IDENTIFYING RESTAURANTS PROPOSING NOVEL KINDS OF CUISINES: USING YELP REVIEWS

by

Haritha Akella

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2017

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Haritha Akella

Thesis Title: Identifying Restaurants Proposing Novel Kinds of Cuisines: Using Yelp Reviews

Date of Final Oral Examination: 13th October 2017

The following individuals read and discussed the thesis submitted by student Haritha Akella, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Edoardo Serra, Ph.D.	Chair, Supervisory Committee
Tim Andersen, Ph.D.	Member, Supervisory Committee
Casey Kennington, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Edoardo Serra, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

dedicated to my beloved family

ACKNOWLEDGMENTS

I would like to extend my gratitude to Dr.Edoardo Serra, my adviser who has been my inspiration all through the time. He has instilled in me an optimistic nature and helped me through the tough times and taught me to relentlessly work towards achieving the desired result. He has continuously bolstered my learning experience. He has motivated me to improve at every stage. He is enthusiastic about teaching new things and exploring novice ideas. He listens patiently to our questions and doubts and makes sure he answers them to our fullest satisfaction. If not for him, I would not have done my thesis in machine learning. He was my Data Science professor and his teaching methodologies inspired me to the core. I really cannot thank you enough for helping me find my cup of tea.

I would also like to thank my professors who have guided me throughout the course and helped me recognize and pursue my area of interest in machine learning and Data Science. I thank my committee members, Dr.Tim Andersen and Dr.Casey Kennington for their valuable feedback and support. Thank you so much.

My colleagues Mikel and Ashish have been a great support and were always there to extend a helping hand when needed. They were willing to share their knowledge and I would take this opportunity to thank them.

Importantly, I am grateful to my parents who have encouraged me to pursue higher studies and have supported me through all the tough times. I would like to express immense gratitude to my father, Mr.Bhaskar Rao, who has assured me that tough times are like passing clouds and I should be goal oriented and focused. He showed me a ray of hope in my difficult times. My mother, Mrs.Lakshmi, has been my role model since my childhood. She has been there to inspire me and she is my friend, philosopher and guide. Thank you both for being my supporting pillars. I would like to thank my sister Manasa and my in laws who taught me to be optimistic and left no stone unturned to help me in every possible way.

I would also like to thank Arjun, my 1 year old son, who has been a stress buster and relief icon to me and my husband, Kishore, who has gone above and beyond to support me through all the means he could and has given me a shoulder when needed and encouraged me to reach the pinnacle of success and motivated me the most. I would like to thank you! Thank you is a very small word compared to what you have done for me. You made me a proud wife and this thesis is dedicated to you both.

There are many people who have directly and indirectly helped me learn, grow and become capable of completing my master's degree in and outside the Boise State University. Finally, I would like to thank them for being so compassionate and believing in me.

Thank you all!

ABSTRACT

These days with TV-shows and starred chefs, new kinds of cuisines appear in the market. The main cuisines like French, Italian, Japanese, Chinese and Indian are always appreciated but they are no longer the most popular. The new trend is the fusion cuisine, which is obtained by combining different main cuisines. The opening of a new restaurant proposing new kinds of cuisine produces a lot of excitement in people. They feel the need to try it and be part of this new culture. Yelp is a platform which publishes crowd sourced reviews about different businesses, in particular, restaurants. For some restaurants in Yelp if the kind of cuisine is available, usually, there is a tag only for the main cuisines, but there is no information for the fusion cuisine. There is a need to develop a system which is able to identify restaurants proposing fusion cuisine (novel or unknown cuisines).

This proposal is to address the novelty detection task using Yelp reviews. The idea is that the semi-supervised Machine Learning models trained only on the reviews of restaurants proposing the main cuisine will be able to discriminate between restaurants providing the main cuisine and restaurants providing the novel ones.

We propose effective novelty detection approaches for the unknown cuisine type identification problem using Long Short Term Memory (LSTM), autoencoder and Term-Frequency and Inverse Document Frequency(). Our main idea is to obtain features from LSTM, autoencoder and TF-IDF and use these features with standard semi-supervised novelty detection algorithms like Gaussian Mixture Model, Isolation Forest and One-class Support Vector Machines (SVM) to identify the unknown cuisines.

We conducted extensive experiments that prove the effectiveness of our approaches. The score that we obtained has a very high discrimination power because the best value of AUROC for the novelty detection problem is 0.85 from LSTM. LSTM outperforms our baseline model of TF-IDF and the main motivation is due to its ability to retain only the useful parts of a sentence.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GPU – Graphics Processing Unit

BVLC – Berkeley Vision and Learning Center

VGG – Visual Geometry Group

LSTM – Long Short Term Memory

ILSVRC – ImageNet Large-Scale Visual Recognition Challenge

ReLU – Rectified Linear Units

CNN – Convolutional Neural Networks

OCR – Optimal character recognition

CRMKL – Convolutional recurrent multiple kernel learning

RNN – Recurrent neural network

RBM – Restricted Boltzmann machines

MIL – Multiple instance learning

YOLO – You Only Look Once

COCO – Common objects in context

FCLN – Fully Convolutional Localization Network

LDA – Latent Dirichlet Allocation

LSA – Latent Semantic Analysis

NNLM – Neural network language mode

RNNLM – Recurrent neural network language model

CHAPTER 1

INTRODUCTION

1.1 Overview

Yelp [75] offers users a myriad of reviews and ratings of businesses all over the world. In Yelp, users publish crowd-sourced reviews about local businesses, especially restaurants. When writing a review about a business, the users also have an option to upload photographs and write a text comment about the business.

The opening of a new restaurant proposing a new kind of cuisine produces a lot of excitement and people feel the need to try it and be part of this new culture. For some restaurants in Yelp if the kind of cuisine is available, usually the cuisine declared are the main ones, but there is no matching information for the fusion cuisine. The goal of the Yelp restaurant novelty detection problem is to build a model that identifies fusion restaurants.

We initially start by solving the pure classification problem of labeling restaurants with main cuisine types and then proceeding with the novelty detection task.

Classification problem of labeling restaurant with main cuisine types: The classification problem can apparently seem an easy task, mostly related to the keyword contained in the Yelp reviews, for instance, if the word ”Indian” appears in a review we can conclude that the kind of cuisine is Indian. However, because of its crowd-sourced nature Yelp contains a lot of misleading sentences. One example is,

”I usually prefer Japanese restaurant, but this Indian restaurant is excellent”. What is now the kind of cuisine provided by this restaurant? As humans, it is easy for us to answer this question, but for a machine this is a difficult task, requiring a deep understanding of the sentence to realize that Japan is not a relevant keyword in that context. In addition, the Yelp result is unbalanced since some cuisines are usually more popular than the other.

For the business label classification task from reviews, we compare two models. One is our baseline using Term-Frequency Inverse-Document-Frequency (TF-IDF) [74] and Random Forest (it is based on keywords) and the second is a Long Short Term Memory(LSTM) that is a recurrent neural network.

We also try to predict the label of main cuisines from restaurant images features obtained from pre-trained CNN. LSTM outperforms the base line approach and the main motivation is due to its ability to retain only the useful parts of a sentence. The details are discussed in detail in further chapters.

