# SPAM DETECTION BY USING NETWORK AND TEXT EMBEDDING APPROACHES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CENNET MERVE YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

AUGUST 2019

Approval of the thesis:

**SPAM DETECTION BY USING NETWORK AND TEXT EMBEDDING APPROACHES**

submitted by **CENNET MERVE YILMAZ** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _____

Assoc. Prof. Dr. Cem İyigün
Supervisor, **Industrial Engineering, METU** _____

Assist. Prof. Dr. Ahmet Onur Durahim
Co-Supervisor, **Management Information Systems, Boğaziçi University** _____

**Examining Committee Members:**

Assoc. Prof. Dr. Altan Koçyiğit
Informatics Institute, METU _____

Assoc. Prof. Dr. Cem İyigün
Industrial Engineering, METU _____

Assoc. Prof. Dr. Pekin Erhan Eren
Informatics Institute, METU _____

Assist. Prof. Dr. Ahmet Onur Durahim
Management Information Systems, Boğaziçi University _____

Assist. Prof. Dr. Bahar Çavdar
Industrial Engineering, METU _____

Date: 21.08.2019

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Surname: Cennet Merve Yılmaz

Signature:

# ABSTRACT

## SPAM DETECTION BY USING NETWORK AND TEXT EMBEDDING APPROACHES

Yılmaz, Cennet Merve
Master of Science, Industrial Engineering
Supervisor: Assoc. Prof. Dr. Cem İyigün
Co-Supervisor: Assist. Prof. Dr. Ahmet Onur Durahim

August 2019, 89 pages

Authenticity and reliability of the information spread over the cyberspace is becoming increasingly important, especially in e-commerce. This is because potential customers check reviews and customer feedbacks online before making a purchasing decision. Although this information is easily accessible through related websites, lack of verification of the authenticity of these reviews raises concerns about their reliability. Besides, fraudulent users disseminate disinformation to deceive people into acting against their interest. So, detection of fake and unreliable reviews is a crucial problem that must be addressed.

In this study, we analyze and compare three different spam review detection approaches, DocRep, NodeRep and SPR2EP, that utilize review text only, network information only and the one that is proposed in this study that incorporates knowledge extracted from the textual content of the reviews with information obtained by exploiting the underlying reviewer-product network structure, respectively. One of the important contributions of this study is the proposed framework, SPR2EP, is that it benefits from both review text and network information. In SPR2EP approach, first, feature vectors are learned for each review, reviewer and product by utilizing state-of-the-art algorithms developed for learning document and node embeddings, and then

these are fed into a classifier to identify opinion spam. It minimizes the feature engineering effort. The effectiveness of our framework approaches that utilize network embeddings over existing techniques on detecting spam reviews is demonstrated in three different data sets containing online reviews.

Keywords: Review Spam Detection, Feature Learning, Document and Node Embeddings, Web mining

# ÖZ

## SOSYAL AĞ ANALİZİ VE METİN MADENCİLİĞİ KULLANARAK YANILTICI YORUMLARI TESPİT ETME

Yılmaz, Cennet Merve
Yüksek Lisans, Endüstri Mühendisliği
Tez Danışmanı: Doç. Dr. Cem İyigün
Ortak Tez Danışmanı: Dr. Öğr. Üyesi Ahmet Onur Durahim

Ağustos 2019, 89 sayfa

Siber ortama yayılmış bilgilerin doğruluğu ve tutarlı olmasının önemi her gecen gün daha da önemli hale gelmektedir. Potansiyel müşterilerin satın alma işlemleri öncesinde müşteri yorumlarını ve önerilerini internet üzerinden görüntüleyebilir olması, bilgi tutarlığının önemini özellikle e-ticaret (sanal ticaret) alanında daha da arttırmaktadır. Günümüzde bilginin bu kadar kolay erişilebilir olmasının yanında, ulaşılan bilginin doğruluğuyla ilgili kontrollerin yetersiz oluşu, bu bilginin güvenilir olup olmadığıyla ilgili kaygılar oluşturmaktadır. Ayrıca, hileli kullanıcılar spam(istenmeyen) bilgiler paylaşarak insanları yanlış yönlendirmektedir. Bu sebeplerle, güvenlik araştırmacıları tarafından hatalı ya da yanlış kullanıcı yorumlarının denetlenmesi kritik bir problem haline dönüşmüştür.

Bu çalışmada, DocRep, NodeRep ve SPR2EP yaklaşımlarıyla spam kullanıcı yorumlarını analiz ettik ve bu yaklaşımların farklarını inceledik. Çalışmamız için kullanıcı yorum metinlerinden ve sosyal ağ bağlantı bilgilerinden yararlandık. Kullandığımız yöntemler, yorum metinlerinden elde edilen bilgi ve kullanıcı – ürün ağ yapısı bilgisini birleştirmektedir. Kullanıcıların, yazdıkları yorumlar ve yorumladıkları ürün arasında makine öğrenmesi algoritmaları kullanılarak, her yorum, kullanıcı ve ürün için, bu birleşenlerin özelliklerini yansıtan vektörler oluşturuldu ve

oluşturulan bu vektörler kullanıcı yorumlarının spam olup olmadığını belirlemek için sınıflandırıcı algoritmaların girdisi olarak kullanıldı. Yapılan karşılaştırmalı çalışma neticesinde, sadece sosyal ağlardan yararlanan yaklaşım kullanıcı yorumu verişinde diğer yaklaşımlara göre daha yüksek performans elde edilmiştir

Anahtar Kelimeler: Yanıltıcı yorum Tespiti, Makine Öğrenmesi, Metin Madenciliği, Sosyal Ağ Analizi

To my family, Burak & Ayşe

## ACKNOWLEDGEMENTS

I would like to gratefully acknowledge various people who have been journeyed with me in recent years as I worked on this thesis. Firstly, I would like to thank Assoc. Prof. Dr. Cem İyigün and Assist. Prof. Dr. Ahment Onur Durahim for their expertise, assistance, guidance and patience throughout the process of writing this study. Without their help, patience and tolerance, this work would not have been possible. I would like to thank my committee members Assoc. Prof. Dr. Altan Koçyiğit, Assoc. Prof. Dr. Pekin Erhan Eren and Assist. Prof. Dr. Bahar Çavdar for their support, suggestions, and encouragement. Secondly, I am so grateful my family namely my parents, Hatice and Ali, and my brothers, Mustafa and Ahmet who helped and encouraged me. I would like to thank my friends, especially Ayşe, Gizem, Çiğdem and Pınar, for their attention and support in this way. It was a very long way for me. Last but not least, I want to express my deepest gratitude to Burak for all kinds of support.

## TABLE OF CONTENTS

## LIST OF TABLES

TABLES

## LIST OF FIGURES

FIGURES

# CHAPTER 1

## INTRODUCTION

The dissemination of disinformation and the difficulty of distinguishing legitimate information from fake one such as fake news and reviews is a vital problem in today's internet world. Nevertheless, the Internet is the most important source of information, particularly the shared experiences of people on products and services, and the size and value of this information increase day by day. In order to get insights from people's experiences, one could find excessive information from the reviews and articles written on a particular topic/product on different websites, social media accounts and blog posts. While the popularity of the Internet grows as an information source, it also leads to growing diffusion of disinformation, and particularly as the number of reviews increases, fake/spam reviews also increase. Besides, the detection of spam reviews rendered difficult as spammers and researchers working on detecting spam reviews play a cat and mouse game. Reviews written on products and services have a great effect on purchasing decisions of customers, and most of the time spammers diffuse disinformation to exploit this fact and deceive customers to act against their own interest. As a consequence of this situation, people always consult reviews but do not trust them. The 70 percent of the American people check reviews before purchasing, but only 59 percent trust recommendations according to the market research conducted by Mintel [1]. Also, Yelp [2] declared that the percentage of fake reviews increased to 20% in 2013 while that ratio was 5% in 2006.

When some spam actions aim to cheat people and reach their private information, some such actions mislead people into purchasing products and services they would not otherwise buy. Jindal and Liu [3] stated that spam reviews can be split into three categories as untruthful reviews, reviews on brands and non-reviews. Reviews on

brands aim to mislead users and gives not useful feedbacks on specific brand or product.

Non-reviews contain unrelated text or advertisements. Therefore, detection of non-reviews and reviews on a brand by buyers are easier than the detection of untruthful reviews. Following are the two YELP reviews, where one of them is fake and the other is authentic.

*Review 1: "Noise, noise, noise! Unbelievable! Between the overly loud music and the tons of loud motorcycles, people screaming - this is ridiculous!"*

*Review 2: "The only place inside the Loop that you can stay for $55/night. Also, the only place you can have a picnic dinner and get a little frisky on the 17th floor roof and then wake up in your room the next morning to an army of ants going in on your picnic leftovers."*

It is hard to distinguish these two reviews whether they are fake or not for regular internet users. However, it is possible to detect and classify these reviews with machine learning techniques with the help of information other than the textual content itself. The only information exposed to internet users is the review text and it is nearly impossible for them to identify whether a particular review is fake or not by just analyzing this raw text. Actually, textual content is not the only information that can be obtained from online reviews. Reviews also contain metadata, such as IP address of the source and session duration, which is not accessible by internet users.

Review metadata is actually a fruitful source of information which can be utilized in spam detection. Reviewer of the review and the reviewed product are considered as the most important attributes of reviews [4]. Some of the reviews are written by the paid reviewers and they are mostly paid by product/service providers. Paid reviewers' accounts are typically new accounts compared to the legitimate user accounts, and paid reviewers have more than one account and use them to promote or demote a particular product/service. As a result, how many reviews written by the same account, IP address and historic actions of account can be utilized as the features through which

one may improve the detection of spam accounts and reviews. Additionally, product/service specific features including the price range of the product and sales amount of the product, average review rating of the product, average feature specific rating of the product and the number of reviews written for a product can help to identify spam accounts or reviews.

Spam detection studies mostly employ data/text mining and network based approaches. There are also studies that aim to combine both data/text mining and network based techniques. In this respect, one of the earliest studies in literature belongs to Jindal and Liu [5] where a logistic regression based spam classifier is built which is fed by review features like content characteristics, rating, title, and review feedback. Besides, there are several works based on text mining approaches that focus on unigram, bigram and trigram bag of words features [6,8]. On the other hand, proposed network based approaches use the relational structure of review network and utilize graph theoretical approaches [9,11].

This study mainly aims to compare and aggregate approaches proposed for detecting untruthful reviews by combining and utilizing features generated via textual and network based approaches. In order to accomplish this task, three different spam review approach are created and compared by utilizing the features generated from the review text and reviewer-product network via adapting document and node embedding algorithms, namely Doc2vec and Node2vec, respectively. We named our classifiers as DocRep, NodeRep and SPR2EP based on the algorithms that we used. DocRep and NodeRep use Doc2vec and Node2vec algorithms, respectively. SPR2EP combines the outputs of these two algorithms, Doc2vec and Node2vec. SPR2EP stands for Semi-Supervised Spam Review Detection Framework. Doc2vec algorithm is proposed by Le and Mikolov [12] and used for generating document embeddings from their textual content. Doc2vec gives an opportunity to assign a single vector representation to each review that can be utilized to distinguish fake reviews from authentic ones. Similarly, Node2vec algorithm [13] can be used to generate node embeddings in a network by taking into account the network connectivity. Node2vec

provides us vector representation for every reviewer and product through the underlying reviewer-product network which is constructed by using review metadata. Every review written on a product with a given rating represents the relation between a reviewer and a product in the reviewer-product network of the proposed framework. The reviewer of the review and the reviewed product are considered as neighboring nodes in the network and the corresponding rating of the review is taken as the weight of the edge between these neighboring nodes. Resulting network is then fed into Node2vec algorithm to transform it to lower dimensional feature space. The reasoning behind employing a node embedding algorithm like Node2vec in spam detection is that spam reviewers mostly write spam reviews, and it is assumed that spam reviewers and products show similar network characteristics.

Both Doc2vec and Node2vec algorithms are unsupervised learning algorithms. And, this is especially advantageous for spam detection since it may not be an easy task to gather annotated dataset for a task where the labeling of the input data, that is identifying a review as spam or legitimate, is proven to be difficult [4]. Besides, huge amounts of unlabeled review data can be easily obtained from the social media and blog websites.

This study investigates the ways to exploit the predictive capabilities of the vector representations of reviews, reviewers and products obtained by applying Doc2vec and Node2vec algorithms on raw spam review dataset. Those vector representations are fed into various binary classification algorithms, both individually and combined, to understand the individual contributions of the document and node embeddings and to make an assessment of the effectiveness of semi-supervised learning in spam detection. Besides, performances of the proposed approaches are compared with the previous works to appreciate the improvements achieved.

Main contributions of this study can be summarized as follows;

- To the best of our knowledge, dense document and node embeddings are utilized, for the first time, for spam review detection. And, the assessment

of the performances of the classifiers built by using document (review) and node (reviewer and product) embeddings separately and in combination have been provided. Also, results of this work are compared with the performance results of the previous state-of-the-art techniques.

- Semi-supervised learning frameworks are compared where manual feature engineering, which is time consuming and additionally requires expert knowledge, is eliminated by creating review, reviewer and product embeddings in an unsupervised manner. Besides, exploiting the value of big data through unsupervised learning is of utmost importance, and accomplished here via consolidating the advantages of efficiently learning dense and low-dimensional review, reviewer and product feature vectors from huge amounts of unlabeled review data and using them for extracting insights.

# CHAPTER 2

## BACKGROUND AND LITERATURE REVIEW

This study aims to solve spam detection problem with machine learning approaches by discovering patterns in the data. It is possible to categorize machine learning approaches to supervised, unsupervised and semi-supervised learning mainly. In supervised learning, machine learning algorithm is worked on a labeled dataset in which each record has outcome information. This gives opportunity to detect the patterns and relationships between target variable and the rest of the dataset based on information it has already. Classification and regression are supervised algorithms. Classification algorithms aims to predict labels when regression algorithms are used to predict continuous target. Our study benefits from the binary classification algorithms which are Logistic Regression, CART Decision Tree, Gradient Boosting, Random Forest and Multilayer Perceptron (Neural Network). On the other hand, unsupervised learning algorithms learn from the unlabeled dataset by detecting the patterns in the unlabeled data. Clustering algorithms which are k-means, hierarchical and probabilistic clustering, etc. and data dimensionality reduction algorithms which are Singular-Value Decomposition (SVD), Principal Component Analysis (PCA), etc. are unsupervised learning algorithms. Additionally, autoencoders are classified as unsupervised deep learning algorithm. This method is similar to data dimensionality reduction techniques which aim to represent input data in latent vector space. The difference is that it uses many layers to represent input, and updates weights many times between layers. Compatible with the difference arising from the availability of labelled data between supervised and unsupervised learning, semi-supervised learning could be defined based on the availability of labelled data as well. It sits between the supervised and unsupervised learning. Semi-supervised learning takes both labelled and unlabelled data as an input. It uses mostly unlabelled data.

Our study includes three main parts. In the first part, different semi-supervised learning techniques which are based on text and network mining are used. In the second part, classification algorithms which are supervised learning are used to predict labels in the dataset. In the last part, performances of supervised methods which are applied on the data produced in semi-supervised manner are compared. Therefore, this chapter is organized as follows. Text mining and network analysis concepts will be introduced. Data dimensionality reduction is described, and feature embeddings will be introduced. After text and network analysis framework is introduced, it will be described how those concepts are used in feature embedding manner. Spam detection approaches will also be introduced. Classification techniques and performance measures will be described.

## 2.1. Text Mining Approaches

Text mining aims to extract useful information from the unstructured textual data by detecting patterns. It converts unstructural data to structural data. Overall process of the text mining includes the steps: text preproccessing, text transformation, feature selection, text mining methods and evaluation which is shown in Figure 2.1 [14].

### Text Preprocessing

Text preproccesing includes tokenization, stopword removal and stemming steps generally. Prior to those steps, there is need to do some basic data cleaning and normalization operations such as converting all letters to same case, cleaning numbers and punctuations, removing punctuations and links in text.

- **Tokenization:** Tokenization is the process of splitting the given text into smaller pieces called tokens. This step divides whole document into words.

- **Stop word Removal:** In this step, most common words like “the”, “a”, “on”, “all” are removed. This is because, they are not shown any statistical importance or any sentimental effect to model.

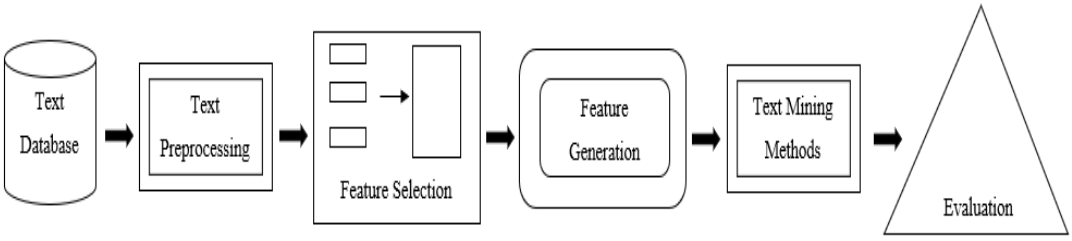



*Figure 2.1* Text mining framework

- **Stemming:** Stemming is used to reduce a word to its root/stem. E.g. Flying, Flew word to Fly. The algorithm proposed by Port, known as a Port’s stemming algorithm is widely used [15].

**Text Transformation / Feature Generation**

In this step, text document is converted into bag of words or vector space document model notation, which can be used for further effective analysis task.

**Feature Selection/Attribute Selection**

This phase mainly performs removing features that are considered irrelevant for mining purpose. This procedure gives advantage of smaller dataset size, less computations and minimum search space required. This step may not be applicable for all text mining procedure and can change algorithm to algorithm.

**Text Mining Methods**

There are different text mining methods as in Data mining had been proposed such as: Clustering, Classification, Information retrieval, Topic discovery, Summarization, Topic extraction, LDA. In this step, the output of the approach is provided. The steps up to now are the prior steps to main text mining algorithm. They help to text mining methods to increase the efficiency and performance of text mining algorithm.

**Interpretation or Evaluation**

In this step, performance of model is measured, and result are assessed based on ROC, Precision & Recall, Accuracy, F1-F2 measure, etc.

### 2.1.1. Word Embeddings

Word embeddings are distributed word representations which are dense as opposed to the shallow representations obtained via bag of words approach. Each dimension of a word embedding encodes latent syntactic and semantic features of the words. They are introduced by Bengio et. al. [16] where the word embeddings are learned by using neural language model. In their study, authors presented a probabilistic approach based on the co-occurrences of the words.

Two recent proposals have become popular for creating the word embeddings, namely Word2vec [17] and Glove [18]. Both of them output word embeddings for each word in a given corpus, and proposed algorithms are based on the co-occurrence of the words in a given context which are used to capture the semantic properties and reflect it in the word embeddings.

Word2vec presents two different algorithms, *"Skip-gram"* and *"Continuous Bag-of-Words (CBOW)"* , and captures the co-occurrences of words in local context windows at any time by maximizing the co-occurrence probability of the target word and the surrounding words inside the context window [17]. On the other hand, *"GloVe"* algorithm constitutes a global co-occurrence matrix of words which includes counts

of co-occurrences of words in the given text in addition to the words considered in their respective context windows [18]. It is stated that Glove is a count-based approach whereas Word2vec is a predictive approach. Another argument is that Word2vec model suffers from ignoring the global word co-occurrence statistics of a given corpus, and it only considers the context windows of the words across the entire corpus [19]. In order to address this problem, Pennington et al. [18], proposed GloVe approach which differs from Word2vec method by taking into account the global statistics of word co-occurrences in a given corpus in addition to the statistics of local context windows.

Word embeddings have extensively been used in research domains that contain natural language processing tasks such as named-entity recognition (NER) and sentiment analysis. In a recent study, Unanue et al. [20] utilized word embeddings together with bidirectional LSTM [26] in order to perform health-domain NER. Santos and Gatti [27], on the other hand, developed a deep convolutional neural network that exploits from character to sentence level information for doing sentiment analysis.

### 2.1.2. Paragraph Embeddings

In text classification tasks, one attempts to categorize documents based on their textual content. So, machine learning algorithms expect feature representations for the documents in order to classify them to their respective categories. In order to use the proposed word embedding algorithms for that purpose, previous studies follow different paths to generate document embeddings such as averaging the words that occur in the document as well as taking weighted average of the words in a document by considering their bag of words scores such as TF-IDF [23]. In order to improve the performance of the document embeddings used in classification tasks computed via combining the word embeddings of the words in the document, Mikolov et. al. [12] proposed Doc2vec algorithm that learns embeddings for word sequences (sentences or paragraphs) known as paragraph embedding. While Word2vec algorithm produces

word embeddings, Doc2vec algorithm constructs paragraph embeddings by keeping semantic and syntactic relations of the words by additionally taking the paragraph structure into account. Similar to Word2vec, Doc2vec also contains two methods; "*Distributed Memory Model of Paragraph Vectors (PV-DM)"* and "*Distributed Bag of Words version of Paragraph Vector (PV-DBOW)*". PV-DM works in the same way with CBOW of Word2vec and uses document vectors to predict its document words. On the other hand, PV-DOW works in the same way with skip-gram method of Word2vec and uses a small window of words and concatenates them with the document vector to predict a document word.

