query Latency	0.415 sec	0.464sec

6 Experimental environment

Environment is based on the Mac and Windows OS.The processor is Google Compute Engine backend (GPU), the memory is 8 GB, the programming language is Python 3, and the scikit-learn libraries for machine learning in python.

7 Conclusion

Data Distribution : The Data Distribution is essential for indexes and efficiency of the prediction of data as it plays an important role in how the prediction would work and the results are dependent on these predictions.

Applications : Machine learning is a study which trains a model to act smartly and produce accurate results .Not only does it make the model faster but also ensures data distribution and is used for large Datasets .The applications of Machine learning models where we have used neural networks CNN (Convolution neural networks) helps us to train the model and predict accurately also helps us to calculate the false positive rate for both the approaches.

Learned Bloom Filter:We can see how the learned bloom filter differentiates among keys and non-keys and the performance in both classification and hash model approaches compared to baseline traditional bloom filter

8 FutureWork

Other Machine Learning Models : Possibility of using different machine learning models like RNN to implement the approaches and Compare it with CNN to analyse the differences .

Programming Languages :To explore more methodologies and languages like C++ for implementation in-order to reduce the overhead caused by Tensor-flow deep learning frameworks.

Different DataSets :To test for more Real -World Datasets such as MNIST and Letor to understand how a model would work with bloom filter in terms of false positive, false negative, latency and memory metrics.

Web Application : We Plan to make a front End of the Approach with a website where a page to enter the URLS and the results to be predicted and the Back-end being python programming.

Traditional bloom filters have some specific false negative rate compared to ideal zero false negative rate because of data and bloom filter size constraints and we will focus on fine tuning the implementation further to match with the original negative rate.

9 Metadata

The presentation of the project can be found at:

-https://gmuedu.sharepoint.com/:v:/s/CS571-GRP/EZHbriju-E9BikJWJNIEbSkB2E42FTlEHiXd1_Qdf7wX0Q?e=Sw9MDa

The code/data of the project can be found at:

https://github.com/SahanaByrapu/CS571.git

10 References

[1] Tim Kraska [MIT], Alex Beutel [Google], Ed H. Chi [Google], Jeffrey Dean [Google], Neoklis Polyzotis [Google]-The Case for Learned Index Structures, SIGMOD’18, June 10-15, 2018, Houston, TX, USA
[2]https://dl.acm.org/doi/10.1145/3183713.3196909
[3]https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset
[4]https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
[5]https://www.geeksforgeeks.org/cifar-10-image-classification-in-tensorflow/
[6]https://dl.acm.org/doi/pdf/10.1145/3371158.3371171
[7]A Malicious URL Detection Model Based on Convolutional Neural Network (hindawi.com) https://www.hindawi.com/journals/scn/2021/5518528/