Main restaurant label classification:

1. Split all the restaurant businesses into train and test. Have the corresponding reviews and images also split based on their associated restaurants.

2. Use restaurant training reviews to train TF-IDF model and restaurant training images on pre-trained CNN model to extract features.

3. Train models like Random Forest and extra tree classifier using these extracted features.

4. Obtain features for the test set from previously trained TF-IDF/CNN models.

5. The trained models like Random Forest and extra tree classifier are finally evaluated with these test set features.

Novelty Detection problem of identifying fusion cuisines:

We formulate our problem as a novelty detection task to identify a fusion restaurant. Our proposal is to provide semi-supervised Machine Learning models trained only on the reviews of restaurants, proposing that the main cuisine will be able to discriminate between a restaurant providing the main cuisine and a restaurant providing the novel ones.

We compute TF-IDF on the reviews of the business providing main cuisines and then train novelty detection models such as Gaussian Mixture Models, One Class SVM, and Isolation Forest. The final result is a score which can discriminate between main cuisines and novel ones.

However, what we want is to use the deep understanding of the LSTM in this task. Then, we train the LSTM to classify only the known main cuisine. After that, we use the last state of the LSTM as the fixed feature vector representing the reviews of a business. This feature vector is again used with novelty detection models.

Novelty Detection:

1. Use main restaurant reviews to train TF-IDF/LSTM/autoencoder model and extract features.

2. Train novelty detection model as Gaussian Mixture Models, One Class SVM, or Isolation Forest using these extracted features.

3. These novelty detection algorithms are finally evaluated with the test set containing both main and fusion cuisines using the AUROC and Average Precision metric scores.

1.2 Contributions and Outline

The main contributions of this thesis are as follows:

1. We propose the new problem of identifying fusion types of restaurants.

2. We initially start tackling the problem of the restaurant label classification (classification task) from image and text features obtained from CNN and TF-IDF, LSTM models, respectively.

3. We discuss the various models and methods used and compare the efficiency of all the approaches including but not limited to TF-IDF, LSTM, Word2Vec, CNN on varied datasets.

4. Initial results help us conclude that text features from Yelp reviews performed better in the classification task than features from Yelp images. We performed 2 tests to evaluate the statement. We tested review features from TF-IDF and image features from the CNN model. Review features of TF-IDF were better in training the classifier compared to image features obtained from Alexnet CNN. Similar results were obtained when comparing the results of classification using LSTM review features and LSTM image features.

5. LSTM outperforms the base line approach(TF-IDF) in the classification task and the main motivation is due to its ability to retain only the useful parts of a sentence.

6. All the use cases of LSTM, Word2vec pre-trained model with fixed embedding, LSTM pre-trained Word2vec allowing the model to change the embedding and LSTM Word2Vec trained specifically on the dataset with fixed embedding and LSTM Word2vec specific training and letting the model change the embedding were tested. None of these use-cases were proven effective when compared to pure LSTM approach.

7. The basic model of LSTM with review features was proven effective when compared to any other model used in this thesis.

8. An effective and efficient solution to the problem of identifying an unknown types of cuisine is proposed by taking in input reviews representing several types of known cuisines. (The terms unknown cuisine, fusion cuisine and novel cuisine are used interchangeably, hereafter.)

9. LSTM and autoencoder models are trained to classify only the known main cuisines and use the last state of LSTM, and the middle state of autoencoder as fixed feature vectors which are used with novelty detection models like Gaussian Mixture Model (GMM), One Class SVM and Isolation Forest.

10. Auroc and average precision are used as performance metrics for the novelty detection task using GMM, One Class SVM and Isolation Forest. LSTM with GMM performed well compared to autoencoder with GMM or autoencoder reconstruction error with GMM. However, using autoencoder reconstruction error with GMM gave better results compared to results obtained from pure features of autoencoder.

In **Chapter 2**, we provide the relevant background details and literature review of previous research works related to our thesis, mostly about machine learning in supervised and unsupervised context, neural networks and transfer learning. We also discuss commonly used standard novelty detection algorithms, performance metrics and LSTM.

In **Chapter 3**, methodology and implementation of the classification task are discussed. We also present the novelty detection task in our problem domain and the methods adopted to determine novel cuisines.

In **Chapter 4** , discussion on the experimental setup is done. We then conclude the chapter with discussion and statistical relevance of the obtained results. Finally, we discuss the future research direction of our research work presented in this thesis in **Chapter 5**.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter provides the relevant background on Machine Learning, neural networks, LSTM, TF-IDF and novelty detection algorithms. For more details on machine learning and neural networks, we recommend readers to refer to a book by Goodfellow et al. [25] . For further details on LSTM and autoencoder refer to [27, 8], respectively. Similarly, for novelty detection algorithms, a survey paper by Pimentel et al. [53] is recommended.

2.1 Machine Learning

Machine learning is a form of applied statistics with emphasis on the use of computers to learn complex mathematical functions. Machine learning algorithms can learn from data (i.e observed facts). More formally, a machine learning process is defined as follows: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [34]. While solving machine learning problems, its very important to formulate what type of task (eg. classify business into multiple categories, machine translation and so on) we want our machine to learn while experiencing the available training data with improved performance (eg. accurately predict an image of pizza as pizza).

Machine learning algorithms can be broadly categorized into supervised and unsupervised learning algorithms depending upon the nature of the dataset they are learning from and the task they have to perform.

2.1.1 Unsupervised Learning

Unsupervised learning algorithms are types of machine learning algorithms that learn the important properties of the underlying training data being experienced without the supervisory labeled responses. Commonly applied unsupervised machine learning algorithms are clustering, anomaly detection and novelty detection algorithms.

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. By contrast with supervised learning, there are no explicit target outputs or environmental evaluations associated with each input; rather, the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output.

2.1.2 Supervised Learning

Supervised machine learning is a task of searching a function from a hypothesis space, given a training data with the corresponding labels. Each training example consists of an independent variable(s) defining the input domain of data and a dependent variable(s) defining the target. Supervised learning can be formulated as enabling a computer to learn a function f : X→y, where X represents the space of independent variables (input space), and y the space of dependent variables (output space). To tackle any machine learning algorithms, we can almost always start by making some general design choices beforehand. Note that machine learning algorithms can be as difficult as learning a very complex mathematical function, and most of the time the most scientific way to tackle such problems is through a process: experimentation, evaluation and decision making. Here are the general design choices for building machine learning algorithms:

1. Choose the form of the model i.e. hypothesis space for candidate models
2. Choose loss function.
3. Select optimization procedure.

We introduced a couple of important machine learning terminologies above. We discuss these design choices in the context of a simple machine learning algorithm, Linear Regression.

Linear Regression: Linear Regression is an algorithm that models the relationship between one or more independent variables X and a scalar and real-valued dependent variable y. In other words, Linear Regression learns a function mapping that takes each observation in X as input and tries to approximate as accurate y as possible. We call this process making a prediction over a range of continuous values. Along the same line, the classification task has a goal to learn a function that can achieve a prediction over a discrete class of objects.