## 2.2. Social Network Analysis Approaches

Social network analysis aims to detect and investigate social structures in networks. Components of network are represented with nodes and edges. Nodes are connected with edges. It is closely related with graph theory. In the next section, graph theory will be introduced to explain social network analysis better.

### 2.2.1. Graph Theory

A graph is a mathematical model defined as an ordered pair $G = (V, E)$, where $V$ is a set of vertices and $E$ is a set of edges such that $E = \{(u, v) \mid u, v \in V\}$. Vertices correspond to entities and edges correspond to relationships between them. Usually, a graph is represented by an adjacency matrix $A$, of dimensions $|V| \times |V|$:

$$A_{ij} = \begin{cases} 1, \text{if } (v_i, v_j) \in E(G) \\ 0, \ otherwise \end{cases}$$

A graph can be of different types and the most important categories are described:

- **undirected**: given a pair of nodes u and v, the edge (u, v) is identical to the one (v, u).

- **directed**: edges have orientations and so E is a set of ordered pair of nodes.
- **weighted**: edges have assigned a number $w_{ij}$, the weight, which can represent different variables, such as a cost, a length or a capacity, depending on the modelled domain. In this case, the adjacency matrix is extended to consider the weights and each entry becomes $a_{ij} = w_{ij}$.
- **multigraph:** multiple edges between each pair of nodes are allowed and labelling for each node and edge is introduced. Formally, it can be defined as $G = (V, E, R)$ where $V$ is a set of vertices, $R$ is a set of relations, and $E$ is a set of labelled edges, $(u, v, r)$, where $u, v \in V$ and $r \in R$.

**Centrality Measures**

In graph theory, there are certain measures which explain graph. In this section, those measures will be introduced.

- **Degree Centrality:** Given a node v, normalized degree centrality is given by where $deg(v)$ is the degree of the node and $N$ is the number of nodes present in the graph.

$$D(v) = \frac{\deg(v)}{N - 1}$$

- **Closeness Centrality:** Given a node $v$, normalized closeness centrality is given by where $d(u, v)$ is the distance between node *u* and *v*.

$$C(v) = \frac{N - 1}{\sum_u d(u, v)}$$

- **Betweenness Centrality:** Given a node *v*, betweenness centrality is given by where $\sigma_{st}(v)$ are the shortest path between *s* and *t* that pass through *v* and $\sigma_{st}$ are all the shortest path between *s* and *t*. [24]

$$B(v) = \sum_{s \neq v \neq t \epsilon V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

### 2.2.2. Network Node Embeddings

Similar to the word and document embedding algorithms, the aim of node embedding algorithm is to learn low-dimensional and dense feature vectors for the nodes from the underlying network by regarding the network structure such as homophily and structural equivalence in an unsupervised manner. These node embeddings then can be used in tasks such as node classification and edge prediction [13].

Early studies in unsupervised tasks [25] and locally linear embedding (LLE) utilize important eigenvectors of the adjacency matrix of a network as feature vector representations of nodes in the network. They set their problem as a dimensionality reduction and solved by applying eigenmaps [26]. More recent methods use neural networks to represent nodes in dense low-dimensional vectors by reflecting the network structure in those vectors. The main idea to preserve the homophily and structural equivalence is to focus on both first-order proximity as well as the higher-order proximity. First-order proximity corresponds the direct connection between nodes and the second-order proximity corresponds the common neighbors of nodes which are not connected directly and so on.

LINE [27] presented a framework where the node embeddings are obtained by concatenating the *first-order* and *second-order proximity* embeddings. DeepWalk is another node embedding algorithm proposed by Perozzi et al. [28] which uses Word2vec's skip-gram algorithm. First, node sequences are generated via random walk, and then these sequences are supplied to the skip-gram algorithm as if they are sentences. Similar to DeepWalk, Node2vec algorithm proposed by Grover and Leskovec [13] also learns node embeddings using skip-gram algorithm based on the sequences generated via random walks. The main difference between Node2vec and DeepWalk is that Node2vec uses biased random walk while generating the node

sequences. The sampling strategy of Node2vec provides flexible neighbourhood sampling strategy which provides smooth interpolation between Breath-first sampling (BFS) and Deep-first sampling (DFS) with biased random walk. In BFS, sampling is based on immediate nodes and it reflects the structural equivalence. In DFS, nodes are sampled sequentially at increasing distance and it corresponds to homophily and macro-view of network. The bias is created in random walk with in-out node parameters which direct the random walk to generate sequences through BFS or DFS fashion.

In the proposed framework, due to the better performances achieved, Node2vec is employed for generating the reviewer and product embeddings.

### 2.3. Spam Detection Approaches

As the amount of opinion reviews shared online grows rapidly, detection of fake reviews becomes more crucial. As a result, spam detection attracts the attention of researchers from both academia and industry. One of the earliest studies on opinion spam detection is done by Jindal and Liu [5]. In their study, ground-truth labels are assumed to be untruthful for the duplicated reviews, whereas non-duplicated reviews are considered as truthful. However, it is possible to say that all duplicate or near-to-duplicate reviews may not be untruthful. This approach is more like plagiarism or duplicate review detection. It is the preferred choice since setting a ground-truth label for untruthful reviews is not an easy task because of the difficulty of differentiating spam reviews from the legitimate ones. In model building, features based on review text, product, and reviewer are used. Information retrieval based evaluation is used in some other studies in order to set ground-truth labels by recruiting human judges [9], [29]. The main drawbacks of using human judgment on spam review labeling are obviously the difficulty of applying this method to large datasets and the questionable trustworthiness of human labeling.

You and Gretzel [30] focused on explaining deceptive opinions with the psychologically relevant linguistic features. They applied a standard statistical test manually to 40 truthful and 42 untruthful hotel review data to show the relation between linguistic features and their corresponding labels. Lim et al. [29] used rating behaviors of reviewers and products and applied regression to identify the spam reviewers. Mukherjee et al. [31] also focused on reviewer behaviors to interpret the YELP filtering algorithm and developed an author spamicity. Bayesian model based on the idea that a group of online users may work together to create spam reviews. They have not used the textual information in their work and claimed that behavioral features are more effective than linguistic features. Fei at al. [32] exploited bursts of reviews by detecting reviewers' co-occurrence in bursts as a Markov Random Field (MRF) and employed Loopy Belief Propagation method to identify the spam reviewers. Cheng et al. [33], in a case study, investigated the characteristics of spam messages and spammer behaviors on a web forum. Bhat and Abulaish [34] proposed a community-based framework to identify spammers in online social networks. Their framework aimed at exploiting social network characteristics of community formation by legitimate users in that respect. Wang et al. [9] proposed utilizing the user-review-product network, and defined scores for trustiness of users, honesty of reviews, and reliability of products to identify online store spam users. On the other hand, Rayana and Akoglu [35] proposed a method that uses a combination of user-review network features in conjunction with the hand-crafted features in their scoring algorithm.

Our study investigates the effectiveness and efficiency of employing previously proposed unsupervised methods used for learning document and node embeddings within a spam review detection framework. Both of these methods aim to generate dense low-dimensional feature vectors for reviews and nodes in the reviewer-product network. In our study, the assessment of the performances obtained from the spam detection models created by feeding these feature vectors into various binary classification algorithms is also performed.

# CHAPTER 3

# PROBLEM DEFINITION AND RELATED WORK

In this study, we propose a solution framework to spam detection problem by using text mining and social network analysis approaches. Therefore, the background information and literature review on spam detection, text mining and social network analysis are given in previous chapter. In this chapter, we firstly explain our problem definition and introduce related work and approach to our problem.

## 3.1. Problem Definition

As discussed in previous chapters, online opinion reviews have an essential effect on the purchasing behaviour of customers. Consequently, the internet is the prior source to provide this information, especially the experience of other customers on products and services. Opinion based information on products and services grow day by day in the internet environment. Although this growth seems as useful for potential buyers, another issue arises here; spam behaviour and the reliability of opinion reviews. Most of the potential buyers have already realized this issue and they read the opinion reviews on the product/service before purchasing but the majority of potential buyers do not trust the online opinion reviews [1]. By addressing this problem, spam detection algorithms aim to distinguish genuine and fake reviews to help potential buyers on their product selection decisions. In literature, there are early studies which take spam detection problem as plagiarism or duplicated review selection problem. Also, some approaches apply human judgement. However, it is hard to distinguish online reviews by human-being, and it is not an efficient way of labeling regarding time and workforce. As a result, the trustworthiness of human labelling and application to large data sets make using human judgement for spam detection questionable. In literature, there are also different approaches which use psychologically relevant linguistic

features or behavioral features. Psychologically relevant linguistic features are created by using text data. Behavioral features benefit from the network data with Bayesian approach. The assumption behind this approach is that online user groups work together to create spam reviews. There are also community-based approaches which benefit from social networks.

Consequently, spam detection literature is built on mostly community-based and text-based approaches. Community-based approaches use social network and graph theory approaches. Text-based methods benefit from linguistic, semantic features of spam data. Here, there is a need to state that the gaps in spam detection literature: there are very few approaches which benefit from community-based and text-based approaches at the same time in spam detection literature. In literature, it is observed that spam detection problem is addressed to either community-based or text-based solutions. In this study, we propose community and text based solutions at the same time to the spam detection problem. This difference is a significant contribution to literature. Some of the text-based approaches use human-judgement labelling and this is not feasible for large data sets. Our approach is applicable to large data sets also. When we compare our approach with the approaches which use community-based and text-based solutions at the same time, the difference is feature engineering effort. In those approaches, there is a lot of effort for hand-crafted model features which benefit from the user-product network and review text data. However, one of the most critical superiority of our approach is feature learning. In this study, user-product network and review information are converted to low-dimensional vector space thanks to Node2vec and Doc2vec algorithms.

Our work proposes a seamless and modular framework to online opinion spam problem by utilizing both network information and text information. We use feature learning algorithms to reduce large data to low dimensional vector space. Those low dimensional vector spaces for network and text information can be used together or separately as an input of classification algorithms. Additionally, our study tries to

answer which of the network and text algorithms performs better in which classification algorithms.

We propose our approach to spam detection problem by considering the gaps in literature. Additionally, we tested our approach, conducted sensitivity analysis with different parameter sets and compared our approach with state-of-art approaches in literature. In the light of this information, our work contributions can be listed as follows:

- Text data and social network data are used together as a model input
- Feature embeddings approach used first time according to our knowledge in spam detection
- It can be used for large data sets
- It minimizes the feature engineering effort and creates features automatically
- It can be used real-time labeling

By considering all those gaps, our solution to this problem is published in ASONAM 2018 [36]. The approach in the published work is the same in this thesis. However, the work is extended with the detailed parameter and classifier performance analyses in this thesis. The performance of SPR2EP approach is measured with the default model parameters and its superior performance is proven with respect to other state-of-art spam detection techniques in ASONAM 2018 publication [36]. In addition to our publication, we search for better parameter setting and to improve the predictability power of our proposed framework in this work. This study also investigates the contribution of network and text embeddings to model performance. Therefore, we introduce three different approaches considered for spam review detection in this work. In the text-only approach, DocRep, representations are learned for the reviews by utilizing a document representation learning algorithm (Doc2vec) and used as features for creating spam review detection model. In the network-only

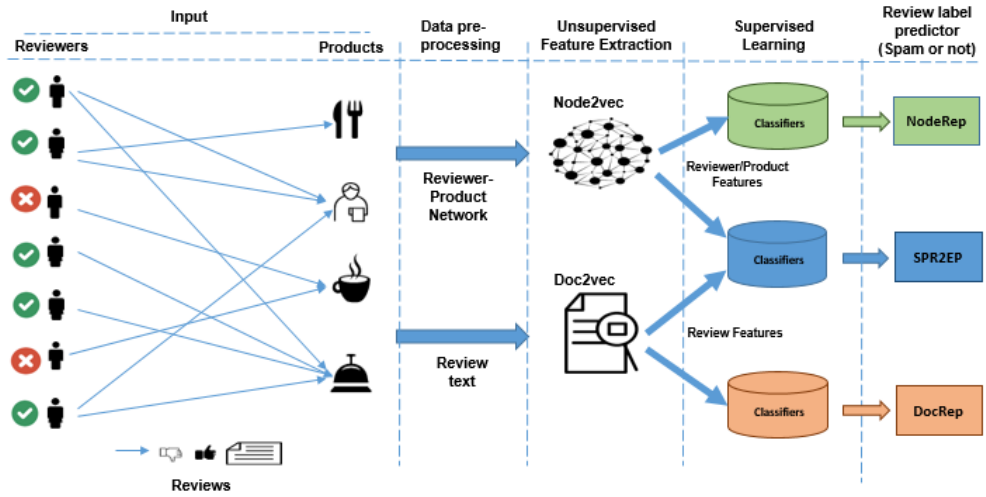



*Figure 3.1.* Overview of SPR2EP

approach, NodeRep, representations are learned for the network nodes via a node representation learning algorithm (Node2vec) and used as features for creating a classifier. Lastly, a recently proposed framework SPR2EP [36] depicted in Figure 3.1 is analyzed. In this framework, reviewer, restaurant/hotel and review feature representations[1] are learned and used in detecting spam reviews. First of all, reviewer-product network is generated by creating a link between reviewers and products if a reviewer has written a review on a product. All those approaches create embeddings in unsupervised manner and their outputs are fed into various classification algorithms to build classifiers for spam review detection.

The performance of machine learning methods depends on preprocessing and transformation of input data. Feature engineering step is the prior and one of the most important steps of modelling. Mostly, feature engineering methods include manual and human intervention steps to improve classifier model performance. In spite of this, representation learning extracts useful common information automatically for feature detection or classification. Also, we investigated the effect of classification algorithm to performance results. In this work, Logistic Regression, Decision Tree, Random

[1] Representations and embeddings have been used interchangeably throughout this study.

Forest, Gradient Boosting Machine and Neural Networks are used as binary classification algorithms. In the next sections, those representation learning and classification algorithms will be explained.

## 3.2. Representation Learning Algorithms

### 3.2.1. Word2vec

Word2vec is a group of related models that are used to produce word embeddings. This study does not utilize from Word2vec algorithm, but Word2vec algorithm is the ancestor algorithm of Doc2vec. Therefore, Word2vec is explained prior to Doc2vec section. In Word2vec two-layer neural networks are used to construct linguistic context of words. Word2vec requires as its input a big corpus of text and generates a vector space, typically several hundred dimensions, with a corresponding vector in the space being allocated to each distinctive phrase in the corpus. Word2vec was developed at Google in 2013 by a research team led by Tomas Mikolov [17].

Word similarity is the measure of the proximity of two-word vectors. Also, word vectors capture linguistic properties of words and relations. With the compositionality of space, arithmethic operations can be applied to vectors. This example can explain better by the usage of word vectors; vec(“king”) - vec(“man”) + vec(“woman”) is close to vec(“queen”) and vec(“Ankara”) - vec(“Turkey”) + vec(“France”) is close to vec(“Paris”).

*Word2vec* algorithm can learn the word embeddings in two different way; *Continuous Bag of Words (CBOW)* and *Skip-gram*. *CBOW* method predicts a target word by using context when *Skip-gram* predicts the context of each word. *Word2vec* algorithm maps every word *w* in dictionary *V* to vector *w(t)* which is a column matrix of *W*. *Word2vec* is an unsupervised algorithm and initial *W* matrix is constructed randomly. The *CBOW* model predicts a word *w(t)* using the context of it, $w(t-n), \ldots, w(t-1), w(t+1)\ldots, w(t+n)$. On the other hand, *Skip-gram* algorithm predicts each word in the context

using the word *w(t)*. In the *CBOW*, the hidden layer is the summation or average of the input word vectors. Given a sentence $w_1$, $w_2$, ..., $w_T$ , the objective function is to maximize the log likelihood of the language model:

$$\frac{1}{T}\sum_{t=k}^{T-k} \log P(w_t|w_{t-k}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+k}) \tag{3.1}$$

The model receives a window of *k* words around the target word $w_t$ at each time step *t*, rather than feeding *k* previous words into the model;

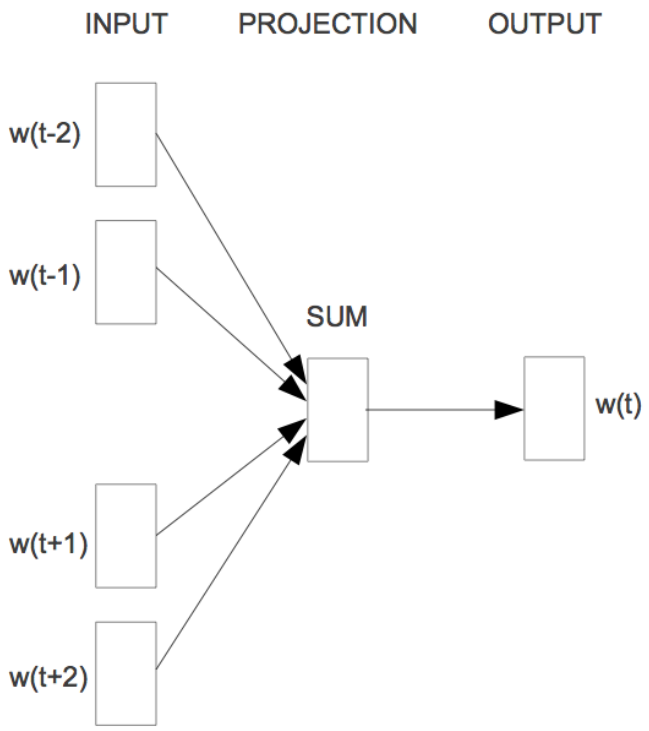



*Figure 3.2.* CBOW model

There is a need to calculate the posterior probability distribution of the target word given a specific context. The softmax function which is shown in Equation (3.2) helps to calculate this distribution for the given specific context:

$$P(w_t|w_{t-k},\dots,w_{t-1},w_{t+1},\dots,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{3.2}$$

where $y_i$ is the output of the *i-th* unit in the output layer

From Equation (3.2), it is also concluded that *CBOW* is a precognitive model which uses the context of word to predict center word. However, *skip-gram* aims to predict the context of word by using the center word. The input layer is a so-called "1-hot vector" for the Skip-gram model, which copies the embedding to the hidden layer for a word in the word vector matrix. The objective function predicts the word appearing in the context of the directed word:

$$\frac{1}{T}\sum_{t=k}^{j=k}\sum_{j=-k,j\neq 0}^{j=k} \log P(w_{t+j}|w_t) \tag{3.3}$$

Where *T* is the length of the whole sentence. For both variants, back propagation of the *neural network* and *Stochastic Gradient Descent (SGD)* on the objective function update the parameter *Θ = (W, U)* iteratively*:*

$$\Theta \leftarrow \Theta - \eta\frac{\partial \log P}{\partial \Theta} \tag{3.4}$$

The Word2Vec framework's goal is to predict word or word context based on used approach in it, i.e. CBOW or skip-gram. By maximizing the objective function, word embeddings are learned. Those word embeddings can be used not only for single

phrases but also for brief sentence segments [5]. They give efficient results for the document representation.

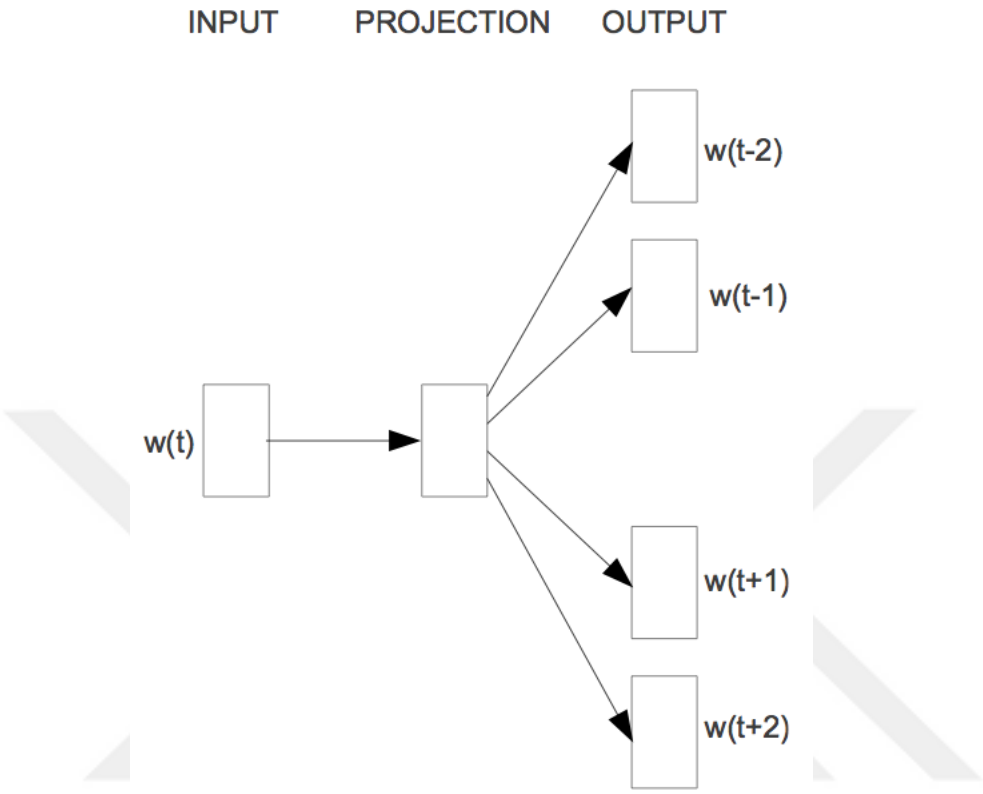



*Figure 3.3.* Skip-gram model

In this study, *Skip-gram* model will be used. Evaluation of *skip-gram* is computationally hard with softmax. Therefore, Mikolov proposed *Hierarchical Softmax* and *Negative sampling* algorithms to learn word embeddings for *skip-gram* algorithm [17]. By using softmax function, skip-gram can be written like below;

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W} \exp({v'_w}^T v_{w_I})} \tag{3.5}$$

where $v_w$ and $v_w{}'$ are the "input" and "output" vector representations of *w*, and *W* is the number of words in the vocabulary. In Equation (3.5), computing time $p(w_O|w_I)$ depends on the size of vocabulary and it is not efficient to use softmax algorithm because of the cost of computing.