In a real-world supervised learning setting, we measure the performance of machine learning algorithms by evaluating the learned function (or model) with previously unseen observations. These observations are commonly called test data and the process of computing output for test data is termed as inference or prediction. The ability of machine learning algorithms to perform well on tasks with the test data is known as algorithm's generalization power. How can a machine learning algorithm do well with the test data when observation is done only on the training data? This is a central idea of how machine learning algorithms work and can be reasoned well with the help of statistical learning. Machine learning algorithms are developed on top of a very important statistical assumption that basically says that the observations in training and test data are observed during data generating process as independent and identically distributed (i.i.d.) samples.

Model Selection Criterion. Cross-validation is one of the successful approaches in evaluating the performance of trained models in machine learning. Training a model with a fixed training data and evaluating the trained model with a fixed and usually very small test data can pose problems during generalization. This is because if the fixed small test data are used to validate the model and if the test data turns out not to represent the underlying true distribution of the data (regardless of the i.i.d. assumption, which is a possible case), then the trained model might not well generalize to other unseen data other than the ones used to validate the model. So to avoid such a scenario, k-fold cross-validation technique is used. Training data is divided into k equal and disjoint samples, and usually, k-1 folds are used to train the model and the remaining 1 fold is used to validate it. This process is repeated for k times selecting the unique validation set for each case. The final performance measure is then averaged over the k-cases hence bolstering the generalization ability of the trained models.

Optimization In machine learning, one of the most commonly used optimization algorithms is gradient descent. If we pose the restriction on selection of an objective function to be only differentiable functions, then we can compute the gradient of the function (using a numerical method or an analytic method using calculus).

Summary In general understanding, many of the concepts discussed with linear models like simple Linear Regression problems are generalizable to the complex machine learning algorithms like SVMs and deep neural networks as well. The design choices will vary depending on the nature and size of training data that the considered algorithms will experience and the machine learning task to be accomplished. With neural networks, there are further decisions to be made on what type of network architecture to choose. The in-depth discussion on these topics is outside the scope of this thesis.

2.1.3 Semi-supervised Learning

Semi-supervised learning (SSL) is half-way between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervision information but not necessarily for all examples. Often, this information standard setting will be the targets associated with some of the examples. Semi-supervised learning is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data. Traditionally, learning has been studied either in the unsupervised paradigm (e.g., clustering, outlier detection) where all the data are unlabeled, or in the supervised paradigm (e.g., classification, regression) where all the data are labeled. The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. Semi-supervised learning is of great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data are scarce or expensive. Semi-supervised learning also shows potential as a quantitative tool to understand human category learning, where most of the input is self-evidently unlabeled.

2.2 Novelty Detection

Novelty detection is a machine learning task of identifying new or unknown data during the inference or generalization phase (test data) that were not present in the training data during the learning phase. In simple terms, the machine learning algorithm is constructed to learn the characteristics of only the "known" training data. Novelty detection algorithm expects a mixture of known and novel examples in test data during the generalization phase. By novel or unknown examples, we mean that these novel examples in test data come from different probability distribution or lies far in the feature space than the training data. Novelty detection has important applications in domains involving large datasets generated from critical systems. Applications include cyber-intrusion detection, terrorist activity, system breakdown, fraud detection [52, 72, 11, 60, 56, 22, 76] data leakage prevention [67] and many other specialized applications [24].

Many novelty detection algorithms have been proposed in the last few decades. The main idea behind most novelty detection algorithms (esp. the ones used in this thesis) is to identify the strict regions where the normal instances (i.e not novel) lie in the feature space. The novelty score is given as a form of distance from these regions or sometimes as a probability that the example is not contained in any of these regions. The clustering based novelty detection techniques obtain their scores by using standard clustering algorithms such as K-Means [66] and Gaussian Mixture Models [3].

Just to build an intuition around novelty detection algorithms, very similar class of algorithms known as outlier detection algorithms also have a near-to-similar goal of identifying outlier or abnormal data points in the test data. The only difference is that the unknown or abnormal instances of data points are also considered during the training phase. Outliers in the dataset possess highly deviated statistical characteristics which are not in agreement with the majority of observations in the training data. Aiding the Yelp application to identify fusion cuisines via implementation of novelty detection algorithms is the primary goal of this thesis.

Novelty detection algorithms are commonly categorized into different categories:

a) probabilistic (eg. Gaussian Mixtures),

b) distance-based (eg. K-means clustering),

c) domain-based (eg. one-class support vector machines),

d) reconstruction-based (eg. neural networks, autoencoder) and so on

We apply Gaussian mixture models, one-class SVM and isolation Forest (based on an ensemble of trees) algorithms to apply novelty detection algorithm in our methodology. We use these algorithms to identify novel cuisine in the test data that are not present during the training phase. We will discuss in more details about the applications of these algorithms specific to our application in Chapter 3. Next, we discuss in detail about how each of these algorithms can be trained to learn the normal characteristics in training phase and how to evaluate each of these models given the previously unseen test data containing both known and novel instances of data.

2.2.1 Gaussian Mixture Models:

Gaussian mixture model is a non-Bayesian, parametric probability density based model and assumes that all the data points are generated from a mixture of a number of Gaussian distributions with unknown parameters. Gaussian Mixture model implements EM-algorithm for fitting the training data. Each sample is fitted to a probability specific to each cluster and a sample is assigned to a cluster for which it has a higher probability.

GMMs estimate the probability density of the target class (here the normal class), given by a training set, typically using fewer kernels than the number of patterns in the training set [14]. The parameters of the model may be estimated using maximum likelihood methods (via optimisation algorithms such as conjugate gradients or expectation-maximisation, EM) or via Bayesian methods (such as variational Bayes) [9]. Mixture models, however, can suffer from the requirement of large numbers of training examples to estimate model parameters. A further limitation of parametric techniques is that the chosen functional form of the data distribution may not be a good model of the distribution that generates the data. However, GMMs have been used and explored in many studies for novelty detection.

2.2.2 One-class SVM:

One-class SVM is an unsupervised algorithm that learns a decision function for outlier detection which classifies new data as similar or different from the samples used in the training set. Given a set of training samples, it will develop a soft boundary defining the known samples and classifies new observations as belonging to the known samples or not.

Gardner et al. [23] apply a One-Class SVM to the detection of seizures in patients. The intracranial EEG time-series is mapped into corresponding sequences of novelty scores by classifying short-time, energy-based statistics computed from one-second windows of data. The model is trained using epochs of normal EEG.

Epochs containing seizure activity exhibit changes in the distribution in feature space that increases the empirical outlier fraction, allowing seizure events to be detected. A different approach for online novelty detection in temporal sequences is presented in [43]. To perform novelty detection, the authors define a matching function to determine how well a test sequence matches the normal model. Clifton et al. [17, 16] investigate the use of a one-class SVM using multivariate combustion data for the prediction of combustion instability. Wavelet analysis is used for feature extraction, from which detail coefficients are used as two-dimensional features. Novelty scores computed using the one-class SVM approach are obtained from each of the input time-series, and different classifier combination strategies are studied. The One-Class SVM has been used for novelty detection in functional magnetic resonance imaging data [26]; audio recordings [57]; text data [82]; medical data to identify patient deterioration in vital signs [15].

2.2.3 Isolation Forest:

Isolation Forest is a model-based method that isolates unknown data points (in context of novelty detection) or anomalies (in context of anomaly detection) instead of learning profiles of the normal training data. Isolation Forest is an ensemble anomaly detection algorithm that computes anomaly score for each sample. This algorithm works by isolating observations by first randomly selecting a feature and then randomly selecting a split from a range of minimum and a maximum value of the selected feature. The average depth of a tree required to isolate a sample over a Forest gives the measure of normality or abnormality of that score in a distribution. Fundamentally, according to isolation Forest, isolating unknown data points is easier as only a few splits (fewer conditions to be checked and hence lower average tree depth) are required to distinguish the unknown data points from the known ones. In our application to identify unknown cuisines, we compute the average anomaly score of examples in test data computed from multiple random trees (base classifiers). The lower is the score for a review, the more the data point is unknown.

2.2.4 Autoencoder:

An autoencoder is an artificial neural network which is commonly applied for unsupervised novelty detection based on the reconstruction error of the training examples and nonlinear dimensionality reduction [62]. A simple autoencoder can be seen very similar to a feed-forward multilayer perceptron neural network. An autoencoder has two architectural parts: encoder and decoder. The encoder part of an autoencoder aims to learn an encoded representation (embeddings) of training in different feature space data by efficiently reducing the dimensionality of the original data space. The decoder phase, on the other hand, tries to reconstruct the original data by taking the embeddings (compressed feature vectors) as input. The output of an autoencoder has the same number of computational units as the input (original feature dimension).

The intuition behind an autoencoder to be used as novelty detection algorithm comes from the ability of autoencoder to effectively reconstruct the examples that have the similar statistical properties in the original feature space, thus obtaining the smaller reconstruction errors for the known type of objects. An autoencoder tends to obtain higher reconstruction errors for the novel or unknown cuisines. Similar to any novelty detection algorithm, autoencoder is also trained with only the known examples of training data. Once the optimized embeddings are learned using the training data, the reconstruction errors are computed for all the known and novel data in the test set. The higher the reconstruction error, the higher is the chance of that data point to be unknown.

Figure 2.1 shows an architecture of a simple 3-layered feed-forward autoencoder. [45] presents a novel unsupervised approach based on a denoising autoencoder. In

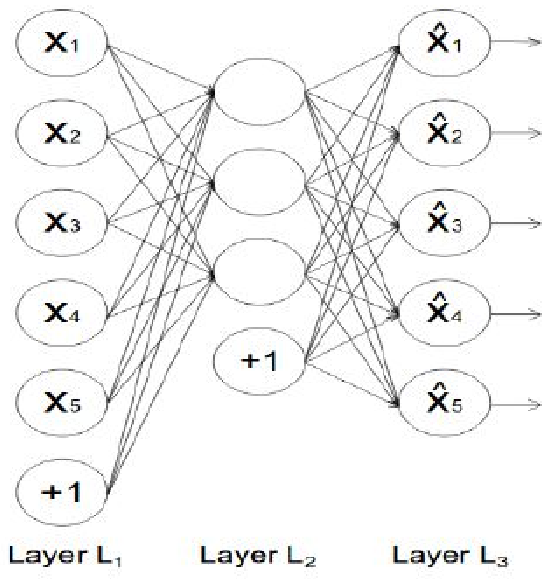


Figure 2.1: Simple architecture of an autoencoder with an input (Layer L1), one hidden (Layer L2) and an output layer (Layer L3). Layer L2 captures the embeddings of input layer into lower dimensional space. These embeddings are used by output layer to reconstruct original data. Source: [62]

their approach auditory spectral features are processed by a denoising autoencoder with bidirectional Long Short-Term Memory recurrent neural networks. They used the reconstruction error between the input and the output of the autoencoder as an activation signal to detect novel events. The autoencoder is trained on a public database which contains recordings of typical in-home situations such as talking, watching television, playing and eating.

In this thesis, we mention autoencoder at the level of feature extraction. We extract the features from the middle layer, which captures the embedding of input layer into lower dimensional space and uses these features in novelty detection of unknown cuisines.

2.3 TF-IDF

Term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

In a large text corpus, some words will be very present (e.g. the, a, is in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms. What this ensures is that if a token occurs frequently in a document that token will have high TF but if that token occurs frequently in a majority of documents then it reduces the IDF. So stop words like an, the, i which occur frequently are penalized and important words which contain the essence of the document get a boost. Both these TF and IDF matrices for a particular document are multiplied and normalized to form TF-IDF of a document. In order to re-weight the count features into floating point values suitable for usage by a classifier, it is very common to use the TF-IDF transform. Tf means term-frequency while TF-IDF means Term-Frequency times Inverse Document-Frequency:

TF-IDF(t,d)= $tf(t,d) \times$ **idf(t)**

We will give a quick informal explanation of TF-IDF before proceeding. Essentially, TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TF-IDF numbers than common words such as articles and prepositions. Let us discuss another interesting algorithm Doc2Vec. Doc2vec is an entirely different algorithm from TF-IDF which uses a 3-layered shallow deep neural network to gauge the context of the document and relate similar context phrases together.

Important note about doc2vec is that it is not a monolithic algorithm like the bag of words. It has two different variations SKIP GRAM and CBOW, also it is used with other variations like with or without negative sampling and with or without hierarchical softmax. Lastly, doc2vec should be trained on a big enough and quality dataset for the model to generate sensible embeddings which will lead to good feature generation. Thus, TF-IDF essentially stresses the similarity of two documents if they're composed of the same words and as mentioned before, it doesn't take into consideration the semantic links between the words. There have been studies on the use of dimensionality reduction techniques such as Probabilistic Latent Semantic (Probabilistic LSA) Analysis [28], which seeks a k-generative model for word occurrences in a document. This method essentially tries to replace the vector-space model with a latent-space model.

2.4 LSTM

Humans do not start their thinking from scratch every second. As the reader reads this thesis, they understand each word based on their understanding of previous words. They do not throw everything away and start thinking from scratch again. Thoughts have persistence. Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist. This chain-like nature reveals that recurrent neural networks are able to process and represent sequences. They are the natural architecture of neural network to use for such data. In the last few years, there has been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning. Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture that remembers values over arbitrary intervals. Stored values are not modified as learning proceeds. RNNs allow forward and backward connections between neurons.

LSTM networks have been demonstrated to be particularly useful for learning sequences containing longer-term patterns of unknown length due to their ability to maintain long term memory. LSTM has been used in various fields for varied tasks in the past few years. One such work is with anomaly detection as in [44] where they stacked LSTM networks for anomaly detection. Acoustic novelty detection aims at identifying abnormal/novel acoustic signals which differ from the reference/normal data that the system was trained with.

An LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models (CRFs) and other sequence learning methods in numerous applications.

An LSTM network contains LSTM units instead of, or in addition to, other network units. An LSTM unit remembers values for either long or short time periods. The key to this ability is that it uses no activation function within its recurrent components. Thus, the stored value is not iteratively modified and the gradient does not tend to vanish when trained with backpropagation through time.

LSTM units are often implemented in "blocks" containing several units. This design is typical with deep neural networks and facilitates implementations with parallel hardware.