One of the efficient algorithms of softmax is hierarchical softmax. The hierarchical softmax utilizes from binary tree representation. In the output layer, each word is the child node of the tree, *W* words and binary tree representation shows the relative probabilities of each word. Those relative probabilities of each word are used to define *random walk.*

More exactly, every term w can be reached by a suitable word order from the tree's root. In Equation (3.6), posterior probability of word co-occurrence where $n(w, j)$ is the *j-th* node on the path from the root to *w*, and let $L(w)$ is the length of this path. Based on this notation root is defined as $n(w, 1)$ . Additionally, $ch(n)$ is an arbitrary fixed child of inner node, *n* and $[\![x]\!]$ is defined as 1 if *x* is true and -1 otherwise. Hence, $p(w_O|w_I)$ is defined by *Hierarchical softmax* as follows:

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w,j+1) = ch(n(w,j))]\!] \cdot {v'_{n(w,j)}}^T v_{w_I}\right) \qquad (3.6)$$

where $\sigma(x) = 1/(1 + exp(-x))$. It can be verified that $\sum_{w=1}^{W} p(w|w_I) = 1$. This implies that the cost of computing $log\ p(w_O|w_I)$ and $\nabla\ log\ p(w_O|w_I)$ is proportional to $L(w_O)$, which is not higher than $log\ W$ on average. The hierarchical softmax formulation proposes only one representation $v_w$ for each word *w* and one representation $v'_n$ for every inner node *n* of the binary tree. This will reduce the computation cost of standard softmax function. This is because, softmax function assigns two different representations which are $v_w$ and $v'_w$ to each word.

It is proposed that *Negative Sampling* can be used in skip-gram algorithm and objective function of model is given in Equation 3.7.

$$log\ \sigma\left({v'_{w_O}}^T v_{w_I}\right) + \sum_{i=1}^{k} E_{w_{i\sim P_n(w)}}\left[log\ \sigma\left(-{v'_{w_i}}^T v_{w_I}\right)\right] \qquad (3.7)$$

which is used to replace every $log\ P(w_O|w_I)$ term in the Skip-gram objective. Hence, the objective is to use logistic regression to differentiate the target word $w_O$ from the noise distribution $P_n$(w), where there are *k* negative samples for each data sample. In other words, *k* is the number of noise words not all words in the vocabulary *(V).* Negative sampling assigns high probabilities to the real words, and low probabilities to noise words in the objective function. Using noise words give opportunity to train model faster.

### 3.2.2. Doc2vec

*Doc2vec* algorithm is derived by *Word2vec* algorithm. Here the basic idea comes from the structure of Word2vec which word vectors are used to predict the next word in context. Although the word vectors start with the random values, they learn the semantic structure in context. Learning semantic features is not prior goal but the additional benefit of this predictive task. This idea is also useful for paragraph vectors which can be utilized for prediction of next word when many contexts sampled from paragraph are given.

*Doc2vec* converts each paragraph to one vector. This final vector is the combination of vector learned by the "*Standard Paragraph Vector with Distributed Memory (PV-DM)"* or *"Vector Learned by the Paragraph Vector with Distributed Bag of Words (PVDBOW)"* [12].

#### 3.2.2.1. Paragraph Vector: A distributed memory model

In Paragraph Vector framework, there are unique vectors for each paragraph and word. Each column of matrix D represents a paragraph and each column of matrix W represents a word (see Figure 3.4). To predict the next word in a window, the paragraph and word vectors can be concatenated or averaged.

The idea behind to use paragraph vector with word vector is to utilize from the information which is gained from other contexts. Word vectors keep only the context information and paragraph vectors add the information coming from other contexts. Since this approach keeps the information from other contexts on memory, it is called *"Distributed Memory Model of Paragraph Vectors (PV-DM)"*.

The paragraph token can be thought of as another word. It acts as a memory that remembers what is missing from the current context – or the topic of the paragraph. This is the reason why it is called "*Distributed Memory Model of Paragraph Vectors (PV-DM)"*.

The contexts of algorithm have fixed-length and are obtained by sliding window over the paragraph. All contexts generated from the same paragraph are used to construct the paragraph vector. In addition to this, the word matrix is shared across all paragraphs. This method can be thought as analogous of the *Skip-gram* of *Word2vec*.

The paragraph vectors and word vectors learn by using *stochastic gradient descent* via backpropagation. For each iteration, a fixed-length context is sampled from a random paragraph, the error gradient is calculated, and the parameters of model are updated. *PV-DM* is represented in Figure 3.4.

After being trained, the paragraph vectors can be used as features. These features can be fed directly to conventional machine learning techniques such as logistic regression, support vector machines or K-means. In this study, we feed these features to logistic regression.

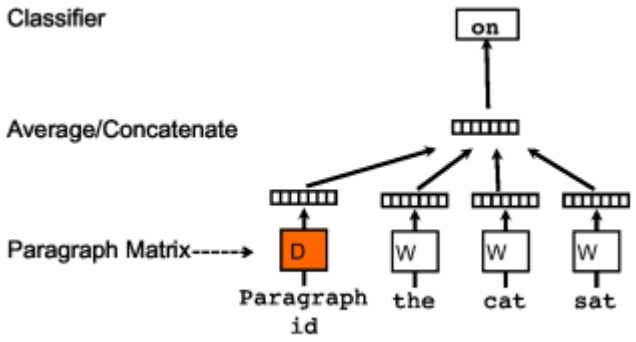



*Figure 3.4.* Distributed Memory Model of Paragraph Vectors (PV-DM) Representation

### 3.2.2.2. Paragraph Vector without word ordering: Distributed bag of words

*"Distributed Memory Model of Paragraph Vectors (PV-DM)"* develops paragraph vector to predict the next word in the context window. *Distributed Bag of Words version of Paragraph Vector (PV-DBOW)* develops paragraph vector to predict words randomly sampled from the paragraph other than next word in the context window. This method is similar to *CBOW* in *Word2vec.* In *PV-DBOW*, a text window is sampled, and a random word is sampled from this text window at each iteration. Stochastic gradient descent is used in *PV-DBOW.* As a result, this technique which is represented in Figure 3.5 forms the classifier vector of paragraph.

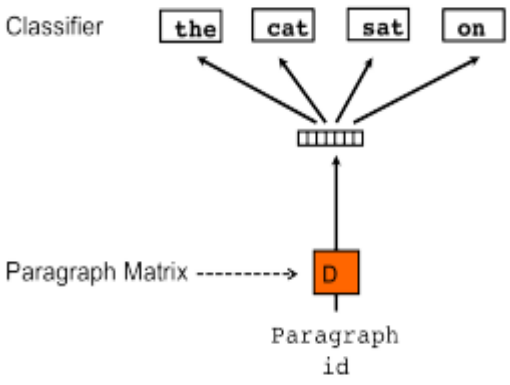



*Figure 3.5*. Distributed Bag of Words version of Paragraph Vector (PV-DBOW) Representation

### 3.2.3. Node2vec

*Node2vec* is inspired from *Word2vec* algorithm, which is proposed by Leskovec [13]. It is a semi-supervised algorithm for scalable feature learning in networks. The algorithm returns feature representations of nodes similar to *word2vec*. They maximize the likelihood of preserving network neighborhoods of nodes in a *d*-dimensional feature space.

Feature learning framework is similar to *word2vec*. Let $G = (V, E)$ be a given network. It can be used both directed and undirected graphs. Additionally, it works with weighted and unweighted network. Mapping function is defined as $f: V \rightarrow R^d$ where $V$ is the set of nodes and $R^d$ is the feature representation with dimension $d$ and f is a matrix with size $|V| \times d$ . The neighborhood of source node $u \in V$ as a $N_s(u) \subset V$ is generated with sampling strategy $S$.

In this algorithm, *Skip-gram* architecture is applied to networks. Objective function is to maximize the log-probability of observing a network neighborhood $N_s(u)$ for a node $u$ given that its feature representation, given by $f$:

$$\max_f \sum_{u \in V} \log Pr(Ns(u)|f(u))$$

The model has two assumptions; *Conditional Independence* and Symmetry in feature space. The assumptions of algorithm are given below;

- Conditional independence assumes that presence probability of a neighborhood node is independent of presence of any other neighborhood node given the feature representation of the source. With this idea, likelihood of objective can be written below;

$$Pr(Ns(u)|f(u)) = \prod_{n_i \in N_s(u)} \Pr(n_i|f(u))$$

- Symmetry in feature space assumes that the effect of a source node and neighborhood node on each other is symmetrical. Therefore, the conditional presence probability of every source-neighborhood node pair is modelled as:

$$\Pr(n_i|f(u)) = \frac{exp\big(f(n_i) \cdot f(u)\big)}{\sum_{v \in V} exp\big(f(v) \cdot f(u)\big)}$$

With these assumptions, objective function is simplified to the equation below;

$$\max_f \sum_{u \in V} \left[ -\log Z_u + \sum_{n_i \in N_s(u)} exp\big(f(n_i) \cdot f(u)\big) \right]$$

where $Z_u$ is the per-node partition matrix;

$$Z_u = \sum_{v \in V} exp\big(f(u) \cdot f(v)\big)$$

*Node2vec* algorithm also uses the random walk when creating paths -similar to sentences in *Word2vec*-, but the random walk of *Node2vec* is biased. *Node2vec* creates biased random walks by combining *Breath-first Sampling (BFS)* and *Depth-first Sampling (DFS)* strategy. *BFS* takes the immediate neighbors of source node and *DFS* makes sequentially sampling at increasing distance from the source node. *Node2vec* algorithm introduced to 2nd order random walk with return *(p)* and in-out *(q)* parameters which seek bias *(α)* to combine the *BFS* and *DFS*.

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} \; if \; d_{tx} = 0 \\ 1 \; if \; d_{tx} = 1 \\ \frac{1}{q} \; if \; d_{tx} = 2 \end{cases}$$

In Figure 3.6, random walk representation is given. The current state of random walk is node v and the previous state was *t*. For the current state, random walk should decide where it will go based on the transition probabilities of neighbor nodes. Transition probability is defined as $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where and $d_{tx}$ denotes the shortest path distance between nodes *t* and *x*. Possible $d_{tx}$ values are in {*0, 1, 2*}, and *p* and *q* are essential parameters to walk. Those parameters intuitively regulate how quickly the walk explores and leaves the starting node u neighborhood. The parameters in specific enable our search method to interpolate between BFS and DFS and thus reflect an affinity for distinct concepts of node equivalences.

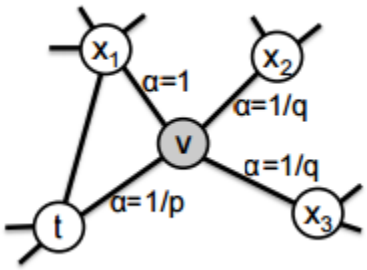



*Figure 3.6*. Biased random walk procedure with parameters *p* & *q*

*Node2vec* algorithm covers the *homophily* and *structural equivalence* with interpolation of BFS and DFS. Nodes that are highly interconnected should be embedded closely together based on the homophily hypothesis. By comparison, nodes that have similar structural roles in networks should be embedded closely together based on the structural equivalence hypothesis. These equivalence concepts are not exclusive in real-world; networks frequently display both behaviors where some nodes display homophily while others represent structural equivalence.

The main contributions of *Node2vec* are given below;

- *Node2vec* is proposed as an efficient scalable algorithm for network-based feature learning that efficiently optimizes a new network-aware, neighborhood objective that uses SGD.
- It shows how *Node2vec* complies with proven network science principles and provides flexibility in discovering embeddings that conform to distinct equivalences.
- *Node2vec* is a feature learning method that works on neighborhood preserving objectives, from nodes to pairs of nodes for edge-based prediction tasks.

As a result, the pseudocode of the node2vec algorithm is given below;

**Algorithm 2:** Node2Vec Algorithm

**LearnFeatures** (Graph $G = (V, E, W)$, Dimensions $d$, Walks per node $r$, Walk length $l$, Context size $k$, Return $p$, In-out $q$)
    $\pi$ = PreprocessModifiedWeights$(G, p, q)$
    $G' = (V, E, \pi)$
    Initialize *walks* to Empty
    **for** *iter* = 1 to r **do**
        **for all** nodes u ∈ $V$ **do**
            walk = node2vecWalk$(G', u, l)$
            Append walk to walks
        $f$ = StochasticGradientDescent$(k, d, walks)$
    **return** $f$

**node2vecWalk** (Graph $G' = (V, E, \pi)$, Start node $u$, Length $l$)
    Inititalize walk to [$u$]
    **for** *walk_iter* = 1 to $l$ **do**
        $curr$ = $walk$[−1]
        $V_{curr}$ = GetNeighbors($curr$, G')
        s = AliasSample($V_{curr}$, π)
        Append $s$ to *walk*
    **return** *walk*

### 3.3. Binary Classifiers

In this section, classifier model algorithms used in this study will be explained. All of those algorithms are used with binary classification purpose. In the binary classification, model tries to predict dependent variables which only take values as 0 or 1. This dependent variable is a target variable and implies event for 1 and non-event for 0. All classifier algorithms models relationship between independent variables and dependent variables. Logistic regression, decision tree, random forest, gradient boosting and neural networks are used in this study and will be explained in this section.

#### 3.3.1. Logistic Regression

Similar to all binary classification algorithms, logistic regression also aims to find the relation between independent variables and dependent variable. Logistic regression is one of the most interpretable algorithms amongst classifiers and easy to explain relations which affect dependent variable. Runtime and calculation effort are very low when compared to other classifiers. Logistic regression proposes probability of event as an output. *p* is defined as probability of event and *1-p* is probability of non- event. Based on this definitions, odds ratio is defined as:

$$odds = \frac{p}{1-p}$$

In Logistic Regression, *p*, probability of event is defined with independent variable *x* as $\alpha+\beta x$. This rough estimation does not give a value which is between 0 and 1. To get results between 0 and 1, logistic regression uses natural logarithm by addressing natural logarithm of odds to relation of independent variables, $\alpha+\beta x$. In equation (3.8), model equation is presented:

$$logit(y) = ln(odds) = ln(\frac{p}{1-p}) = \alpha + \beta x \qquad (3.8)$$

By extracting p from equation (3.8), the following equation is derived:

$$p = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} = \frac{1}{1+e^{-(\alpha+\beta x)}} \quad (3.9)$$

Equation (3.9) gives the formula of event occurrence probability for only one variable and formula is defined for many independent variables in the Equation (3.10).

$$p = \frac{e^{\alpha+\beta_1 x_1+\cdots+\beta_k x_k}}{1+e^{\alpha+\beta_1 x_1+\cdots+\beta_k x_k}} = \frac{1}{1+e^{-(e^{\alpha+\beta_1 x_1+\cdots+\beta_k x_k})}} \quad (3.10)$$

Logistic regression model is fitted by using the Equation (3.10) based on maximum likelihood estimation where $y_i$ is the value of random variable between 0 and 1, and $p_i$ is the probability of $i^{th}$ observation. Maximum likelihood estimation finds the values of α and β which maximizes the function L.

$$L = \prod_{i=1}^{n} P_i^{1-y_i}(1 - P_i)^{y_i} \quad (3.11)$$

Although logistic regression model looks like simple linear regression model, the underlying distribution is binomial, and α and β parameters cannot be estimated in the same way as for simple linear regression [37].

### 3.3.2. Decision Tree Algorithm

Decision tree algorithms [38,39] are some of the most popular machine learning algorithms to build models have higher interpretability power. The reason why they are so popular is their capability and advantage to select from all attributes used to describe the data, a subset of attributes that are relevant for classification, identify complex predictive relations among attribute and produce classifiers that are easy to comprehend for humans [40].

A decision tree breaks data to subsets which are called nodes. The final subsets are called leaf nodes which is not split and there are internal nodes before final subsets

and they can split. Internal nodes are splitted with respect to rules on model feature values. In decision tree, there are branches which connect internal nodes to each other and internal nodes to leaf nodes. Leaf nodes are labeled based on the majority of dependent values. For the binary classification, leaf nodes are final subsets of model data which labeled 0 or 1.

In literature, many different decision tree algorithms which include ID3, C4.5, J48, NB Tree and CART. In this study, CART algorithm is used. Relevant notation is given below:

| | |
|---|---|
| $Y$ | The dependent variable, or target variable. It can be ordinal categorical, nominal categorical or continuous. If Y is categorical with J classes, its class takes values in C = {1, …, *J*}. |
| $X_m, m = 1, ...,$M | The set of all predictor variables. A predictor can be ordinal categorical, nominal categorical or continuous. |
| $h = \{x_n, y_n\}_1^N$ | The whole learning sample. |
| $h(t)$ | The learning samples that fall in node *t*. |
| $w_n$ | The case weight associated with case *n*. |
| $f_n$ | The frequency weight associated with case *n*. Non-integral positive value is rounded to its nearest integer. |
| $\pi(j), j = 1, \dots, J$ | Prior probability of *Y = j, j = 1, …, J*. |
| $p(j,t), j = 1, \dots, J$ | The probability of a case in class *j* and node *t*. |
| $p(t)$ | The probability of a case in node *t*. |
| $p(j\|t), j = 1, \dots, J$ | The probability of a case in class j given that it falls into node *t*. |
| $C(j\|t)$ | The cost of miss-classifying a class j case as a class i case. Clearly $C(j\|j) = 0$ |

Decision tree algorithm tries to create splits which has the “purest” child nodes. At each iteration, purity of nodes increases by splitting to child nodes. At each split, nodes are split based on only one variable. A tree starts to grow from the root node and repeats the following steps on each node:

- Among available model features, there is need to define the best split and feature at the root node.

- Split the node to child nodes with respect to criterion which gives the maximum purity on each child node. There are different criteria to calculate impurity of node, for example Gini, Twoing, and ordered Twoing criteria. In this study, Gini is used as splitting criteria.
- Repeat the process if stopping criteria or maximum purity does not apply.

At node *t*, the best split *s* is chosen to maximize a splitting criterion $\Delta i(s,t)$. When the impurity measure for a node can be defined, the splitting criterion corresponds to a decrease in impurity

If *Y* is categorical, there are three splitting criteria available: Gini, Twoing, and ordered Twoing criteria. At node *t*, $\pi(j)$ is the prior probability and let probabilities $p(j,t)$ , $p(t)$ *and* $p(j \mid t)$ be estimated by:

$$p(j,t) = \frac{\pi(j) N_{w,j}(t)}{N_{w,j}},$$

$$p(t) = \sum_j p(j,t),$$

$$p(j|t) = \frac{p(j,t)}{p(t)} = \frac{p(j,t)}{\sum_j p(j,t)}$$

where

$$N_{w,j} = \sum_{n \epsilon h} w_n f_n \, I(y_n = j)$$

$$N_{w,j}(t) = \sum_{n \epsilon h(t)} w_n f_n \, I(y_n = j) \qquad (3.12)$$

with *I(a = b)* being indicator function taking value 1 when $a = b$ , 0 otherwise.

The Gini impurity measure at a node t is defined as:

$$i(t) = \sum_{i,j} C(i|j)p(i|t)p(j|t) \tag{3.13}$$

The Gini splitting criterion is the decrease of impurity defined as:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{3.14}$$

where $p_L$ and $p_R$ are probabilities of sending a case to the left child node $t_L$ and to the right child node $t_L$ respectively. They are estimated as $p_L = p(t_L)/p(t)$ and $p_R = p(t_R)/p(t)$[38].

### 3.3.3. Random Forest

Random forest [41] is a machine learning algorithm which consists of the individually created decision trees. This is the reason why it is called "Random Forest". It is an ensemble-based learning algorithm and the idea behind ensemble approach is to decrease variance. In Random Forest, bagging which stands for Bootstrap Aggregation is used for ensembling model. Bootstrap is a method from statistics traditionally used to measure uncertainty of some estimator. Bagging also allows for handling of missing features.