LSTM blocks contain three or four "gates" that control information flow. These gates are implemented using the logistic function to compute a value between 0 and 1. Multiplication is applied to this value to partially allow or deny information to flow into or out of the memory. For example, an "input" gate controls the extent to which a new value flows into the memory. A "forget" gate controls the extent to which a value remains in memory. An "output" gate controls the extent to which the value in memory is used to compute the output activation of the block. (In some implementations, the input and forget gates are merged into a single gate. The motivation for combining them is that the time to forget is when a new value worth remembering becomes available.) The weights in an LSTM block W and U are used to direct the operation of the gates. These weights are applied to the values that feed into the block (including the input vector xt and the output from the previous time at step ht-1 at each of the gates. Thus, the LSTM block determines how to maintain its memory as a function of those values, and training its weights causes the block to learn the function that minimizes loss.

Let us understand it in detail with examples from [1]. Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in the phrase: *" the clouds are in the ———–"* , we dont need any further context its pretty obvious the next word is going to be

[1]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

the sky. In such cases, where the gap between the relevant information and the place that its needed is small, RNNs can learn to use the past information.

But there are also cases where we need more context. Consider trying to predict the last word in the phrase: *" I grew up in France, I speak fluent ———-"*. Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. Its entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information. In theory, RNNs are absolutely capable of handling such long-term dependencies. A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs dont seem to be able to learn them. LSTMs dont have this problem!

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. All recurrent neural networks have the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer as shown in Figure 2.2

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way as in Figure 2.3.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It is very easy for information

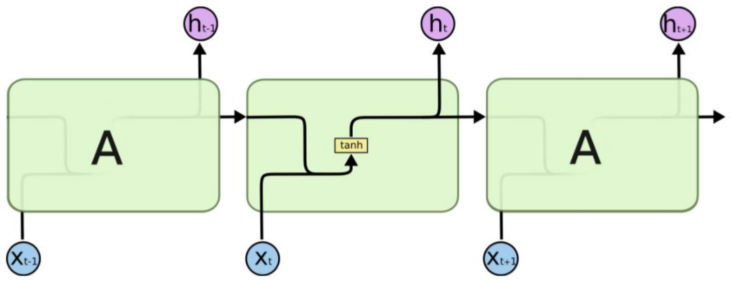


Figure 2.2: The repeating module in a standard RNN contains a single layer. Source [18]

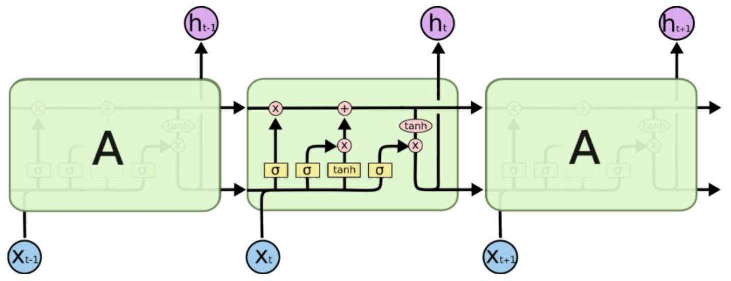


Figure 2.3: The repeating module in an LSTM contains four interacting layers. Source [18]

to just flow without changing. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means let nothing through, while a value of one means let everything through! An LSTM has three of these gates, to protect and control the cell state.

The first step in our LSTM is to decide what information we are going to throw away from the cell state. This decision is made by a sigmoid layer called the forget gate layer. It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents completely keep this while a 0 represents completely get rid of this.

The next step is to decide what new information we are going to store in the cell state. This has two parts. First, a sigmoid layer called the input gate layer decides which values we will update. Next, a tanh layer creates a vector of new candidate values, C_t, that could be added to the state. In the next step, we will combine these two to create an update to the state.

It is now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it. We multiply the old state by ft, forgetting the things we decided to forget earlier. Then we add $i_t \times C_t$. This is the new candidate values, scaled by how much we decided to update each state value.

Finally, we need to decide what we are going to output. This output will be based on our cell state but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we are going to output. Then, we put the cell state through tanh (to push the values to be between –1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

LSTMs were a big step in what we can accomplish with RNNs

2.5 word2vec

Understanding the labels and the semantic relationship between them is important. Investigating on word2vec [59] could be a possible exploration in this regard. It provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. This helps understand the closeness of similar words. Many different types of models were proposed for estimating continuous representations of words, including the well-known LSA and LDA. [46] uses NNLM and RNNLM architectures and focused on distributed representations of words learned by neural networks, as it was previously shown that they perform significantly better than LSA for preserving linear regularities among words [47]; We can use pre-trained word2vec models or can train our own model. We have tried both the scenarios in this thesis.

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. CBOW is faster while skip-gram is slower but does a better job for infrequent words.

Doc2vec Numeric representation of text documents is a challenging task in machine learning. Such a representation may be used for many purposes, for example, document retrieval, web search, spam filtering, topic modeling etc. The main purpose of Doc2Vec is associating arbitrary documents with labels, so labels are required. Doc2vec is an extension of word2vec that learns to correlate labels and words, rather than words with other words. The first step is coming up with a vector that represents the meaning of a document, which can then be used as input to a supervised machine learning algorithm to associate documents with labels.

2.6 CNN and transfer learning

L. Cavigelli et al. extended the image classification task employing CNN to make use of higher-dimensional multispectral input images [13]. K. Nogueira et al. presented an analysis and extensive experimental evaluation of three possible ways of using the existing CNN: full training, fine tuning and using models as feature extractors [48]. They discuss the overhead (time and space) that researchers have to face while fully training the CNN for the new dataset. They showed that fine-tuning specific layers of CNN properly applied based on the nature of dataset being experienced produce better models that can generate sufficiently representative features. This approach saves researchers from spending a lot of time fully training the models for each domain-specific dataset they might experience in their research. Similar to this work, W. Zhou et al. [64] also evaluated pre-trained and fine-tuned CNN architectures on a public scene dataset to investigate whether these models are able to learn dataset-specific features or not.

One of the core problems is obtaining labels for restaurant businesses based on photo features. Our plan involves a Convolutional Neural Network (CNN) in the form of AlexNet pre-trained on ImageNet dataset. The state of the art in CNN gets pushed ever so slightly with each passing year with more complicated and computationally demanding models. As of now, the leading model for the ImageNet Challenge is the Inception-v4 model architecture that was able to achieve 3.08% top error on the ImageNet challenge by using 75 trainable layers [69], beating out the ResNet and GoogleNet that were the previous reigning champs in image classification. CNN to tackle the ImageNet challenge in 2012 with the AlexNet architecture greatly improved accuracy over existing methods.

Within the field of Convolutional Neural Networks, there has been a lot of work done with Transfer Learning. Transfer learning refers to the process of taking a pre-trained CNN, replacing the fully-connected layers (and potentially the last convolutional layer), and training those layers on the pertinent dataset. We perform transfer learning using the dataset provided by Yelp of its various businesses.

AlexNet: It has to be noted that AlexNet was not trained on Yelp images for ImageNet. AlexNet. Krizhevsky et al. [25] won the annual Imagenet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012 with their breakthrough ConvNet model popularly known as AlexNet. This architecture constitutes eight layers network (five convolutional and three fully-connected). Each convolutional layer is scanned by a various number of kernels (each specifically responsible to find specific feature in the image), and the network tends to learn from general to more specific features as it goes into the deeper layers. Inspired by the working procedure of human visual cortex, the size of the kernels signify the size of the receptive field in the input image to be considered for learning feature specific to that particular area. Similar to the standard neural networks, units in the fully-connected layers are connected to all the units in the preceding layer. To add non-linearity to the network, this architecture uses Rectified Linear Units (ReLU)[35] as activation function after every convolutional and fully-connected layer. This network adopts gradient descent to optimize weights in the network. ReLU adds non-saturating non-linearity in the network which is much faster than the other activation functions like tanh and sigmoid while applying gradient descent. The local response normalization used in several layers implements a form of lateral inhibition that creates competition amongst neuron outputs for important activities. This basically aids generalization on top of the non-saturating ReLUs. The pooling layers provide ways to select outputs of the neighboring neuron groups mapped by the same kernels and use a single value for a group. Alexnet uses overlapping MAX-pooling approach to select a much-represented value from neighborhood pixels. To avoid overfitting, AlexNet used various data augmentation techniques to feed the network with varieties of transformed input images by enlarging the dataset using label-preserving transformations. The dropout layers reduce complex co-adaptations of neurons, thus strengthening the representation of such neurons in the network by randomly dropping out some other neurons which are deprived to contribute in the forward pass and back-propagation. Meaning that, for each sample, the network randomly selects a weight shared sub-networks to boost the contribution of selected subset of neurons thus preventing overfitting and enhancing the network's generalization power. Figure 2.4 (top) shows the architecture of 7-layered Alexnet.

Deep learning methods are representation learning approaches that can pick up the large dataset (images, audio, video etc.) in their raw form and can build hierarchical feature representation of dataset using the multi-layered deep network architectures [36]. Deep Neural Networks have recently been widely used in the field of computer vision in a range of visual recognition tasks, trained through layers of deep convolutional networks and back-propagation algorithm [37, 36]. In the last 5-6 years, due

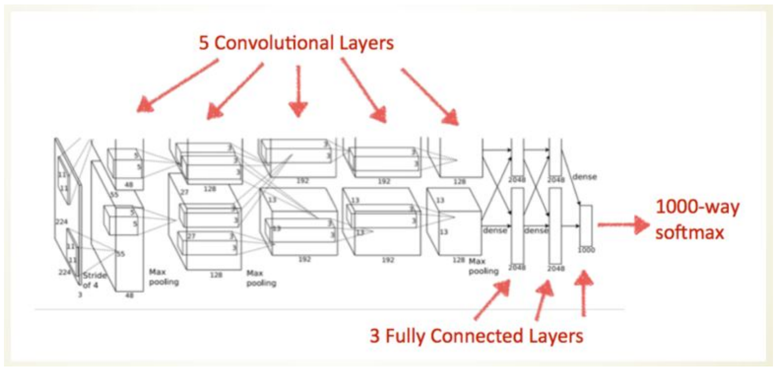


Figure 2.4: CNN architectures: Convolutional neural network architecture of Alexnet model with 7 learnable layers (top) and convolutional neural network architecture of VGGNet with 19 layers (including all convolutional and pooling layers) (bottom). Source: [31]

to the upsurge of high-performance computing devices such as GPUs and large-scale distributed systems and availability of large-scale public image repositories such as ImageNet, CNN have gained a great success range of visual recognition tasks such as image classification [35, 48], object localization and detection [81, 58, 39], image segmentation [41, 50, 79] and anomaly detection [61].

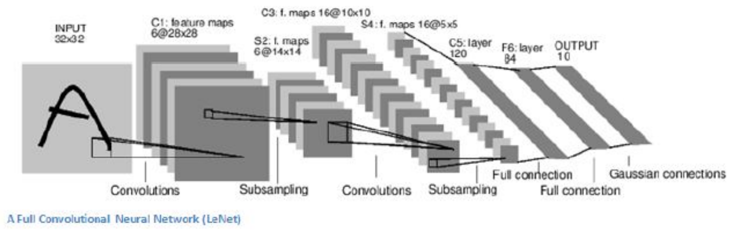


Figure 2.5: LeNet convolutional neural network architecture Source: [38]

Figure 2.5 [38] shows a simple fully convolutional neural network that starts with an input layer (the raw pixels are attuned to be represented as vectors) which is passed to a first convolutional layer. Convolutional layers contain filters or kernels.

Kernels can be interpreted as feature analyzers. In this case, there are 6 kernels that will produce 6 feature maps after convoluting the input image. Convoluting an image involves following operations: a) scanning an image from left-top to the bottom-right, b) performing an element-wise dot-product between the kernel and that specific part of an image currently being convoluted (also known as receptive fields), c) continuing convolution with a stride size that basically defines how many pixels should the kernels move to the right side. Kernels are generally smaller in size and are represented as squared matrix, and are shared among all the input examples. Next, the sub-sampling or pooling layer is applied to the activation map resulted from each kernel to reduce the size of the operational matrix yet preserving the most information. The commonly used pooling functions are max pooling; average pooling; etc. The output of the preceding layers are fed as input to the next layer, and the output is traversed until the output a layer of the network. For example in classification problem, result of the output layer is commonly termed as logits and the operational network layers such as softmax is applied to convert logits into a probability distribution representing each number to be the probability of that input belonging to particular category.

Deep Neural Networks (DNNs), esp. CNN have recently gained great success in lots of large-scale visual recognition tasks [35, 68, 71]. This has become possible because of publicly available large-scale image databases such as ImageNet (more than 14 millions of natural images from 20000+ different categories) [20] and powerful computing infrastructures such as GPUs and multi-node distributed clusters. In the past, ImageNet Large Scale Visual Recognition Competition (ILSVRC) has been challenging researchers around the world to develop effective and efficient visual recognition algorithms which led to the invention of deep CNN like Alexnet [35], VGGNet [68] and GoogLeNet [71]. These CNN achieved high and increasingly better accuracy in lots of visual recognition tasks such as image classification [48], object detection [58] and image segmentation [41]. Deep CNN take weeks to train on large datasets like ImageNet even with powerful GPUs. In real-world scenario, finding a sufficiently large amount of application-specific dataset for training CNN models and conducting such many experiments with multiple hyper-parameters set as short period of time is ideal. However, it is very common to use pre-trained CNN models like Alexnet, VGGNet and GoogLeNet already trained on large datasets such as ImageNet and use these trained weights for training these CNN on new and relatively smaller datasets. This particular way of transferring general feature representations from one domain (or dataset) to another is known as transfer learning. In the transfer learning context, these pre-trained models can be used either as weight initialization or as feature extractors for new datasets. Using pre-trained models as weight initialization to train the CNN in another domain or with other datasets is generally termed as a fine-tuning approach. Deep CNN tend to learn general features (eg. edges and color blobs in images) on the lower level layers and more abstract and dataset-specific features on the higher level layers of CNN [77]. So CNN can be efficiently trained on new datasets by using pre-trained models as weight initialization, freezing the learning rates of lower level layers and updating weights only for higher level layers (generally last one or two fully connected layers). This allows the newly trained CNN to adopt feature representations already learned from source dataset and apply that knowledge to new datasets. This approach keeps researchers from having to train these complex CNN from scratch thus saving a lot of computation time. CNN transfer learning approach applied for many classification tasks has obtained accuracy scores greater than 90% for smaller training dataset and training time [48] .