A downside to bagged trees is that the interpretability inherent in the single decision tree is lost. One method by which to re-gain some amount of insight is through a technique called variable importance measure. For each feature, find each split that uses it in the ensemble and average the decrease in loss across all such splits. With random forests, it is allowed a subset of features to be used at each split. This provides a decrease in correlation $\rho$ which leads to a decrease in variance. Again, there is also an increase in bias due to the restriction of the feature space, but as with vanilla bagged decision trees this proves to not often be an issue. Finally, even powerful predictors will no longer be present in every tree (assuming sufficient number of trees and sufficient restriction of features at each split), allowing for more graceful handling of missing predictors.

There is a true population N that we wish to compute an estimator for, as well a training set *S* sampled from *N (S ∼ N).* While it can be found an approximation by computing the estimator on *S,* we cannot know what the error is with respect to the true value. To do so there is a need to multiple independent training sets $S_1, S_2, \ldots$ all sampled from *N*. However, the assumption that S = N is set, it can be generated a new bootstrap set *Z* sampled with replacement from *S (Z ∼ S, |Z| = |S|).* In fact, we can generate many such samples $Z_1, Z_2, \ldots, Z_M$. We can then look at the variability of our estimate across these bootstrap sets to obtain a measure of error.

Now, returning to ensembling, machine learning model $G_m$ is trained on each $Z_m$ and, and a new aggregate predictor is defined as:

$$G(X) = \sum_m \frac{G_m(x)}{M} \tag{3.15}$$

Bagging creates less correlated predictors than if they were all simply trained on *S,* thereby decreasing $\rho$. While the bias of each individual predictor increases due to each bootstrap set not having the full training set available, in practice it has been found that the decrease in variance outweighs the increase in bias. Also note that increasing the number of predictors *M* can't lead to additional overfitting, as $\rho$ is insensitive to *M* and therefore overall variance can only decrease. An additional advantage of bagging is called out-of-bag estimation. It can be shown that each bootstrapped sample only contains approximately $\frac{2}{3}$ of S, and thus the other $\frac{1}{3}$ can be used as an estimate of error, called out-of-bag error. In the limit, as M $\rightarrow\infty$, out-of-bag error gives an equivalent result to leave-one-out cross-validation [42].

### 3.3.4. Gradient Boosting

Bagging is a variance-reducing technique, whereas boosting is used for bias reduction. Boosting models uses decision trees which produce high bias, low variance models.

These trees are also known as weak learners. Continuing our exploration via the use of decision trees, we can make them into weak learners by allowing each tree to only make one decision before making a prediction; these are known as decision stumps.

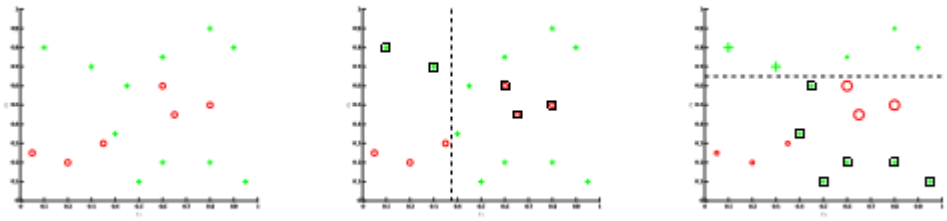


*Figure 3.7* Boosting procedure

Figure 3.7 shows the intuition behind the boosting. First graph represents the initial dataset and it is splitted with the single decision stump on the middle plot. The key idea is to track which examples the classifier got wrong, and increase their relative weight compared to the correctly classified examples. In the third plot, new single decision stump is introduced and, it is more incentivized to correctly classify these "hard negatives". This process continue and examples are reweighted at each step. At the end of the process, the combination of those weak learners create ensemble classifier.

**Algorithm 1:** Forward Stagewise Additive Modeling

**Input:** Labeled training data $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$
**Output:** Ensemble classifer $f(x)$
Initialize $f_0$(x) = 0
**for** m = 0 **to** $M$ **do**
  Compute $(B_m, \gamma_m) = argmin_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta G(x_i; \gamma))$
  Set $f_m(x) = f_{m-1}(x) + \beta_m G(x; y_i)$
**End**
$f(x) = f_m(x)$

Close inspection reveals that few assumptions are made about the learning problem at hand, the only major ones being the additive nature of the ensembling as well as the fixing of all previous weightings and parameters after a given step. We again have

weak classifiers $G(x)$, though this time we explicitly parameterize them by their parameters $\gamma$. At each step we are trying to find the next weak learner's parameters and weighting so to best match the remaining error of the current ensemble. As a concrete implementation of this algorithm, using a squared loss would be the same as fitting individual classifiers to the residual $y_i - f_{m-1}(x_i)$.

In general, it is not always easy to write out a closed-form solution to the minimization problem presented in Forward Stagewise Additive Modeling. One of the most obvious things to do in this case would be to take the derivative of the loss and perform gradient descent. In gradient boosting, we instead compute the gradient at each training point with respect to the current predictor (typically a decision stump):

$$g_i = \frac{\partial L(y, f(x_i))}{\partial f(x_i)} \tag{3.16}$$

We then train a new regression predictor to match this gradient and use it as the gradient step. In Forward Stagewise Additive Modeling, this works out to [42]:

$$\gamma_i = argmin_\gamma \sum_{i-1} (g_i - G(x_i; \gamma))^2 \tag{3.17}$$

### 3.3.5. Neural Networks

A multilayer perceptron (MLP) is a type of feedforward artificial neural network which includes at least three layers of nodes: an input layer, a hidden layer and an output layer. Each node other than the input nodes is a neuron that uses a nonlinear activation function. At the training stage of Multilayer perceptron algorithm, backpropagation is used for learning and updating parameters [43]. Activation function of MLP is non-linear and this gives an opportunity to recognize data which is not separable with linear activation function.

In the first layer of MLP, there are M linear combinations of the d-dimension inputs:

$$b_j = \sum_{i=0}^{d} w_{ji}^{(1)} x_i \qquad j = 1,2, \dots, M. \tag{3.18}$$

In Equation (3.18), $b_j$ is activation, $w_{ji}^{(1)}$ is the weight and $x_i$ is the bias. Nonlinear activation function, specifically a sigmoid in here transforms all of $b_j$'s.

$$z_j = h(b_j) = \frac{1}{1 + e^{-b_j}} \tag{3.19}$$

In Equation (3.19), $z_j$ represents the output of hidden layers and takes the activation value of first layer, $b_j$ as an input. Sigmoid function transforms first-layer to hidden-layer. Hidden-layer is transformed to second-layer. It needs to note that transformation from hidden-layer to second-layer is liner in order to get K dimensional output when the transformation from first-layer to hidden-layer is non-linear:

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \qquad k = 1,2, \dots, K. \tag{3.20}$$

$w_{kj}^{(2)}$ represents the weights for second layer and $z_j$ is the bias. Output of MLP is derived by running sigmoid function on the second-layer:

$$y_k = g(a_k) = \frac{1}{1 + e^{-a_k}} \tag{3.21}$$

As a result, the closed form formula can be written for MLP output as [44]:

$$y_k = \sum_{j=0}^{M} w_{kj}^{(2)} h_j \sum_{j=0}^{d} w_{ji}^{(1)} x_i \qquad k = 1,2,\dots,K. \tag{3.22}$$

This is illustrated in Figure 3.8.

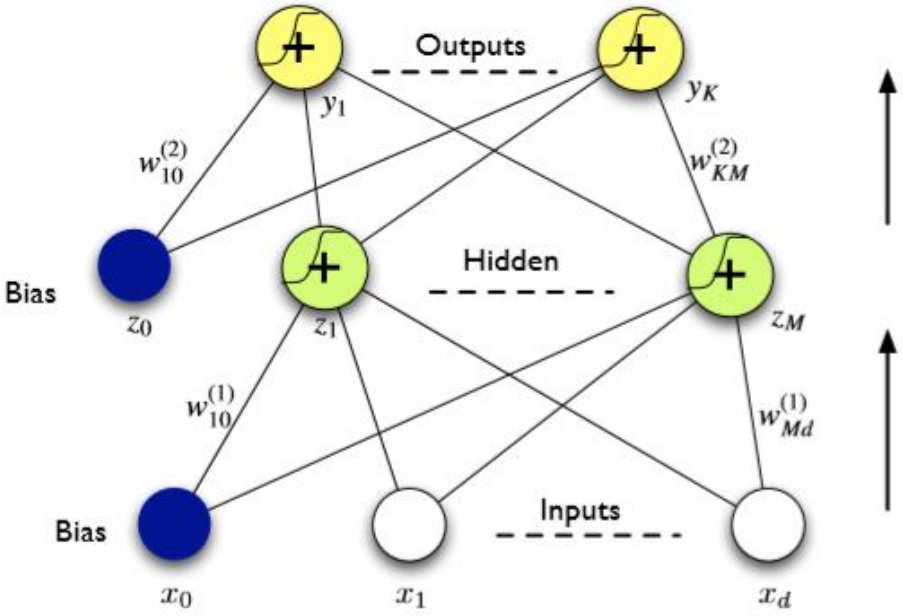



*Figure 3.8.* Multi-layer perceptron (MLP) representation

# CHAPTER 4

# PROPOSED ALGORITHM AND EXPERIMENTAL RESULTS

In this chapter, we introduce three different approaches considered for spam review detection. In the text-only approach, DocRep, representations are learned for the reviews by utilizing a document representation learning algorithm (Doc2vec) and used as features for creating spam review detection model. In the network-only approach, NodeRep, representations are learned for the network nodes via a node representation learning algorithm (Node2vec) and used as features for creating a classifier. Lastly, a recently our proposed framework SPR2EP [36]. In this framework, reviewer, restaurant/hotel and review feature representations are learned and used in detecting spam reviews. First of all, reviewer-product network is generated by creating a link between reviewers and products if a reviewer has written a review on a product.

The product mentioned here refers to the restaurants and hotels in the datasets used in this study. Then, by running Node2vec algorithm on this network, vector representations for reviewers and products are learned. Similarly, by running Doc2vec algorithm where inputs are the reviews, vector representations for the reviews are learned again in an unsupervised manner. Finally, a combination of these representations is fed into various classification algorithms to build classifiers for spam review detection.

In the following subsections, detailed explanation of the approaches considered in this study will be presented. Firstly, datasets and their characteristics are described. Then, performances of the models generated separately via document and node embeddings, DocRep and NodeRep respectively, are compared. Finally, the models created using the concatenated document and node embeddings will be explained and their performances will be compared to those of DocRep and NodeRep and the previous

studies. In order to perform a fair comparison, we considered various embedding sizes, and classification algorithms which are employed with their parameters left at their default values.

The performance of machine learning methods depends on preprocessing and transformation of input data. Feature engineering step is the prior and one of the most important steps of modelling. Mostly, feature engineering methods include manual and human intervention steps to improve classifier model performance. In spite of this, representation learning extracts useful common information automatically for feature detection or classification. In the next sections, those algorithms will be explained.

### 4.1. Data Description

To assess the approaches considered in this study, three different datasets which include the reviews gathered from yelp.com are used. The first one named YELPCHI includes the reviews for restaurants and hotels in Chicago. This dataset includes separate datasets for Restaurants and Hotels and the names of users and products are anonymized. The other two datasets, named YELPNYC and YELPZIP include reviews only for restaurants. YELPNYC includes the reviews in New York area, whereas YELPZIP dataset is larger and contains reviews for the restaurant in states of New Jersey (NJ), Vermont (VT), Connecticut (CT), and Pennsylvania (PA). Table 4.1 gives the summary statistics of these data sets. The YELPCHI dataset is collected by Mukherjee et al. [39] and used by Mukherjee et al. [39] and Rayana and Akoglu [43]. Other two datasets are collected and used by Rayana and Akoglu [43] and shared also with the authors of this study.

YELP [2] uses an undisclosed spam review classification algorithm to label reviews as genuine and fake. It separates the fake reviews as filtered and shows unfiltered reviews as recommended by default. These three datasets include both genuine and fake reviews. It is worthy of notice that, although the labeling and filtering algorithm of YELP could not be perfect, the labels included in these datasets are assumed to be

providing us the ground-truth labels as it is also stated in the previous works [39,43]. Since the detection of the spam reviews just from the raw review text is difficult for humans, the labeling algorithm of YELP using various features that human may not obtain is assumed to provide more reliable ground-truth labels than manual-only approaches used in some of the previous works.

Table 4.1. *Review datasets used in this study*

| Dataset | # Reviews (filtered %) | # Users (spammer % ) | # Products (rest. & hotel) |
|---|---|---|---|
| YELPCHI | 67,395 (13.23%) | 38,063 (20.33%) | 201 |
| YELPNYC | 359,052 (10.27%) | 160,225 (17.79%) | 923 |
| YELPZIP | 608,598 (13.22%) | 260,277 (23.91%) | 5,044 |

### 4.2. Learning Review Embeddings

In this study, three different approaches are presented and compared based on their spam review detection performances. For a fair comparison between the performances of these approaches, three different dimensions of 96, 192 and 384 are selected as feature vectors size and these features are fed into classification algorithms to achieve the best performance. So, in order to generate a classification model in NodeRep where the input dimension is 384, the size parameter for the node embeddings in Node2vec algorithm is set to 192 so as to get the combined reviewer-product embedding size of 384. Similarly, the corresponding DocRep model with 384 input features is generated by running Doc2vec on review text with embedding size of 384. On the other hand, in order to get input feature vector of size 384 for the SPR2EP approach, node embeddings of size 128 is obtained by running Node2vec and then

these reviewer-product representations, where each node contribute 128 features, are combined with additional 128 features coming from the review representation which is obtained by running Doc2vec. Respective representations for input features of sizes 96 and 192 are obtained similarly and used as input in generating review detection models.

In the first approach, DocRep, Doc2vec [12] algorithm is used to create review embeddings from raw review texts. Doc2vec algorithm learns feature vectors for every review in each one the three datasets independently. PV-DM is the method of choice of Doc2vec algorithm in our study. Window size, which is the maximum distance between predicted word and context word within a document, and the number of epochs is both set to 10. Another parameter of choice is the minimum count which is used to ignore all the words with a total frequency lower than this value. The minimum count value is set to 1 in order to achieve the best representations for the reviews.

Every review is converted to the vector embeddings of various sizes since both the performances of DocRep and SPR2EP review detection models utilize features generated by Doc2vec method.

In the second approach, NodeRep, Node2vec [13] algorithm is utilized to extract the feature embeddings for the nodes of reviewer-product network. The weighted version of Node2vec is used in this study. The reviewer-product network, constructed by taking into account the reviews regarding a reviewer-product pair together with its rating, is fed into the Node2vec algorithm as an undirected weighted network. Here, review ratings are used as edge weights to take advantage of the rating information. The random walk length, window size and the number of walks started on each node are all set to 10. Random walks are repeated with respect to number of walks parameter.

In order to get the most out of the network information, we performed a grid search over the two important parameters of Node2vec algorithm, breadth first sampling parameter p and deep first sampling parameter q. We have decided to run a grid search

over p, q ϵ {0.25, 0.5, 1, 2, 4} as it is done in the original work [13] for each dataset. In order to find the best pair of parameters, we generate logistic regression models with feature vector size of 128 and perform 10-fold cross validation. As a result, we achieved best performances for each dataset with the p and q values in Table *4.2* and used them to generate node features for further analysis.

Table 4.2. *Best Node2vec settings*

| | YELPCHI | YELPNYC | YELPZIP |
|---|---|---|---|
| Node2vec settings (p,q) | (4, 1) | (0.5, 2) | (1,0.25) |

Last approach, SPR2EP framework is based on learning the feature vectors of the reviewers and hotels/restaurants which are considered as nodes on the reviewer-product network and concatenating them with the learned feature vectors for the reviews. So, SPR2EP consists not only of generating and using the representations for the review text but also representations obtained for the weighted reviewer-product network nodes. Therefore, two processes operate in parallel for exploiting the review text and the underlying reviewer-product network structure separately.

The main objective of this study is to analyse and compare various spam review detection approaches. In this respect, three different approaches are considered for learning representations of the reviews to be used in creating classification models. These approaches aimed at learning these representations both from the raw review text as well as the reviewer-product pairs since these pairs are considered as attributes of the reviews. Here, Doc2vec algorithm outputs representations for each review via considering the structure of raw review texts. On the other hand, Node2vec output features for reviewer and product nodes, by exploiting the underlying reviewer-product network structure, which are then combined for a particular review. As a

result, Doc2vec algorithm provides vector embeddings of size 96, 192 and 384 for every review. Node2vec algorithm also provides vector embeddings of size 96, 192, and 384 for every review to be used in NodeRep and SPR2EP where each one of the embeddings obtained for the corresponding reviewer and product nodes contribute the half.

In the next section, accuracy performances of the three models will be presented and compared with the performances obtained in the previous state-of-the-art studies. To make a fair comparison among all the approaches considered here and to observe the relative improvements achieved through textual and network features, and the effect of different embedding sizes on the performance, feature dimensions are set to 96, 192, and 384. So, both of the three spam review detection models are generated with review feature vectors of size 96, 192, and 384. DocRep model uses the feature embeddings created by considering only the review text, whereas NodeRep model uses the combination of two feature vectors of size 192 obtained for each reviewer and product pair regarding a review in case of feature dimension of 384. The last model, SPR2EP, utilizes the feature vectors for each review that are generated by concatenating the corresponding document vector for the review, and node feature vectors of the corresponding reviewer and product. For this combined model, we have run both Doc2vec and Node2vec algorithms with embedding vector sizes of 32, 64, and 128, so that the size of the concatenated embedding vectors for each review is 96, 192 and 384, respectively. For example, SPR2EP combines the output of DocRep with feature vector of size 128 and the output of NodeRep with feature vector of size 128 for feature vector of size 384. NodeRep algorithm is trained with feature vector of size 128 but it produces 256 feature size by augmenting reviewer and product features. As a result, SPR2REP input with feature vector of size 384 includes 128 features coming from DocRep and 256 features coming from NodeRep. In Table 4.3, model parameters are given for each setting. It should be noted that NodeRep model produces double features because of combining the reviewer and product features. The vector

embedding combinations of feature size of 96 and 384 are represented at Figures 4.1 and 4.2.

Table 4.3 *Embedding algorithm training parameters for each feature size*

| | | | SPR2EP | |
|---|---|---|---|---|
| Feature Size | NodeRep | DocRep | NodeRep | DocRep |
| 96 | 48 | 96 | 32 | 32 |
| 128 | 64 | 128 | 48 | 48 |
| 384 | 192 | 384 | 128 | 128 |

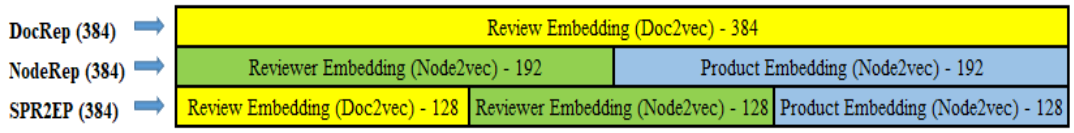



*Figure 4.1* Representation of embedding combinations at feature size of 384

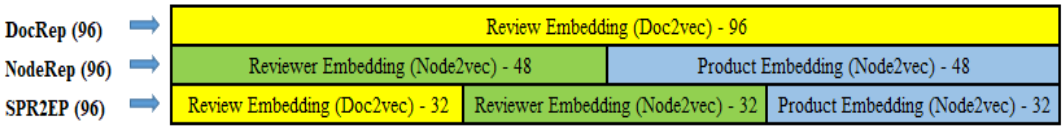



*Figure 4.2* Representation of embedding combinations at feature size of 96

All computational results in this study are obtained by using 10-fold-cross validation. Multilayer Perceptron, Logistic Regression, CART Decision Tree, Gradient Boosting, and Random Forest classifiers are adapted to discriminate between spam and legitimate reviews. All the classification algorithms are employed with their parameters left at their default values as explained in scikit-learn[2]. Implementation of the proposed framework is done in Python[3] programming language. Doc2vec features

[2] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html
[3] www.python.org

are obtained using gensim[4] library. Similarly, Node2vec implementation also uses Word2vec module of the gensim library after the creation of random walks in the network, based on a blend of BFS and DFS strategies. Additionally, networkx[5] library is utilized in Node2vec algorithm, which provides an opportunity to work with graph structures. For the data preprocessing and cleaning stages, pandas[6] and numpy[7] libraries are used.

### 4.3. Performance Measures

#### 4.3.1. ROC AUC

This measure is selected because our datasets are unbalanced. Also, we use this measure to compare the same size datasets. Basically, ROC AUC is Area Under Curve of Receiver Operating Curve. To calculate this measure, we used prediction scores and target class information of observations. ROC is a graphical plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. TPR is also known as recall and sensitivity. TPR is the ratio of correctly predicted positive observations (TP) to positive observations (P);

$$TPR = \frac{TP}{P}$$

FRP is also known as (1-specifity) or Type I Error. FPR is the ratio of incorrectly predicted negative observations as positive (FP) to negative observations (N);

$$FPR = \frac{FP}{N}$$

It measures the quality of ranking. When calculating TP and FP, it is also essential to define thresholds. It is possible to create many different thresholds between 1 and n

[4] https://radimrehurek.com/gensim/
[5] https://networkx.github.io/
[6] https://pandas.pydata.org/
[7] http://www.numpy.org/

which is the scored number of observations. In this study, we used score and target values. This approach sets different thresholds for each observation. It sorts scores from the highest to the lowest and sets lowering thresholds. If the observation has true value, the curve gets along in TPR direction. If the observation has false value, the curve gets along in FPR direction. ROC is scaled between 0 and 1 in x and y axes. ROC AUC calculates the area under the curve. The maximum value that ROC AUC can take is 1, which is a perfect classification. For random guessing, it takes 0.5. The contribution of the prediction algorithm is compared based on random guessing most of the time.