In this thesis, we use CNN transfer learning to address the problem of the classification task. We formulate our problem as label classification task to identify labels based on user-uploaded images. We use the CNN transfer learning approach to train CNN on Yelp image dataset. We use these trained models to extract features from all the images. Features extracted from these images are used to train classifiers which are finally evaluated with the test set of image features.

Traditional machine learning techniques have the major assumption that the training and the testing data must have the same feature space and should come from a similar (if not the same) probability distribution family [49]. In many fields of study, even today, it is very common to learn different new models (supervised, semi-supervised and unsupervised) even for closely related domains and problems. The common approach of building algorithms with a fixed-variated dataset as input after initializing a new model with zero knowledge and training those models with the popular optimization techniques does not allow the algorithm developers and practitioners to transfer prior knowledge to the related problems in the similar problem domain. In recent years, transfer learning has emerged as one of the effective techniques in machine learning and data mining as a way to transfer learned knowledge from one domain task to another in order to make the generalizability of the learned models more robust. The investigation on transfer learning was motivated by the fact that humans intelligently use knowledge learned from one task and apply it to the tasks in other similar domains that allow humans to learn fast and efficiently. In the field of machine learning, transfer learning was fundamentally motivated from the NIPS-95 workshop on "Learning to learn," that was focused on the need of long-running machine learning algorithms that can reuse the knowledge learned previously. Learning to learn, life-long learning, knowledge transfer, inductive transfer, multi-task learning, knowledge consolidation, context-sensitive learning, knowledge-based inductive bias, and so on are the alternative terms used for the transfer learning.

The most state-of-the-art approach that could be applied to understanding several objects in the Yelp data is the use of faster-rCNN region proposals. This technique uses a neural net to learn region proposals and then detect objects according to those regions. This proves exceptionally useful for understanding multiple objects in the same image and keeping track of where they occur spatially. This method has yielded a mean average precision of 58.85 percent on the ImageNet object detection challenge [35]. Another technique like YOLO, which formulates object detection as a regression problem, is also capable of tackling object detection even though in practice it performs slightly worse than faster-rCNN [78]. Another paper [40] talks about Microsoft COCO which is yet another object recognition and labeling semantic objects technique. However, training these architectures requires very specifically labeled datasets that have every object of interest labeled. Leveraging these models would require us to do additional separate labeling on the Yelp data to make these effective.

[69] Introduces a custom use of MIL to predict the attributes of restaurants based on images by performing transfer learning. MIL has proven very successful in the past. For example, in [6], MIL is used for solving a Visual Tracking problem. They presented a novel algorithm, in addition to a novel loss function, that they used to train their models to perform online MIL classification. The loss function itself has crafted artfully, as it took into account the gradient boosting framework detailed in [78], to maximize the log-likelihood function across several bags of instances. Another notable paper that detailed MIL is [30], specifically in its formulation of the loss function.

Multiple kernel learning (MKL) is a feature selection method where features are organized into groups and each group has its own kernel function [70]. In [29] the authors generate user-defined feature-based summaries, which is not scalable in large datasets. This thesis utilizes the models mentioned in [55] to combine features from both images and text. Traditional methods could only classify texts into different topics irrespective of user interests. More recent research in textual recognition has been conducted at either feature level [19] or decision level [65]. Only a few research works have been carried out on multimodal emotion recognition or sentiment analysis using textual clues. They have used MKL to fuse the 2 modalities, text and video. While the state of the art [51] uses a single kernel Support Vector Machine (SVM) classifier to fuse all three modalities, [55] uses multiple kernels to adapt to different modalities. They use the CRMKL model, which combines sentiment features in audio, video and text and, hence, achieve higher accuracy. They proposed a deep neural network that can be viewed as a composite of simple, unsupervised models such as RBM where each hidden layer serves as the visible layer for the next RBM. They have used CNN to extract textual features since, the CNN sentence model uses convolution as an operator to combine semantically-related word vectors and the convolution layers extract features in a hierarchical manner. Each RBM layer is trained in an unsupervised manner and then the complete deep model can be fine-tuned using a subset of the dataset with known labels. The features learned in an unsupervised manner in each layer may not be the best for classification but can be used to train state-of-the-art classifiers such as SVM or Naive Bayes. Depending on the length of a sentence, the higher-order features can be short and focused or long and global, spanning the entire sentence. CNN form local features for each word and combine them to produce a global feature vector for the whole text using several hidden layers. They integrate RNN with CNN and MKL to create the CRMKL model and extract features from audio, video and text and combine them using MKL. This could be adapted to the current project to fuse text and image modalities.

2.7 Review feature extraction

Feature extraction involves pre-processing, feature selection and feature cleansing as explained in [5]. Pre-processing of data includes Parts of speech or POS tagging, which is a linguistic technique used since 1960. POS tagging [4] assigns a tag to each word in a text and classifies a word to a specific category such as noun, verb, adjective, etc. POS taggers are efficient for explicit feature extraction in terms of the accuracy they achieve. Stemming and Lemmatization are two essential morphological processes of preprocessing modules during feature extraction [33]. The stemming process converts all the inflected words present in the text into a root form called as stem [4]. The lemma of a word includes its base form plus inflected forms [63]. Lemmatization groups together various inflected forms of a word into a single one [5]. Stemming removes word inflections only, whereas Lemmatization replaces words with their base form, hence lemmatization is considered to be more accurate. Stop word concept in pre-processing was first introduced by Hans Luhn, H.P [42]. Stop words are common and high-frequency words. Different methods are available for stop-word elimination [4]; ultimately they enhance the performance of the feature extraction algorithm [4]. The stop words removal reduces the dimensionality of the data sets and thus key words left in the review corpus can be identified more easily by the automatic feature extraction techniques. Feature selection methods are grouped into four main categories: NLP or heuristic-based, Statistical, Clustering based and Hybrid. NLP based techniques mainly operate on three basic principles:(a) Noun, noun phrases, adjectives, adverbs usually express features [33].(b)Terms occurring near subjective expressions can act as features [32]. (c) P is product and F is a feature in phrases like F of P or P has F [54]. They have got high accuracy, but low recall, with dependency on the accuracy of the part of speech being tagged. Clustering or Machine Learning based feature extraction techniques are implemented by [5], requiring few parameters to tune [21]. The key weakness of clustering is that only major features can be extracted and it is difficult to extract minor features [80]. Statistical techniques are univariate, multivariate and hybrid [1, 5, 2]. Univariate methods, also called feature filtering methods, take attributes separately; examples of this type include information gain (IG), chisquare, occurrence frequency, log likely-hood and minimum frequency thresholds. Univariate techniques have computational efficiency, but they ignore attribute interactions. Decision tree models, recursive feature elimination and genetic algorithms are the examples of multivariate methods, which considers a group of attributes and uses the wrapper model for attribute selection [2]. Hybrid techniques combine univariate, multivariate and other methods for achieving accuracy and efficiency [1]. This thesis uses NLP based feature selection techniques for reasons explained in [5].