In Figure 4.4, score values and observation classes are given for 20 data points. There is a representation of the ROC curve [45].

| Inst# | Class | Score | Inst# | Class | Score |
|---|---|---|---|---|---|
| 1 | p | .9 | 11 | p | .4 |
| 2 | p | .8 | 12 | n | .39 |
| 3 | n | .7 | 13 | p | .38 |
| 4 | p | .6 | 14 | n | .37 |
| 5 | p | .55 | 15 | n | .36 |
| 6 | p | .54 | 16 | n | .35 |
| 7 | n | .53 | 17 | p | .34 |
| 8 | n | .52 | 18 | n | .33 |
| 9 | p | .51 | 19 | p | .30 |
| 10 | n | .505 | 20 | n | .1 |

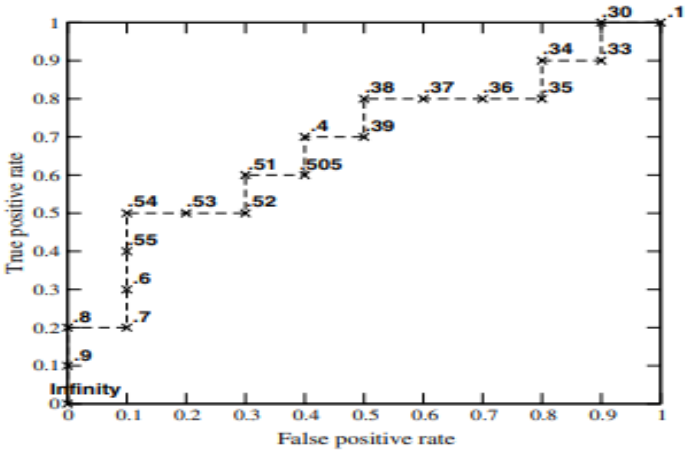



*Figure 4.3.* Receiver Operating Characteristic Curve Interpretation

### 4.3.2. Average Precision

This measure is selected because our datasets are unbalanced. Also, we use this measure to compare the same size datasets. It is calculated by utilizing precision-recall curve. Precision measures the accuracy of predictions. Precision is the ratio of correctly predicted positive predictions (TP) to all positive predictions which are the sum of correctly predicted positive predictions (TP) and incorrectly predicted negative observations (FP);

$$Precision = \frac{TP}{TP + FP}$$

Recall is True Positive Ratio (TPR). It measures the performance of detecting positive observations. It is ratio of the ratio of correctly predicted positive observations (TP) to positive observations (P);

$$Recall = \frac{TP}{TP + FN}$$

Precision and Recall are the single measures for each scored dataset. However, it is possible to calculate them for each ranked observation and Precision-Recall curve can be plotted from the highest-score observation to lowest-score observation, which is the curve of precision function of recall, between 0 and 1.

Average Precision, AP uses precision and recall values at the threshold k, for every observation and sums the Precision of each observation with the weights which is a difference between recall of current observation and the recall of previous observation [46];

$$AP = \sum_{k=1}^{n} (R_k - R_{k-1}) P_k$$

### 4.4. Evaluation

All classification models are evaluated and compared in terms of average precision (AP) and area under the ROC curve (AUC). We compared performances of NodeRep, DocRep, SPR2EP (Node embeddings (Node2vec) + Review embeddings (Doc2vec)) with SpEagle [43], method of Wang et al. [9] and random review ranking where the performance values are taken from [43]). Both SpEagle and method of Wang et al. are applied on the same dataset used in this study and with the same purpose, spam detection, and have made comparisons with previously proposed approaches. SpEagle [20] uses behavioural text and product-reviewer relation by applying manual feature engineering methods. Similar to our approach, SpEagle handles spam detection with methods that utilize both graph and textual information altogether. On the other hand, the approach of Wang et al. [9] uses only network information and does not use review text. Our approach has three different perspectives, and considers utilizing the underlying network structure, and review text individually, and then a combination of these two. Therefore, we have selected SpEagle and the approach of Wang et al. as the state-of-the art approaches for performance comparisons.

Table 4.4 provides performance results of the methods in terms of AP and AUC over all three datasets. Here only the best results obtained from three methods proposed here are given. Detailed performance results for each feature dimensions and classification algorithms can be found in Tables 4.5, 4.6 and 4.7 for YELPCHI, YELPNYC and YELPZIP datasets respectively. The result of NodeRep algorithm is quite high with respect to DocRep algorithm. This may be due to the fact that NodeRep algorithm utilizes both underlying network structure together with the review rating information, whereas DocRep only considers review text to generate feature vectors for the reviews.

Table 4.4. *AP and AUC Performances of Compared Methods on Datasets*

| | AP | | | AUC | | |
|---|---|---|---|---|---|---|
| | Y'CHI | Y'NYC | Y'ZIP | Y'CHI | Y'NYC | Y'ZIP |
| Random | 0.133 | 0.103 | 0.132 | 0.500 | 0.500 | 0.500 |
| Wang et. Al | 0.152 | 0.126 | 0.180 | 0.506 | 0.542 | 0.598 |
| SpEagle | 0.324 | 0.246 | 0.332 | 0.789 | 0.770 | 0.794 |
| NodeRep | 0.373 | 0.397 | 0.506 | 0.788 | 0.816 | 0.860 |
| DocRep | 0.276 | 0.224 | 0.296 | 0.715 | 0.725 | 0.734 |
| SPR2EP | 0.659 | 0.323 | 0.422 | 0.890 | 0.806 | 0.831 |

It is possible to conclude that, it is crucial for the spam review detection models to exploit the relational structure between reviewers and products for improving their performances. The resulting performance values for NodeRep models are also better than Random and method of Wang et.al [9] for every dataset. As well, NodeRep by itself obtains results that are very close to the results of the state-of-the-art SpEagle method. The most promising results are achieved by NodeRep and SPR2EP methods where the classifier is trained by node embeddings only and concatenating the embeddings obtained from Node2vec and Doc2vec algorithms altogether, respectively.

As it is stated, previous part compares our approach by using logistic regression with the existing algorithms. In this part, we will train our model and compare within our own experimental setting. To achieve this purpose, we propose 3 different parameter settings for feature size. In the previous part, we only used logistic regression classifier for making a fair comparison with the other approaches. As it is described, Logistic regression is one of the most basic algorithms. It has low computational effort, but it

is not perfect when it comes to predictability power. Therefore, we used other classification algorithms with Logistic Regression to see the performances under different classifiers and feature size settings. We applied different feature sizes and classifiers for all 3 datasets.

We didn't conduct hyperparameter tuning for classifiers and their parameters are given below;

- Logistic Regression (LR): L2 Regularization
- Decision Tree (CART): Splitting Criteria is Gini
- Gradient Boosting Machine (GBM): #of estimators is 100 and max depth=3
- Neural Network (NN): # of hidden layer is 100
- Random Forest (RF): #of estimators is 10

Performance measures that we used in this part are the same here, ROC AUC and Average Precision. For CHI Dataset, performances are given in Table 4.5 and the relations between performances are visualized in Figures 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9.

In CHI Dataset, Decision Tree classifier gives a poor performance for 3 approaches obviously. As it is stated before, Decision Tree uses Gini coefficient as a splitting criterion, and there is no max dept criterion is set. Best performances for NodeRep, DocRep and SPR2EP are observed at GBM, LR and NN, respectively. For NodeRep, there is an increasing trend when feature size increases. Min ROC AUC performance changes between 0.634 and 0.783 where feature size is 96. This range is quite wide, and when we exclude CART performance due to the low performance, ROC AUC range changes between 0.738 and 0.783 for feature size 96 for NodeRep. All of the algorithms except CART shows ROC AUC performance above 0.70. It is possible to say that they show fair performances in LR, NN, RF and GBM classifiers. Specifically, LR ROC AUC performance is 0.738 and NN, RF and GBM

performances between 0.773 and 0.783 for the feature size of 96. As a conclusion, NN, RF and GBM performances shows superior performance with respect to LR and CART in this feature size for NodeRep algorithm. For other feature sizes, 192 and 384, ROC AUC performances (except CART performance) are between 0.751 and 0.788 and between 0.757 and 0.798 respectively. It is observed that min and max ROC AUC performances increase when the feature size increases. NN ROC AUC performance of NodeRep decreases when feature size increases. As a result, GBM and RF performances increase with the feature size and they show better performances than other algorithms. There is need to note that this performance increase with feature size is a slight increase. For NodeRep, this behaviour will be investigated on other data sets. Average Precision is another performance measure, and its trend is compatible with ROC AUC performance. Both GBM and RF show better performance for AP similar to ROC AUC.

In DocRep for CHI Dataset, CART shows significantly low performance. Additionally, CART, RF and NN ROC AUC performances are below 0.70, which can be classified as poor performance. DocRep LR and GBM performances are very close to each other, and they show better performances. When GBM performance is stable on different dimensions, there is a slight increase in LR performances. Average Precision results show the same behavior with ROC AUC.

It can be concluded that LR and GBM show better performances for NodeRep and DocRep in CHI Dataset. NodeRep performances are better than DocRep performances. When the best ROC AUC performances of NodeRep are in 0.77-0.80 interval, the interval is 0.70-0.72 for DocRep. According to this comparison, it can be concluded that captured network information with NodeRep has more contribution than the captured review information with DocRep in the same feature sizes.

In the last setting, SPR2EP augments NodeRep and DocRep features and comparisons amongst all three approaches are done in the same feature size. CART performances of SPR2EP in CHI Dataset are poor like other approaches' CART performances. RF

performances are also quite low, similar to DocRep trend. For SPR2EP, LR, GBM and NN show better performances which are between 0.80 and 0.90 in all feature size settings and LR and GBM performances are very close to each other. ROC AUC performance interval of SPR2EP are above the other best performance intervals of algorithms. Combining network and review features boosted ROC AUC performance. However, this effect can be seen in AP performances more dramatically. SPR2EP AP performances for NodeRep, DocRep and SPR2EP are in 0.341-0.373, 0.26-0.276, 0.631-0.659 ranges, respectively. For YELPCHI Dataset, SPR2EP shows higher performance than NodeRep and DocRep.

Table 4.5. *AP and AUC Performances of different classifiers for 3 dimensions on YELPCHI Dataset*

| | | NodeRep | | DocRep | | SPR2EP | |
|---|---|---|---|---|---|---|---|
| Feature Dimension | Classifier | AUC | AP | AUC | AP | AUC | AP |
| 96 | LR | 0.738 | 0.291 | 0.704 | 0.260 | 0.807 | 0.436 |
| 96 | CART | 0.634 | 0.214 | 0.520 | 0.138 | 0.600 | 0.183 |
| 96 | GBM | 0.783 | 0.338 | 0.703 | 0.259 | 0.833 | 0.472 |
| 96 | NN | 0.774 | 0.323 | 0.643 | 0.204 | 0.890 | 0.659 |
| 96 | RF | 0.773 | 0.341 | 0.565 | 0.159 | 0.774 | 0.374 |
| 192 | LR | 0.751 | 0.306 | 0.710 | 0.268 | 0.814 | 0.453 |
| 192 | CART | 0.640 | 0.221 | 0.515 | 0.137 | 0.599 | 0.182 |
| 192 | GBM | 0.788 | 0.346 | 0.704 | 0.254 | 0.832 | 0.461 |
| 192 | NN | 0.766 | 0.301 | 0.638 | 0.196 | 0.878 | 0.624 |
| 192 | RF | 0.777 | 0.350 | 0.549 | 0.152 | 0.767 | 0.355 |
| 384 | LR | 0.757 | 0.312 | 0.715 | 0.276 | 0.828 | 0.477 |
| 384 | CART | 0.640 | 0.222 | 0.511 | 0.135 | 0.601 | 0.183 |
| 384 | GBM | 0.793 | 0.352 | 0.701 | 0.250 | 0.828 | 0.456 |
| 384 | NN | 0.766 | 0.294 | 0.648 | 0.205 | 0.880 | 0.631 |
| 384 | RF | 0.788 | 0.373 | 0.542 | 0.149 | 0.748 | 0.328 |

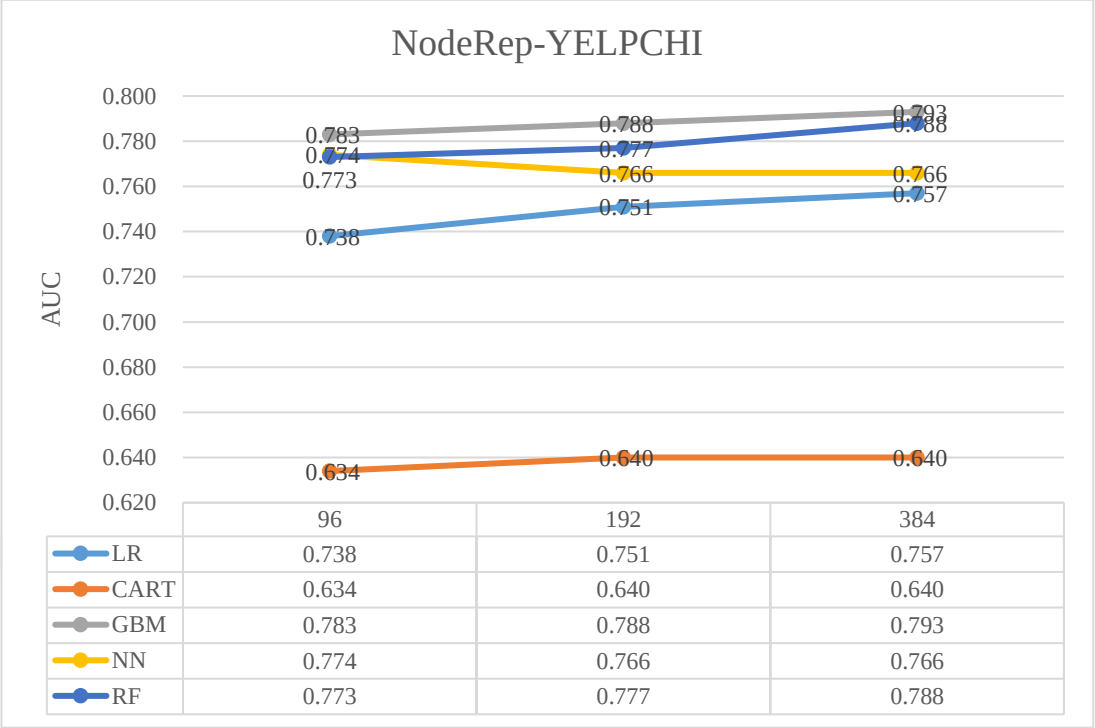



*Figure 4.4.* ROC AUC Performances on YELP CHI Dataset for NodeRep

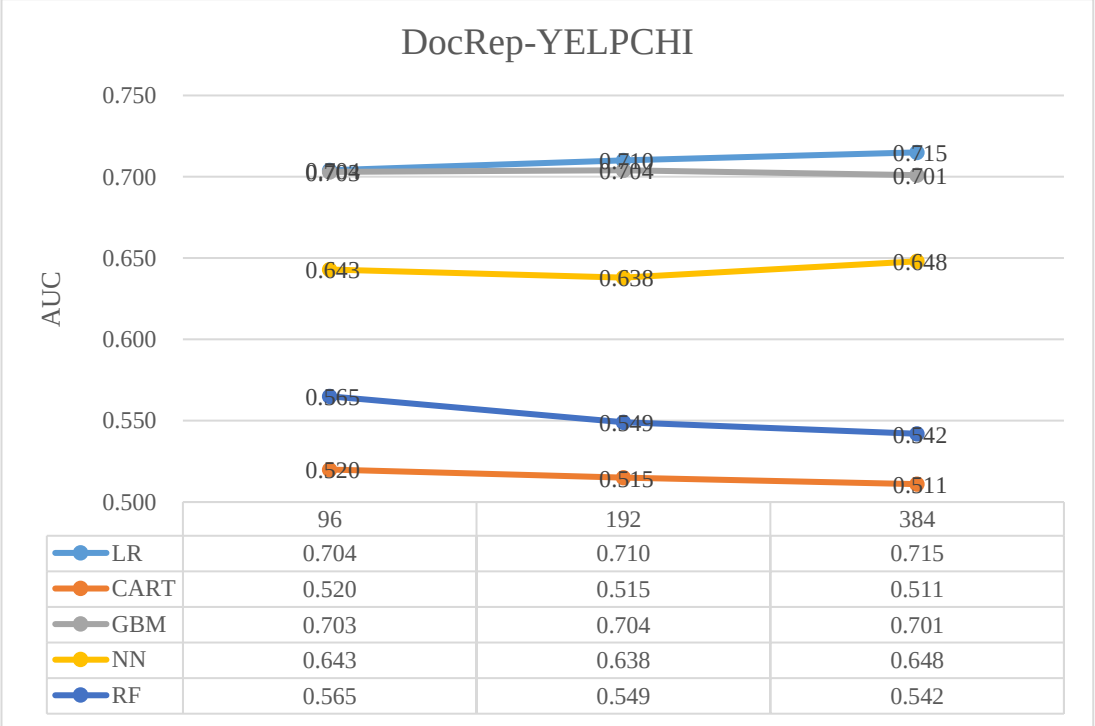
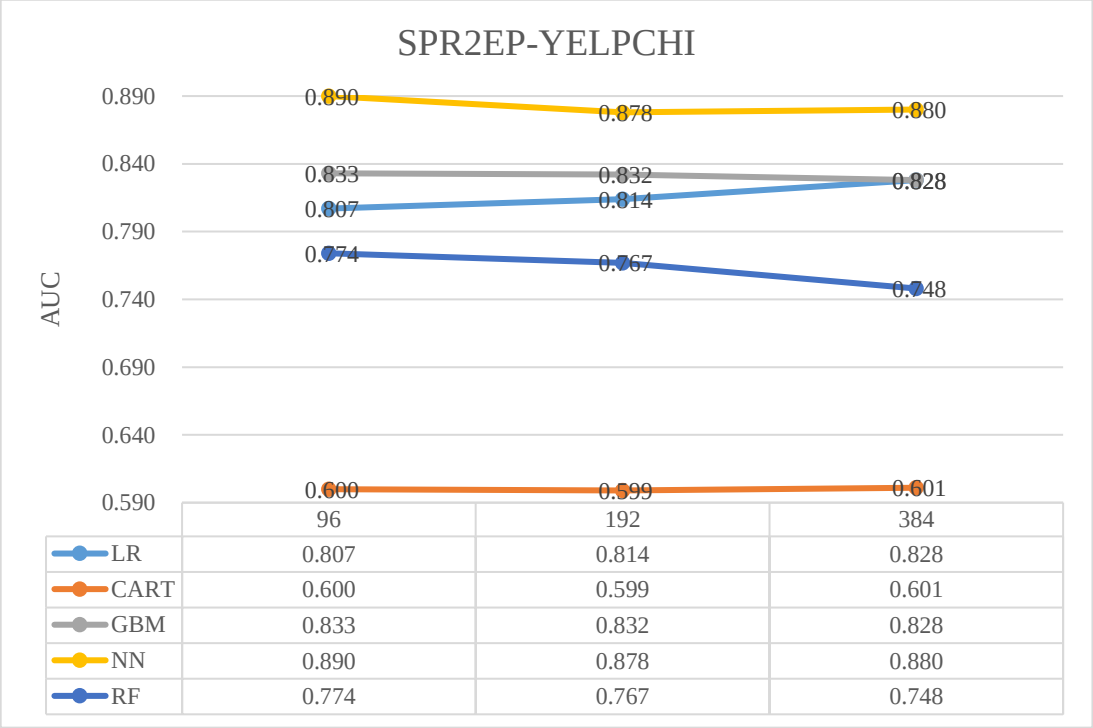
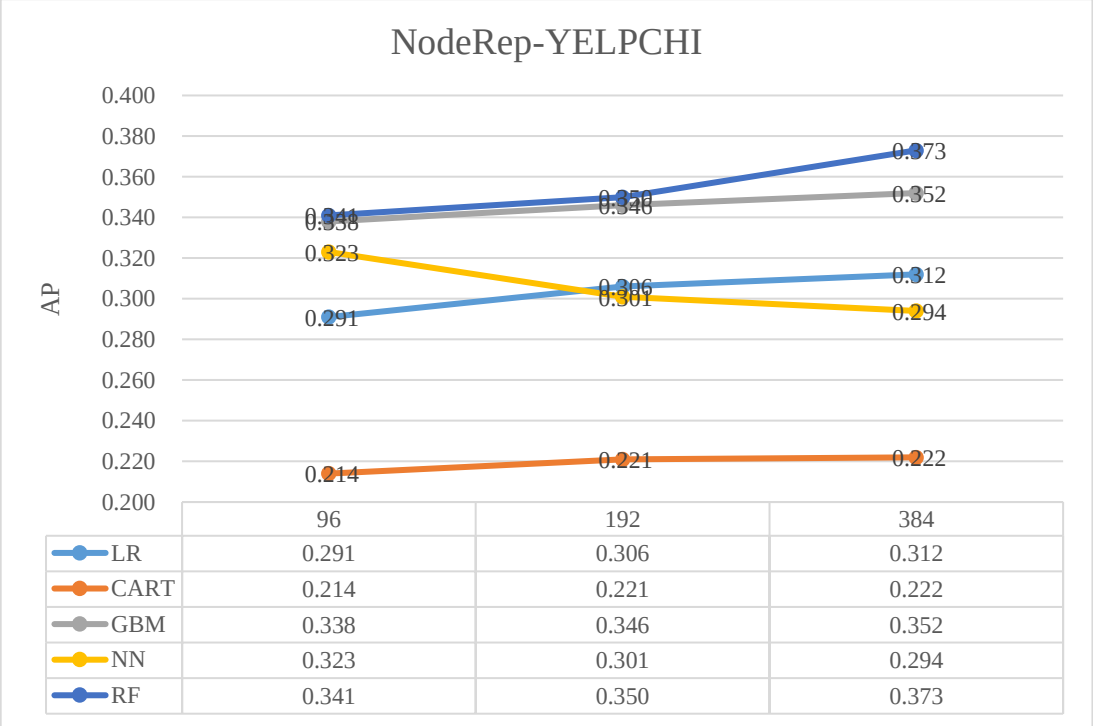



*Figure 4.5.* ROC AUC Performances on YELP CHI Dataset for DocRep



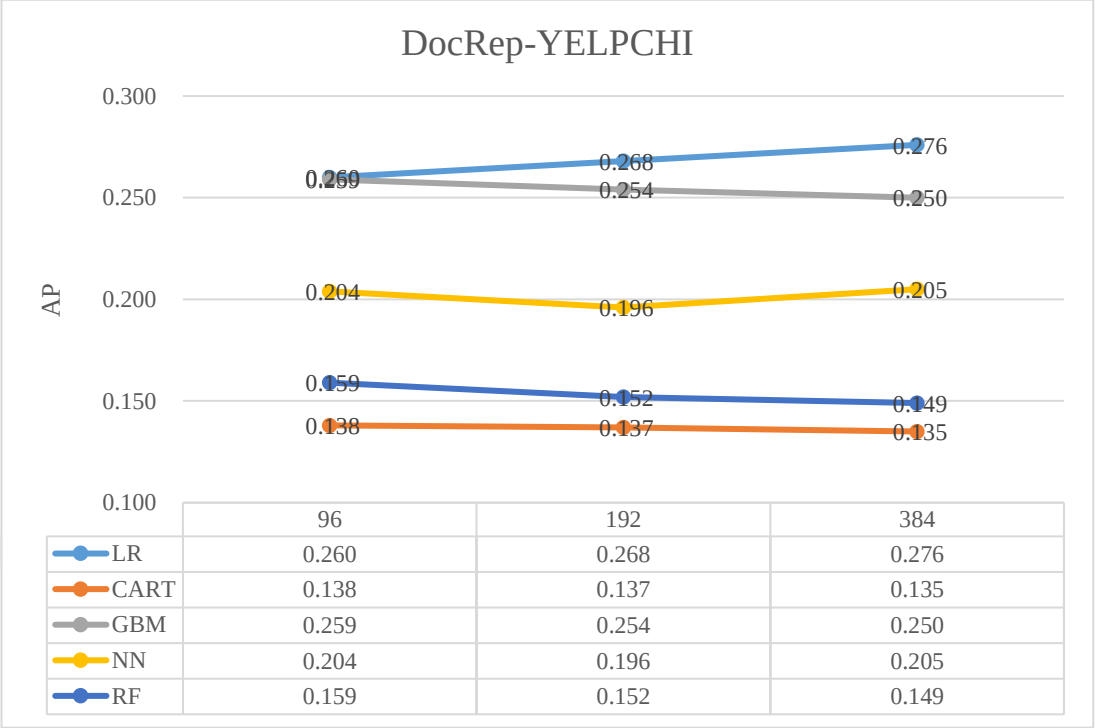

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.807 | 0.814 | 0.828 |
| CART | 0.600 | 0.599 | 0.601 |
| GBM | 0.833 | 0.832 | 0.828 |
| NN | 0.890 | 0.878 | 0.880 |
| RF | 0.774 | 0.767 | 0.748 |

*Figure 4.6.* ROC AUC Performances on YELP CHI Dataset for SPR2EP



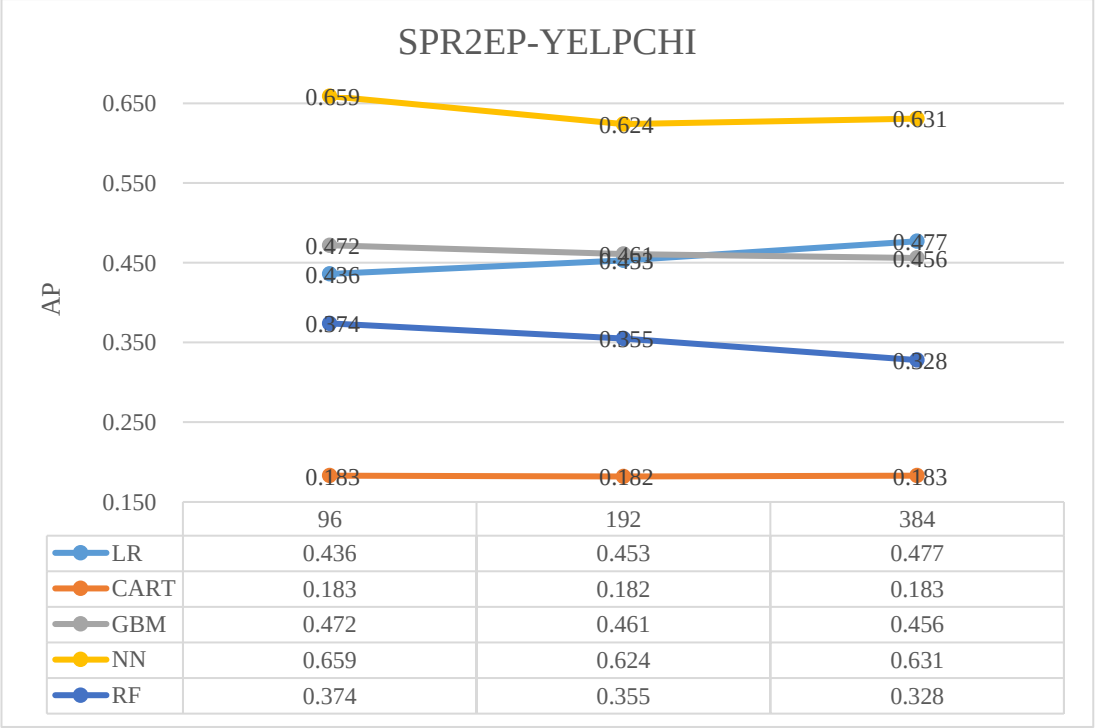

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.291 | 0.306 | 0.312 |
| CART | 0.214 | 0.221 | 0.222 |
| GBM | 0.338 | 0.346 | 0.352 |
| NN | 0.323 | 0.301 | 0.294 |
| RF | 0.341 | 0.350 | 0.373 |

*Figure 4.7.* AP Performances on YELP CHI Dataset for NodeRep




| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.260 | 0.268 | 0.276 |
| CART | 0.138 | 0.137 | 0.135 |
| GBM | 0.259 | 0.254 | 0.250 |
| NN | 0.204 | 0.196 | 0.205 |
| RF | 0.159 | 0.152 | 0.149 |

*Figure 4.8.* AP Performances on YELP CHI Dataset for DocRep




| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.436 | 0.453 | 0.477 |
| CART | 0.183 | 0.182 | 0.183 |
| GBM | 0.472 | 0.461 | 0.456 |
| NN | 0.659 | 0.624 | 0.631 |
| RF | 0.374 | 0.355 | 0.328 |

*Figure 4.9.* AP Performances on YELP CHI Dataset for SPR2EP

For NYC Dataset, performances are given in Table 4.6 and the relations between performances are visualized in Figures 4.10, 4.11, 4.12, 4.13, 4.14 and 4.15. In NYC Dataset, Decision Tree classifier gives a poor performance for three approaches. Best performances for NodeRep, DocRep and SPR2EP are observed at RF, LR and NN, respectively. For the NodeRep, there is an increasing trend when feature size increases. Min ROC AUC performance changes between 0.647 and 0.813 where feature size is 96. This range is quite wide, and when we exclude CART performance due to the low performance, ROC AUC range changes between 0.747 and 0.783 for feature size 96 for NodeRep. All of the algorithms except CART shows ROC AUC performance above 0.70. It is possible to say that they show fair performances in LR, NN, RF and GBM classifiers. Specifically, LR ROC AUC performance is 0.747 and NN, RF and GBM performances between 0.783 and 0.813 for the feature size of 96. As a conclusion, NN, RF and GBM show superior performance with respect to LR and CART in this feature size for NodeRep algorithm. For other feature sizes, 192 and 384, ROC AUC performances (except CART performance) are between 0.759 and 0.818 and between 0.763 and 0.816 respectively. It is observed that min and max ROC AUC performances don't show any trend when feature size increases. As a result, NN and RF algorithms show better performances than other algorithms. Average Precision is another performance measure, and its trend is compatible with ROC AUC performance. However, AP performance of RF is significantly higher than other algorithms for NodeRep.

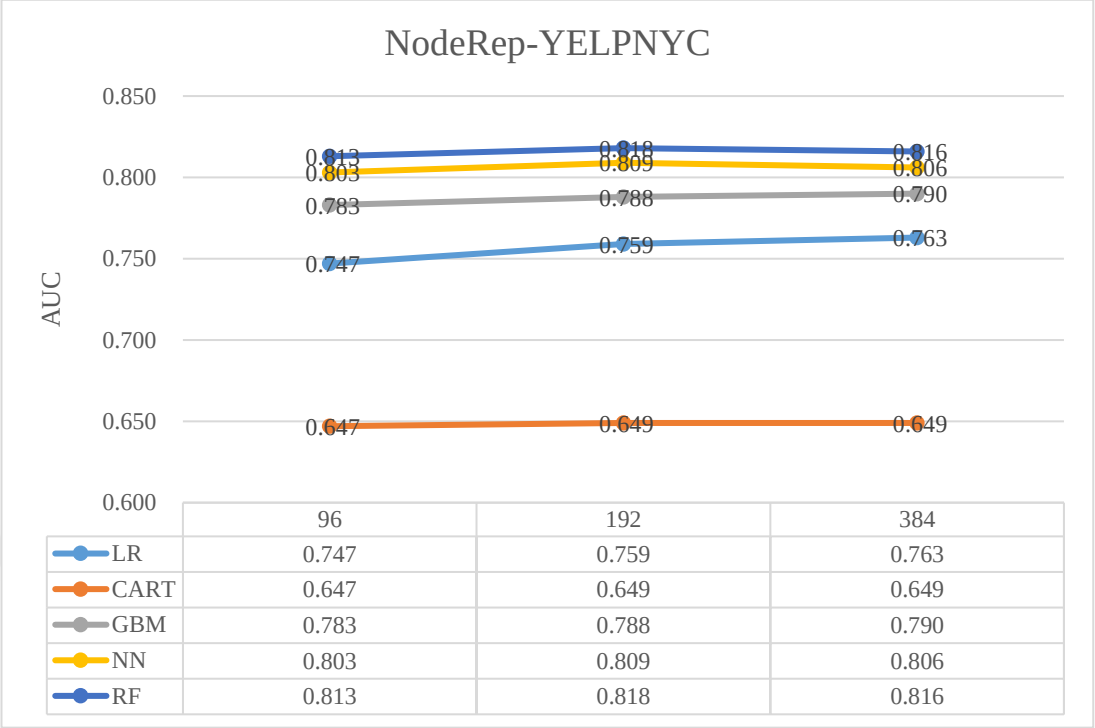
In DocRep for NYC Dataset, CART shows significantly low performance. Additionally, CART and RF ROC AUC performances are below 0.70, which can be classified as poor performance. This behavior is also seen in CHI Dataset. DocRep LR and GBM performances are very close to each other, and they show better performances. NN performances show a different trend in NYC Dataset when it is compared with CHI Dataset. When GBM performance is stable on different dimensions, there is a slight increase in LR performances. Average Precision results show the same behavior with ROC AUC.

It can be concluded that RF and NN show better performances for NodeRep and DocRep in NYC Dataset. NodeRep performances are better than DocRep performances. When the best ROC AUC performances of NodeRep are in 0.81-0.82 interval, the performance interval is 0.71-0.73 for DocRep. According to this comparison, it can be concluded that captured network information with NodeRep has more contribution than the captured review information with DocRep in the same feature sizes for NYC dataset. The same behavior is observed in CHI Dataset.

In the last setting, SPR2EP augments NodeRep and DocRep features and comparisons amongst all three approaches are conducted in the same feature size for NYC Dataset. CART performances of SPR2EP in NYC Dataset are poor, like other approaches' CART performances. CART performance behavior of SPR2EP is also similar to CHI Dataset. RF performances are also quite low, similar to DocRep trend. For SPR2EP, LR, GBM and NN show better performances which are between 0.77 and 0.81 in all feature size settings and LR and GBM performances are very close to each other. ROC AUC performance interval of SPR2EP are very close to the best performance intervals of NodeRep but quite lower than it. It shows better performance than DocRep.

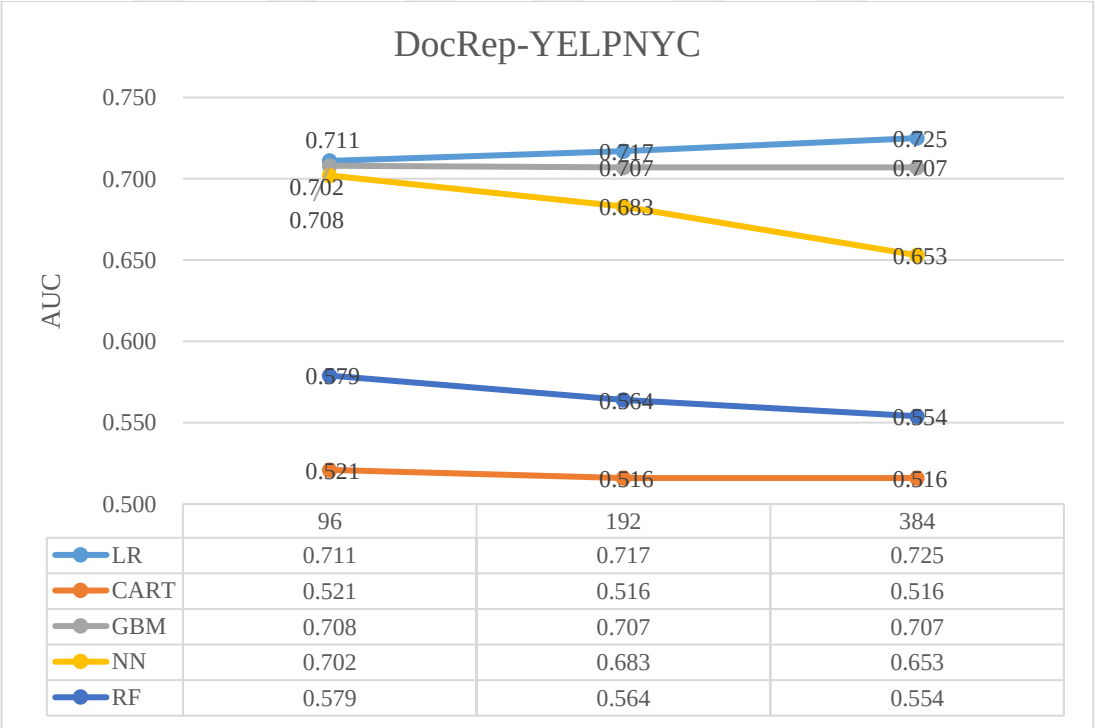
Table 4.6. *AP and AUC Performances of different classifiers for 3 dimensions on YELP NYC Dataset*

| | | NodeRep | | DocRep | | SPR2EP | |
|---|---|---|---|---|---|---|---|
| Feature Dimension | Classifier | AUC | AP | AUC | AP | AUC | AP |
| 96 | LR | 0.747 | 0.228 | 0.711 | 0.209 | 0.779 | 0.266 |
| 96 | CART | 0.647 | 0.194 | 0.521 | 0.108 | 0.583 | 0.138 |
| 96 | GBM | 0.783 | 0.276 | 0.708 | 0.214 | 0.797 | 0.300 |
| 96 | NN | 0.803 | 0.314 | 0.702 | 0.209 | 0.806 | 0.323 |
| 96 | RF | 0.813 | 0.371 | 0.579 | 0.132 | 0.736 | 0.248 |
| 192 | LR | 0.759 | 0.239 | 0.717 | 0.217 | 0.788 | 0.279 |
| 192 | CART | 0.649 | 0.196 | 0.516 | 0.106 | 0.596 | 0.147 |
| 192 | GBM | 0.788 | 0.290 | 0.707 | 0.213 | 0.802 | 0.305 |
| 192 | NN | 0.809 | 0.317 | 0.683 | 0.187 | 0.801 | 0.308 |
| 192 | RF | 0.818 | 0.391 | 0.564 | 0.126 | 0.759 | 0.287 |
| 384 | LR | 0.763 | 0.248 | 0.725 | 0.224 | 0.797 | 0.296 |
| 384 | CART | 0.649 | 0.195 | 0.516 | 0.106 | 0.609 | 0.157 |
| 384 | GBM | 0.790 | 0.293 | 0.707 | 0.213 | 0.807 | 0.310 |
| 384 | NN | 0.806 | 0.312 | 0.653 | 0.158 | 0.781 | 0.273 |
| 384 | RF | 0.816 | 0.397 | 0.554 | 0.122 | 0.767 | 0.307 |



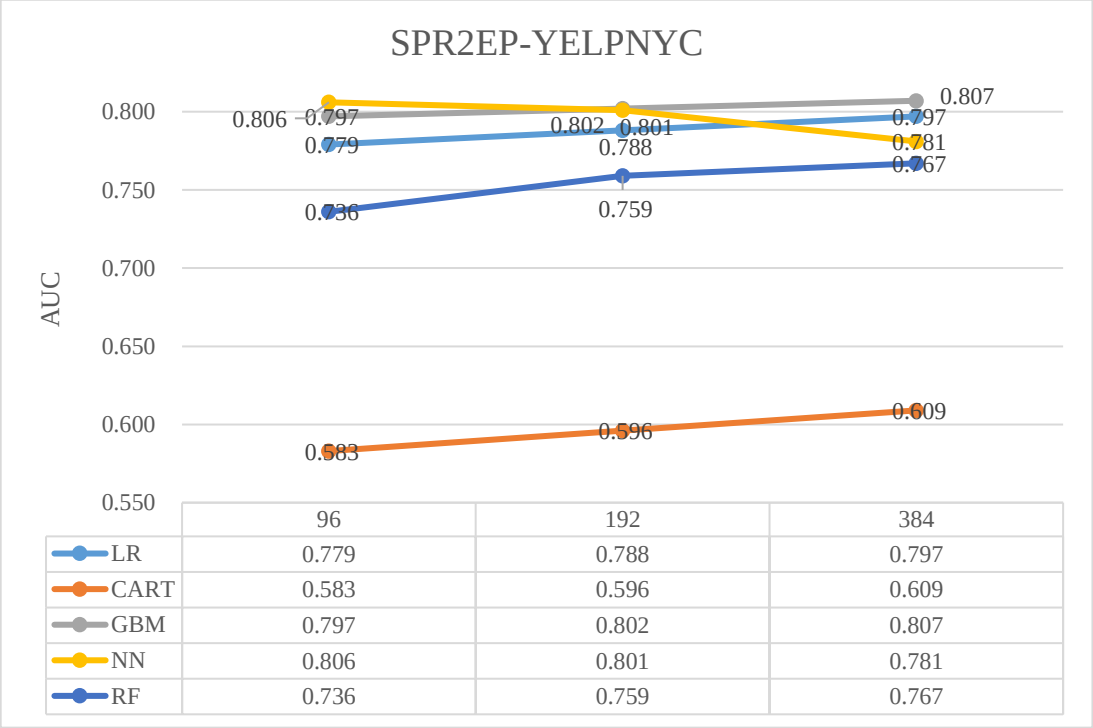

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.747 | 0.759 | 0.763 |
| CART | 0.647 | 0.649 | 0.649 |
| GBM | 0.783 | 0.788 | 0.790 |
| NN | 0.803 | 0.809 | 0.806 |
| RF | 0.813 | 0.818 | 0.816 |

*Figure 4.10.* ROC AUC Performances on YELP NYC Dataset for NodeRep



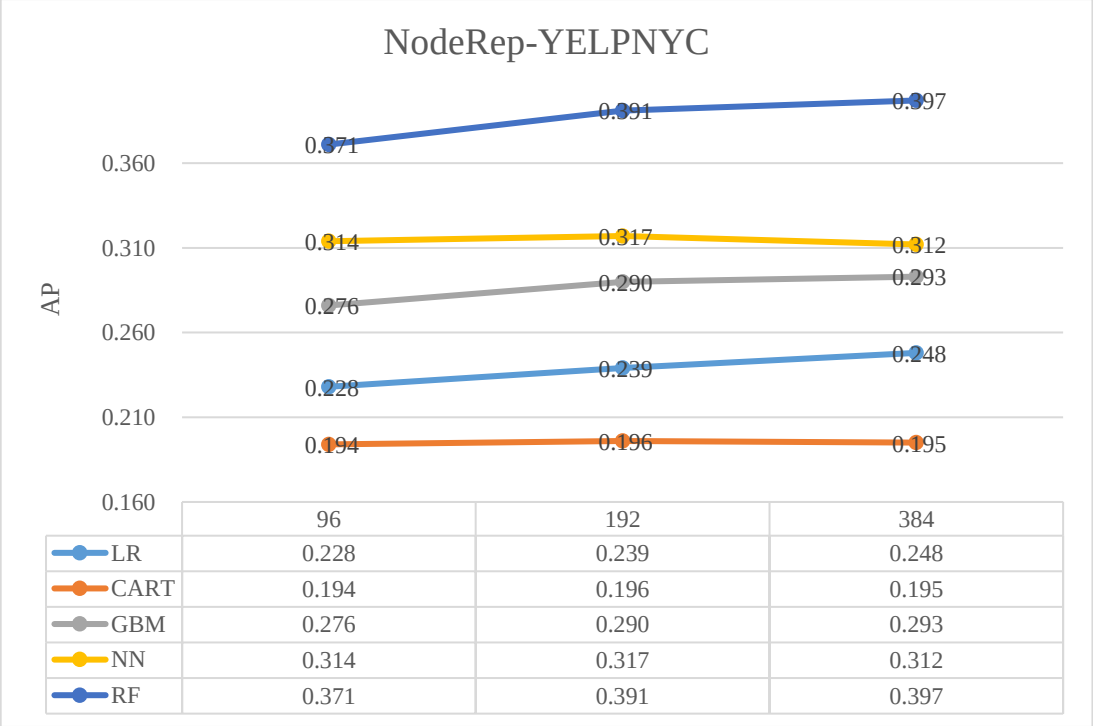

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.711 | 0.717 | 0.725 |
| CART | 0.521 | 0.516 | 0.516 |
| GBM | 0.708 | 0.707 | 0.707 |
| NN | 0.702 | 0.683 | 0.653 |
| RF | 0.579 | 0.564 | 0.554 |

*Figure 4.11.* ROC AUC Performances on YELP NYC Dataset for DocRep




| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.779 | 0.788 | 0.797 |
| CART | 0.583 | 0.596 | 0.609 |
| GBM | 0.797 | 0.802 | 0.807 |
| NN | 0.806 | 0.801 | 0.781 |
| RF | 0.736 | 0.759 | 0.767 |

*Figure 4.12.* ROC AUC Performances on YELP NYC Dataset for SPR2EP




| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.228 | 0.239 | 0.248 |
| CART | 0.194 | 0.196 | 0.195 |
| GBM | 0.276 | 0.290 | 0.293 |
| NN | 0.314 | 0.317 | 0.312 |
| RF | 0.371 | 0.391 | 0.397 |

*Figure 4.13.* AP Performances on YELP NYC Dataset for NodeRep

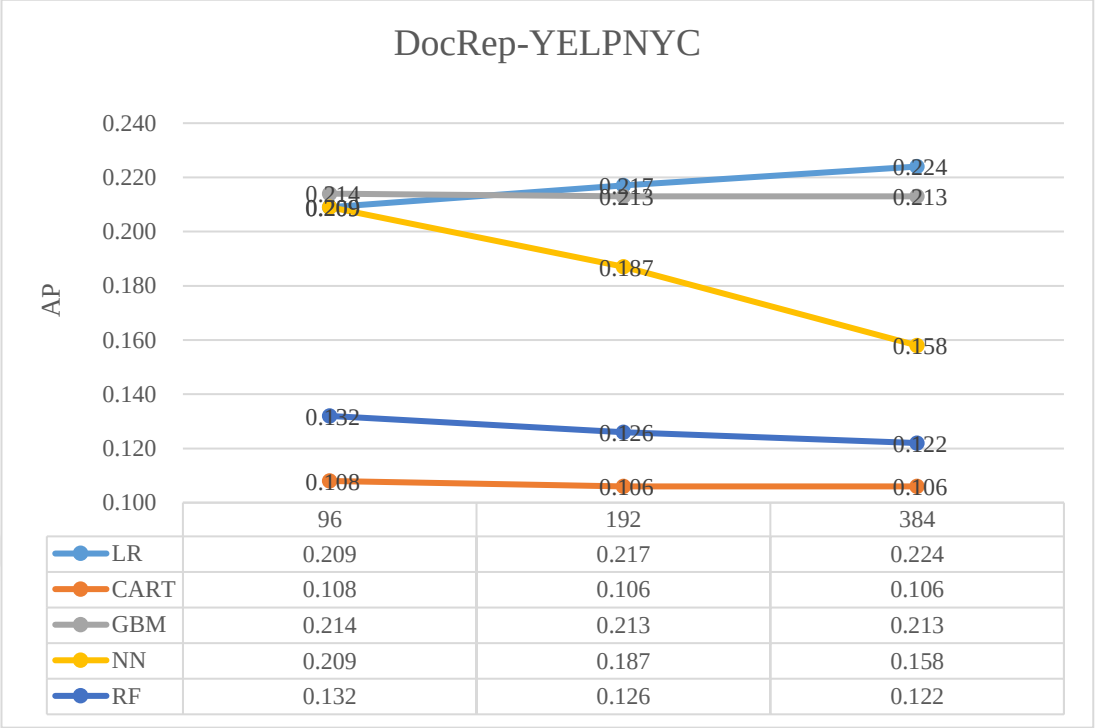



*Figure 4.14.* AP Performances on YELP NYC Dataset for DocRep

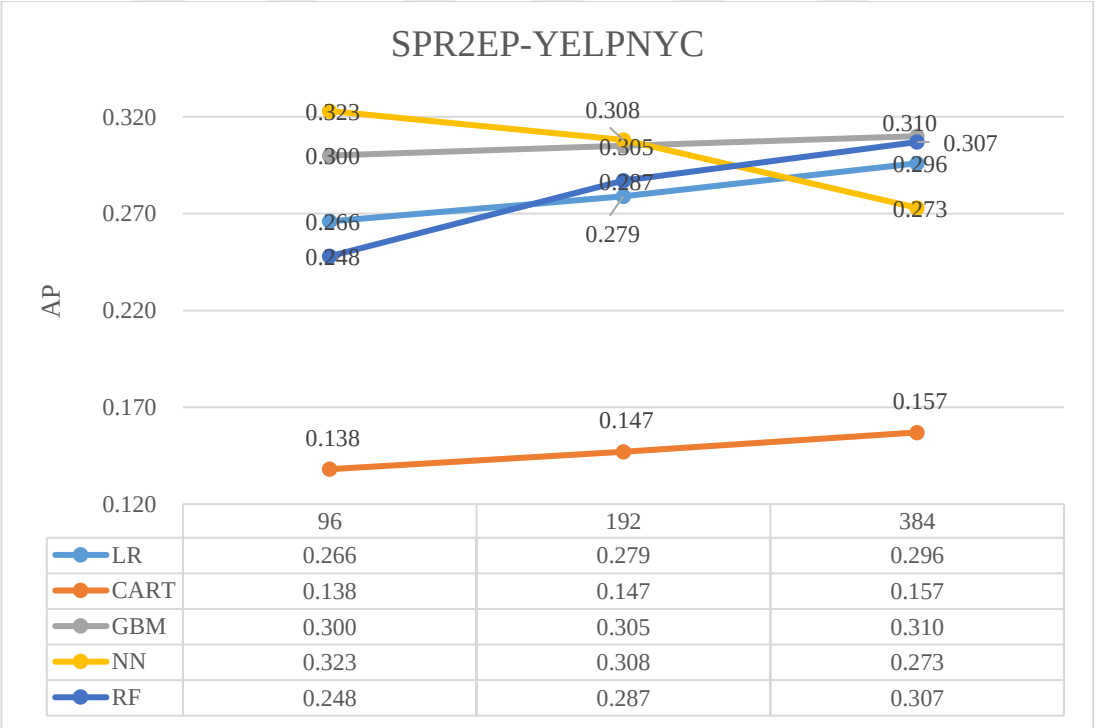



*Figure 4.15.* AP Performances on YELP NYC Dataset for SPR2EP

For ZIP Dataset, performances are given in Table 4.7 and the relations between performances are visualized in Figures 4.16, 4.17, 4.18, 4.19, 4.20 and 4.21. In ZIP Dataset, Decision Tree classifier gives a poor performance for 3 approaches obviously. Best performances for NodeRep, DocRep and SPR2EP are observed at RF, LR and NN, respectively. For the NodeRep, there is an increasing trend when feature size increases. Min ROC AUC performance changes between 0.686 and 0.86 where feature size is 96. This range is quite wide, and when we exclude CART performance due to the low performance, ROC AUC range changes between 0.781 and 0.86 for feature size 96 for NodeRep. All of the algorithms except CART shows ROC AUC performance above 0.78. It is possible to say that they show good performances in LR, NN, RF and GBM classifiers. Specifically, LR ROC AUC performance is 0.781 and NN, RF and GBM performances between 0.801 and 0.85 for the feature size of 96. As a conclusion, NN, RF and GBM performances show superior performance with respect to LR and CART in this feature size for NodeRep algorithm. This is also the case for other datasets. For different feature sizes, 192 and 384, ROC AUC performances (except CART performance) are between 0.784 and 0.856 and between 0.789 and 0.86 respectively. It is observed that min and max ROC AUC performances don't show any trend when feature size increases. Performances in different feature sizes are very close to each other for NodeRep. Average Precision is another performance measure, and its trend is compatible with ROC AUC performance. However, AP performance of RF is significantly higher than other algorithms for NodeRep.

In DocRep for ZIP Dataset, CART shows significantly low performance. Additionally, CART and RF ROC AUC performances are below 0.70. This can be classified as poor performance. This behavior is also seen in CHI and NYC Datasets. DocRep LR and GBM performances are very close to each other, and they show better performances. NN performances show the same trend in ZIP Dataset when it is compared with NYC Dataset. When GBM performance is stable on different dimensions, there is a slight increase in LR performances. This is also similar to NYC

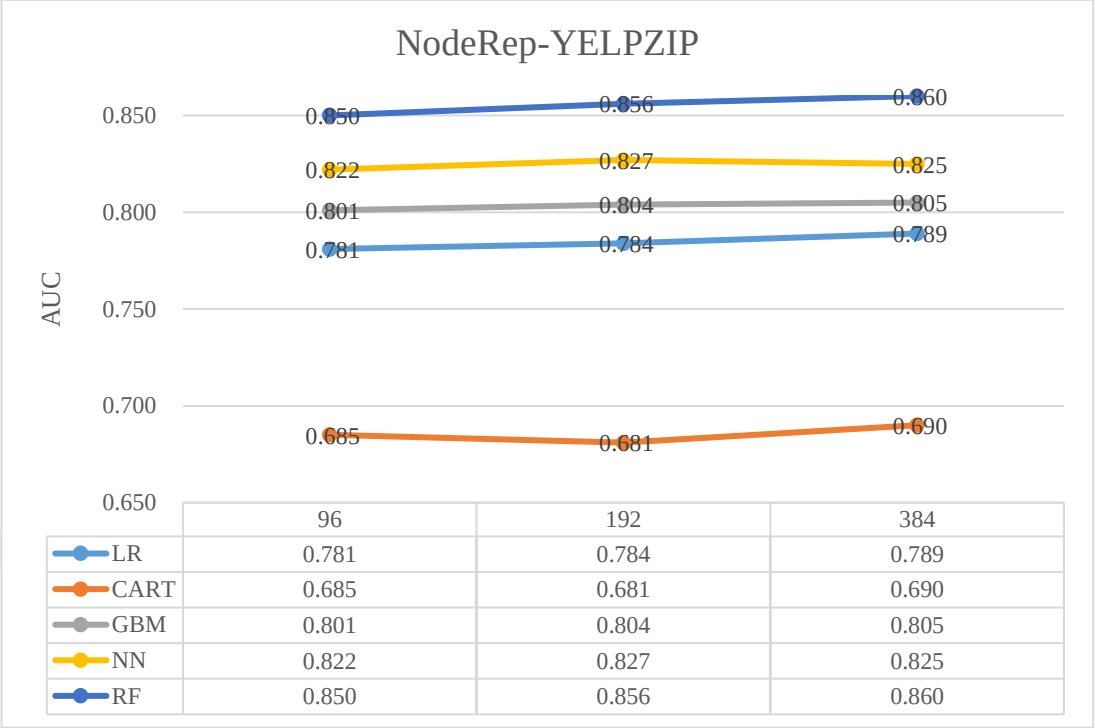
Dataset, but it is different than CHI Dataset. Average Precision results show the same behavior with ROC AUC.

It can be concluded that RF and NN shows better performances for NodeRep in ZIP Dataset. NodeRep performances are better than DocRep performances. When the best ROC AUC performances of NodeRep are in 0.85-0.86 interval, the performance interval is 0.72-0.74 for DocRep. According to this comparison, it can be concluded that captured network information with NodeRep has more contribution than the captured review information with DocRep in the same feature sizes for ZIP dataset. The same behavior is observed in other datasets.

In the last setting, SPR2EP augments NodeRep and DocRep features and comparisons amongst all three approaches are done in the same feature size for ZIP Dataset. CART performances of SPR2EP in ZIP Dataset are poor, like other approaches' CART performances. CART performance behavior of SPR2EP is also similar to other datasets. For SPR2EP, all algorithms except CART show good performances which are between 0.79 and 0.83 in all feature size settings and LR and GBM performances are very close to each other. ROC AUC performance interval of SPR2EP are very close to the best performance intervals of NodeRep but quite lower than it. It shows better performance than DocRep.

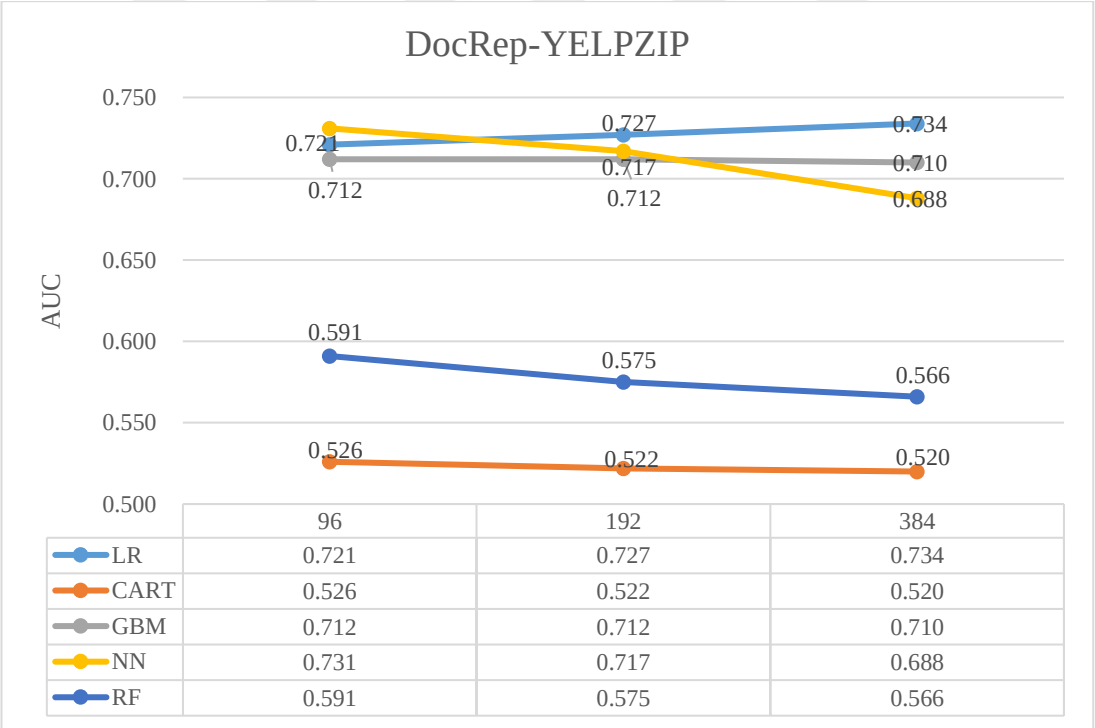
Table 4.7. *AP and AUC Performances of different classifiers for 3 dimensions on YELP ZIP Dataset*

| | | NodeRep | | DocRep | | SPR2EP | |
|---|---|---|---|---|---|---|---|
| Feature Dimension | Classifier | AUC | AP | AUC | AP | AUC | AP |
| 96 | LR | 0.781 | 0.339 | 0.721 | 0.280 | 0.803 | 0.373 |
| 96 | CART | 0.685 | 0.273 | 0.526 | 0.140 | 0.628 | 0.208 |
| 96 | GBM | 0.801 | 0.362 | 0.712 | 0.269 | 0.811 | 0.379 |
| 96 | NN | 0.822 | 0.408 | 0.731 | 0.293 | 0.831 | 0.422 |
| 96 | RF | 0.850 | 0.475 | 0.591 | 0.173 | 0.797 | 0.362 |
| 192 | LR | 0.784 | 0.342 | 0.727 | 0.289 | 0.809 | 0.382 |
| 192 | CART | 0.681 | 0.268 | 0.522 | 0.139 | 0.646 | 0.226 |
| 192 | GBM | 0.804 | 0.364 | 0.712 | 0.266 | 0.813 | 0.380 |
| 192 | NN | 0.827 | 0.415 | 0.717 | 0.275 | 0.828 | 0.414 |
| 192 | RF | 0.856 | 0.493 | 0.575 | 0.165 | 0.817 | 0.404 |
| 384 | LR | 0.789 | 0.347 | 0.734 | 0.296 | 0.811 | 0.387 |
| 384 | CART | 0.690 | 0.280 | 0.520 | 0.138 | 0.644 | 0.224 |
| 384 | GBM | 0.805 | 0.368 | 0.710 | 0.263 | 0.815 | 0.380 |
| 384 | NN | 0.825 | 0.408 | 0.688 | 0.238 | 0.814 | 0.383 |
| 384 | RF | 0.860 | 0.506 | 0.566 | 0.160 | 0.821 | 0.419 |



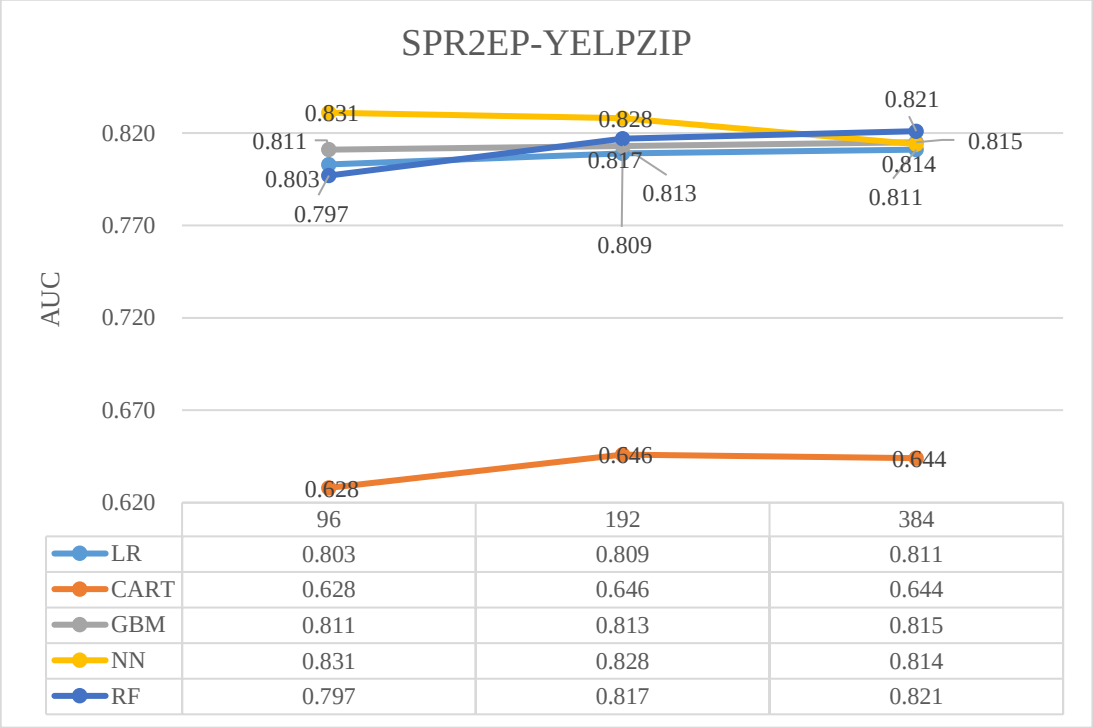

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.781 | 0.784 | 0.789 |
| CART | 0.685 | 0.681 | 0.690 |
| GBM | 0.801 | 0.804 | 0.805 |
| NN | 0.822 | 0.827 | 0.825 |
| RF | 0.850 | 0.856 | 0.860 |

*Figure 4.16.* ROC AUC Performances on YELP ZIP Dataset for NodeRep



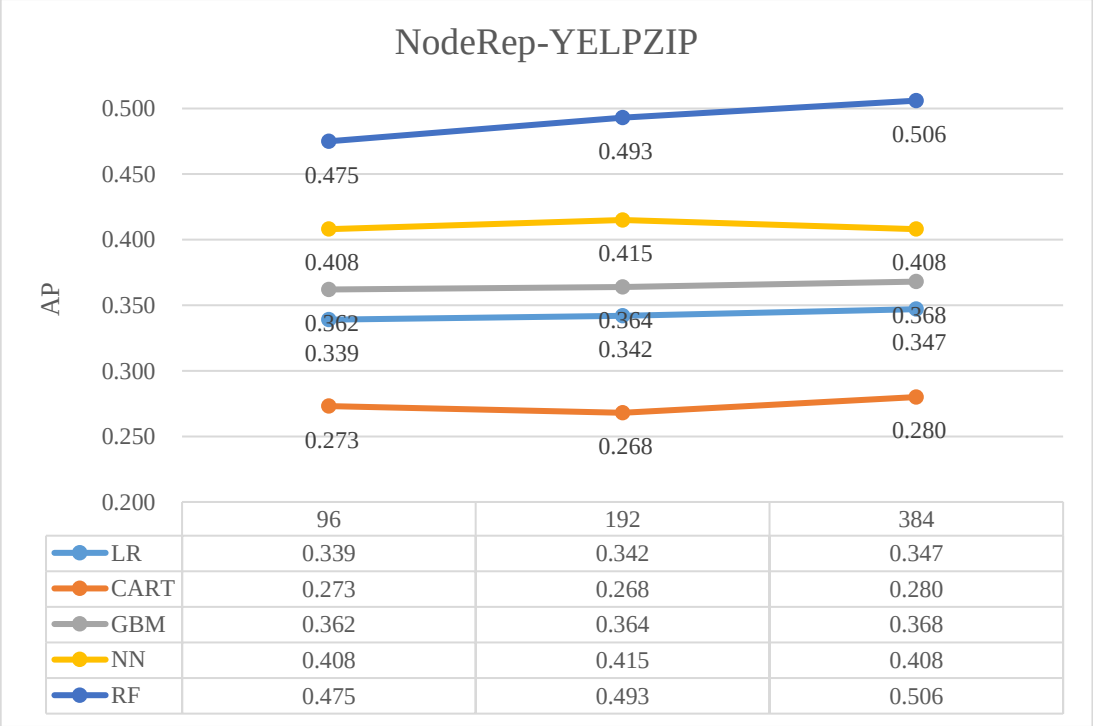

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.721 | 0.727 | 0.734 |
| CART | 0.526 | 0.522 | 0.520 |
| GBM | 0.712 | 0.712 | 0.710 |
| NN | 0.731 | 0.717 | 0.688 |
| RF | 0.591 | 0.575 | 0.566 |

*Figure 4.17.* ROC AUC Performances on YELP ZIP Dataset for DocRep



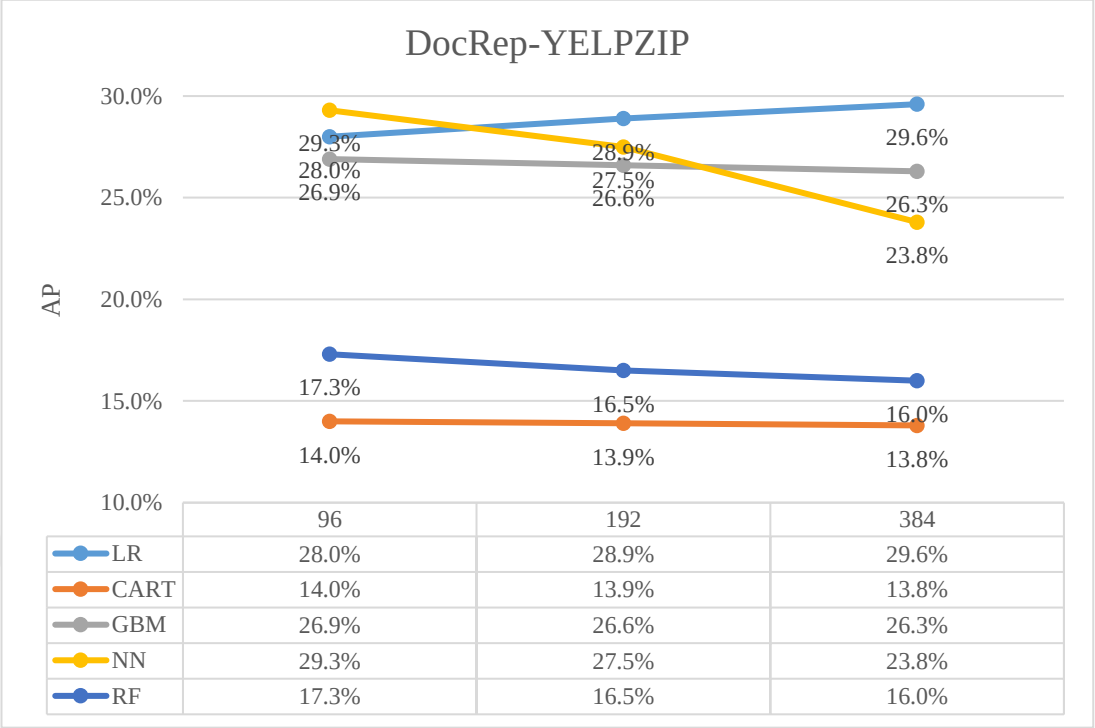

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.803 | 0.809 | 0.811 |
| CART | 0.628 | 0.646 | 0.644 |
| GBM | 0.811 | 0.813 | 0.815 |
| NN | 0.831 | 0.828 | 0.814 |
| RF | 0.797 | 0.817 | 0.821 |

*Figure 4.18.* ROC AUC Performances on YELP ZIP Dataset for SPR2EP



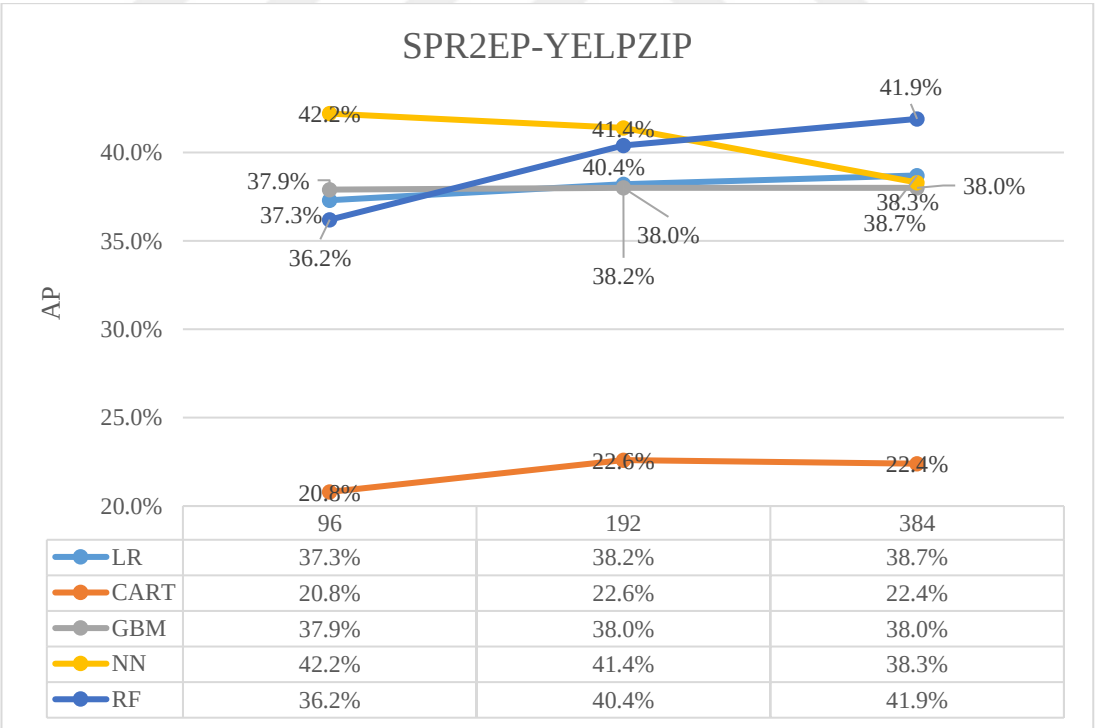

| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 0.339 | 0.342 | 0.347 |
| CART | 0.273 | 0.268 | 0.280 |
| GBM | 0.362 | 0.364 | 0.368 |
| NN | 0.408 | 0.415 | 0.408 |
| RF | 0.475 | 0.493 | 0.506 |

*Figure 4.19.* AP Performances on YELP ZIP Dataset for NodeRep




| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 28.0% | 28.9% | 29.6% |
| CART | 14.0% | 13.9% | 13.8% |
| GBM | 26.9% | 26.6% | 26.3% |
| NN | 29.3% | 27.5% | 23.8% |
| RF | 17.3% | 16.5% | 16.0% |

*Figure 4.20.* AP Performances on YELP ZIP Dataset for DocRep




| | 96 | 192 | 384 |
|---|---|---|---|
| LR | 37.3% | 38.2% | 38.7% |
| CART | 20.8% | 22.6% | 22.4% |
| GBM | 37.9% | 38.0% | 38.0% |
| NN | 42.2% | 41.4% | 38.3% |
| RF | 36.2% | 40.4% | 41.9% |

*Figure 4.21.* AP Performances on YELP ZIP Dataset for SPR2EP

As a conclusion of all comparisons, there are some highlights on the performances. CART performances of all algorithms are very poor. DocRep shows the lowest performances amongst the three approaches. NodeRep and SPR2EP performances are close to each other. It is observed that SPR2EP shows better performance than NodeRep performance in CHI Dataset. In NYC and ZIP Datasets, NodeRep performances are quite better than SPR2EP algorithm. However, it can be concluded that SPR2EP and NodeRep show performances which are close to each other. For all algorithms and datasets, LR performances are increases with the feature sizes. RF algorithm can be selected as the best performing algorithm for NodeRep based on the performances on three different datasets. Similarly, LR for DocRep and NN for SPR2EP show the best performances. Increasing feature size does not have a significant effect on the performances.

# CHAPTER 5

## CONCLUSION AND FUTURE WORK

In this study, a comprehensive comparison of spam review detection approaches is performed by taking into account the reviewer-product network together with review texts. In this respect, low-dimensional dense feature vectors are automatically learned without applying any hand-crafted feature engineering effort in an unsupervised manner. Then, these are used to build various binary classification models to distinguish spam reviews from authentic ones.

In the network-only approach, only the learned network embeddings are used as input features whereas in the review-only approach, only the features learned for the review texts are used in building respective spam review detection classifiers.

Three processes constitute the last approach, a semi-supervised learning framework named SPR2EP, where first two processes create node embeddings from the network data, and document embeddings from the reviews' textual data which can be run independently in parallel. The last process integrates the embeddings learned in the previous steps by concatenating them into feature vectors for the reviews, and uses them to train a classifier.

In order to investigate the effectiveness of utilizing the information extracted from the review text and the underlying network structure, we built binary classification models for each approach with varying embedding sizes and different classification algorithms. Then, we compare their performances against each other and the existing techniques proposed in the literature, as well.

Accordingly, the first model, DocRep, is built by using review embeddings learned by using only review texts. The second model, NodeRep, is created by using reviewer and product embeddings learned from the underlying reviewer-product network structure. And the last one, SPR2EP, is built by utilizing combined feature vectors of

the reviews which are obtained by concatenating the embeddings learned for review text, reviewer and product.

All three models are built using five different classification methods and for three different feature dimensions and comparisons are made by considering AP, average precision and AUC, area under the ROC curve performance measures obtained via 10-fold cross-validation.

The experimental results show that the detection model that is built by utilizing the combined feature vectors, SPR2EP, achieves the best performance, with AUC values of 0.890, for CHI Dataset. On the other hand, models generated with only utilizing the node embeddings, NodeRep, achieved the best performance for NYC and ZIP datasets with AUC values of 0.816 and 0.860, respectively. Although the best performing models are SPR2EP for CHI Dataset and NodeRep for the other datasets, their performances are very close to each other.

The huge gap between the performances of DocRep approach and the other two approaches provides an evidence for the effectiveness of utilizing underlying network structure in spam review detection. So, this observation implies that bringing the features extracted from network and textual data together and utilizing them in creating a machine learning model would exhibit improved spam detection capability if the network size is small. Otherwise, one should exploit the information contained in the underlying network structure which happens to be more valuable than the one obtained from textual content.

To conclude, our results show the fact that NodeRep and SPR2EP methods are superior to the previously proposed state-of-the-art techniques and can be efficiently used for spam review detection.

Incorporating the time dimension into the representation generation framework can be considered as a future work. Besides, the use of deep neural network architectures can also be investigated for designing an end to end spam review detection framework which incorporates more powerful representations of spam reviews extracted from

raw review texts and underlying reviewer-product network structure into distinguishing of the spam reviews from the legitimate ones. Additionally, the effect of data and network size to the performance can be investigated with a different experiment setting on ZIP Dataset which is the biggest dataset that we used in our experiment in this work. It can be divided into the increasing size of bins. Reviews can be sampled randomly with different sizes, and the effect of increasing data size on the performance of NodeRep, DocRep and SPR2REP approaches can be analyzed.

## REFERENCES


[1] M. Team, "70% of Americans seek out opinions before purchasing", Mintel, (2018). [Online]. Available: http://www.mintel.com/press-centre/social-and-lifestyle/seven-in-10-americans-seek-out-opinions-before-making-purchases. [Accessed: 08- Apr- 2018].

[2] "Yelp", Yelp, (2018). [Online]. Available: https://www.yelp.com. [Accessed: 08- Apr- 2018].

[3] N. Jindal, B. Liu, Analyzing and detecting review spam, Seventh IEEE International Conference on Data Mining (ICDM 2007), IEEE, (2007), 547–552. doi: 10.1109/icdm.2007.68.

[4] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter, and H. Al Najada, Survey of review spam detection using machine learning techniques, Journal of Big Data 2 (1), (2015), 23. doi: 10.1186/s40537-015-0029-9.

[5] N. Jindal, B. Liu, Opinion spam and analysis, Proceedings of the 2008 International Conference on Web Search and Data Mining, ACM, (2008), 219–230. doi: 10.1145/1341531.1341560.

[6] F. Li, M. Huang, Y. Yang, and X. Zhu, Learning to identify review spam, IJCAI Proceedings International Joint Conference on Artificial Intelligence 22, (2011), 2488.

[7] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, 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 1, Association for Computational Linguistics, (2011), 309–319.

[8] S. Feng, R. Banerjee, and Y. Choi, Syntactic stylometry for deception detection, Proceedings of the 50th Annual Meeting of the Association for

Computational Linguistics: Short Papers 2, Association for Computational Linguistics, (2012), 171–175.

[9] G. Wang, S. Xie, B. Liu, and S. Y. Philip, Review graph based online store review spammer detection, 2011 IEEE 11th international conference on Data Mining, IEEE, (2011), 1242– 1247. doi: 10.1109/ICDM.2011.124.

[10] H. Li, A. Mukherjee, B. Liu, R. Kornfield, and S. Emery, Detecting campaign promoters on twitter using markov random fields, 2014 IEEE International Conference on Data Mining, IEEE, (2014), 290–299. doi: 10.1109/ICDM.2014.59.

[11] Q. Le, T. Mikolov, Distributed representations of sentences and documents, Proceedings of the 31st International Conference on Machine Learning (ICML-14), (2014), 1188– 1196.

[12] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, (2016), 855–864. doi: 10.1145/2939672.2939754.

[13] E.P. Lim, V.A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, Detecting product review spammers using rating behaviors, Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, (2010), 939–948. doi: 10.1145/1871437.1871557.

[14] KB. Sathees, RA Karthika, Survey on text mining process and techniques, International Journal of Advanced Research in Computer Engineering & Technology, (2014) Jul, 2279-2284.

[15] R.A. Saravanan, MR. Babu Text Mining Techniques and Its Applications: A Survey, International Journal of Computer Science and Technology. (2017) Sep, 33-35.

[16] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, A neural probabilistic language model, Journal of machine learning research 3 (Feb), 2003, 1137–1155.

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

[18] J. Pennington, R. Socher, and C. Manning, Glove: Global vectors for word representation, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), (2014), 1532–1543. doi: 10.3115/v1/D14-1162.

[19] M. Baroni, G. Dinu, and G. Kruszewski, Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors, ACL (1), (2014), 238–247. doi: 10.3115/v1/P14-1023.

[20] I.J. Unanue, E.Z. Borzeshi, and M. Piccardi, M, Recurrent neural networks with specialized word embeddings for health-domain named-entity recognition, Journal of biomedical informatics, 76, (2017) 102-109. doi: 10.1016/j.jbi.2017.11.007.

[21] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9(8), (1997), 1735-1780. doi: 10.1162/neco.1997.9.8.1735.

[22] C. dos Santos, M.Gatti, Deep convolutional neural networks for sentiment analysis of short texts, Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, (2014), 69-78. doi: 10.1109/ICCAR.2017.7942788.

[23] A. Bharati, K. V. C. Kamisetty, and R. S. S. Bendre, A document space model for automated text classification based on frequency distribution across categories, International Institute of Information Technology technical report, (2002).

[24] M. Gasparini, Community Analysis Using Graph Representation Learning on Social Networks, Politecnico do Milano, (2017).

[25] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, Advances in neural information processing systems, (2002), 585–591.

[26] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, science 290 (5500), (2000), 2323–2326. doi: 10.1126/science.290.5500.2323.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, Line: Large-scale information network embedding, Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, (2015), 1067–1077. doi: 10.1145/2736277.2741093.

[28] B. Perozzi, R. Al-Rfou, and S. Skiena, Deepwalk: Online learning of social representations, Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, (2014), 701–710. doi: 10.1145/2623330.2623732.

[29] E.P. Lim, V.A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, Detecting product review spammers using rating behaviors, Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, (2010), 939–948. doi: 10.1145/1871437.1871557.

[30] K.H. Yoo, U. Gretzel, Comparison of deceptive and truthful travel reviews, Information and communication technologies in tourism 2009, (2009), 37–47.

[31] A. Mukherjee, V. Venkataraman, B. Liu, and N. S. Glance, What yelp fake review filter might be doing?, ICWSM, (2013), 409-418.

[32] G. Fei, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh, Exploiting burstiness in re-views for review spammer detection, ICWSM 13, (2013), 175–184.

[33] L.C. Cheng, J. C. Tseng, and T.-Y. Chung, Case study of fake web reviews, Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, ACM, (2017), 706–709. doi: 10.1145/3110025.3110119.

[34] S. Y. Bhat, M. Abulaish, Community-based features for identifying spammers in online social networks, Advances in Social Networks Analysis

and Mining (ASONAM), 2013 IEEE/ACM International Conference, (2013), 100-107. doi: 10.1145/2492517.2492567.

[35] S. Rayana, L. Akoglu, Collective opinion spam detection: Bridging review networks and metadata, Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining, ACM, (2015), 985–994. doi: 10.1145/2783258.2783370.

[36] C. M. Yilmaz and A. O. Durahim, SPR2EP: A Semi-Supervised Spam Review Detection Framework, 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, (2018), 306-313. doi: 10.1109/ASONAM.2018.8508314.

[37] N. Torosyan, Application of binary logistic regression in credit scoring, University of Tartu, (2017).

[38] JR. Quinlan, Simplifying decision trees, International journal of man-machine studies. 1987 Sep 1;27(3):221-34.

[39] L. Breiman, J. H. Friedman, R. A. Olshen,, and C. J. Stone, Classification and Regression Trees, Chapman & Hall/CRC, (1984).

[40] XL. Li, editor, Biological data mining in protein interaction networks, Igi Global, (2009 May 31).

[41] Hastie, T., Tibshirani, R., Friedman, J., The elements of statistical learning: data mining, inference and prediction, Springer, (2009).

[42] R.John, L.,Townshend, Ensembling Models, CS229 Lecture notes, Standford University

[43] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, (1961).

[44] S. Renals, Multi-Layer Neural Networks, Teaching Courses, University of Edinburg, (2014).

[45] T. Fawcett T, An introduction to ROC analysis, Pattern recognition letters, (2006);27(8):861-74.

[46] M., Zhu, Recall, Precision and Average Precision, Teaching Courses, University of Waterloo, (2004).