2.8 Imbalanced data and methods to combat the issue

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. This is the case with our restaurants dataset. So to combat the issue we came up with multiple alternatives:

1. Penalizing Model: Penalized classification imposes an additional cost on the model for making classification mistakes on the minority class during training. These penalties can correct the model to pay more attention to the minority class.

2. Trying different algorithms that work best with imbalanced data.

3. Generating synthetic samples.

4. Resampling dataset: Either undersampling by deleting instances from the over-represented class or by adding copies of instances from the under-represented class called over-sampling (or more formally, sampling with replacement).

5. Changing performance metrics.

6. Collecting more data if possible.

The penalizing model worked well in our thesis.

2.9 Performance Metrics

Similar to any other machine learning algorithms, it is important that we evaluate the performance of our LSTM and novelty detection algorithms specific to our unknown fusion cuisine identification tasks. In our experiments, we perform five-fold cross-validation to evaluate our trained models. To validate novelty detection algorithms, we compute the novelty score for the test data after training various novelty detection algorithms. We use these scores and the ground-truth (whether the unknown cuisine in our case is known or novel) in order to compute Area under Receiver Operating Characteristics (AUROC) and Average Precision scores.

2.9.1 AUROC

AUROC scores measure the discriminating ability of classifiers or novelty detection algorithms to correctly classify objects in different categories: known category and novel category in novelty detection. AUROC score basically tells how well the novelty detection algorithm is able to distinguish known objects as known and novel objects as novel in the test dataset. The AUROC is a plot with a false positive rate of a discriminating model as the x-axis and true positive rate in the y-axis.

2.9.2 Average Precision

While AUROC score gives us the idea about how well the novelty detection algorithm discriminated the known and novel examples, Average Precision score tells us the ability of novelty detection algorithm to discriminate novel objects as the novel ones. In a real-world scenario, it is common to have a very small proportion of novel examples compared to the normal examples. Therefore, we want to make sure that our novelty detection algorithms are able to discriminate the relatively small population of novel examples. False negative (eg. an unknown or fusion cuisine classified as known main cuisine when it is unknown) rate is very critical for the discriminating models like novelty detection and we would want to lower the false negative rates as much as possible.

In summary, in this chapter, we discussed the technical background and literature review on various machine learning based and other related topics relevant to the methodology and experimental design of our thesis work. In the next chapters, we present the methodology and implementation of these concepts in our thesis.

CHAPTER 3

METHODOLOGY

In this chapter, we discuss in detail the algorithmic implementation of convolutional neural networks, TF-IDF, LSTM, word2vec and novelty detection algorithms in the supervised and unsupervised learning contexts. Just to recap the goal of our thesis, following is the problem definition.

Problem Definition: We define the problem of identifying novel cuisines. Given the multiple types of cuisine reviews as training experience to the novelty detection algorithms, we want to investigate how well our novelty detection algorithms perform in detecting unknown cuisines, given the combination of known and unknown reviews (test data) not seen during the training session. Thus we say that our applied machine learning algorithms experience reviews in both training and test sessions and evaluate the performance in terms of novelty score with test data. To achieve the above-mentioned goal of building novelty detection machine learning techniques for our problem domain, we initially start by solving the pure classification problem of labeling the restaurants with main cuisine types. Then we present a more sophisticated way of using LSTM to extract representational features of text reviews before applying standard novelty detection algorithms.

3.1 Restaurant label classification

We compare two models for predicting labels based on text features: one is our base line using Term-Frequency Inverse-Document-Frequency (TF-IDF) and Random Forest (it is based on keywords) and the second is a Long Short Term Memory that is a recursive neural network.

We also try to predict the label based on image features obtained from pre-trained CNN.

Before we get into further details, let us understand a few concerns and the reason that it is suggestible to consider a combination of image and text features for the classification task.

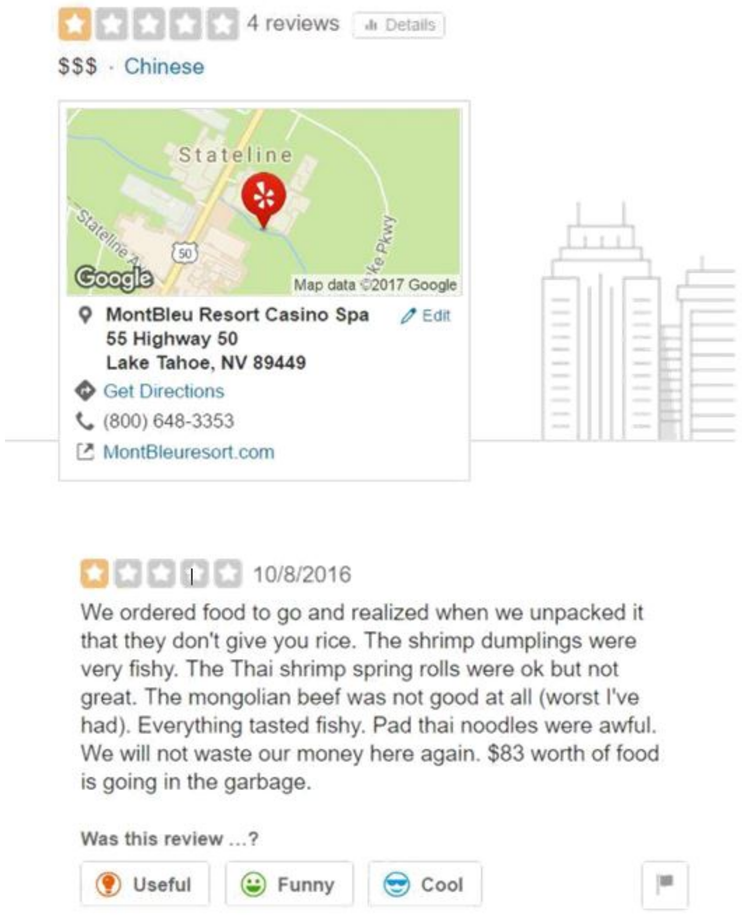
Given a photograph and review for a business, the labels associated with it are ambiguous or sometimes absent, as shown in Figure 3.1 .

There are also cases, for example, where users only tag the business as restaurant and do not mention or wrongly mention the sub-category as shown in Figure 3.2 .

The Yelp dataset has many images and reviews associated with each business. The problem is with the labels associated with each business which are either missing, ambiguous or incorrect. For example, a business could be labeled as a restaurant. But whether it is Mexican cuisine, Thai cuisine or Chinese cuisine is not mentioned sometimes. There are also many photographs that do not reveal much about the actual business as shown in Figure 3.3, which is the image of a person and not food.

The features obtained from such images, do not really help in labeling the restaurant business. The review is related to the restaurant business. Sometimes it is not a question of just the key word, but all the related features of the text.

Also, there are cases where a single business is categorized into multiple businesses


4 reviews
Details
$$$ · Chinese

MontBleu Resort Casino Spa
55 Highway 50
Lake Tahoe, NV 89449
Edit
Get Directions
(800) 648-3353
MontBleuresort.com

10/8/2016
We ordered food to go and realized when we unpacked it that they don't give you rice. The shrimp dumplings were very fishy. The Thai shrimp spring rolls were ok but not great. The mongolian beef was not good at all (worst I've had). Everything tasted fishy. Pad thai noodles were awful. We will not waste our money here again. $83 worth of food is going in the garbage.

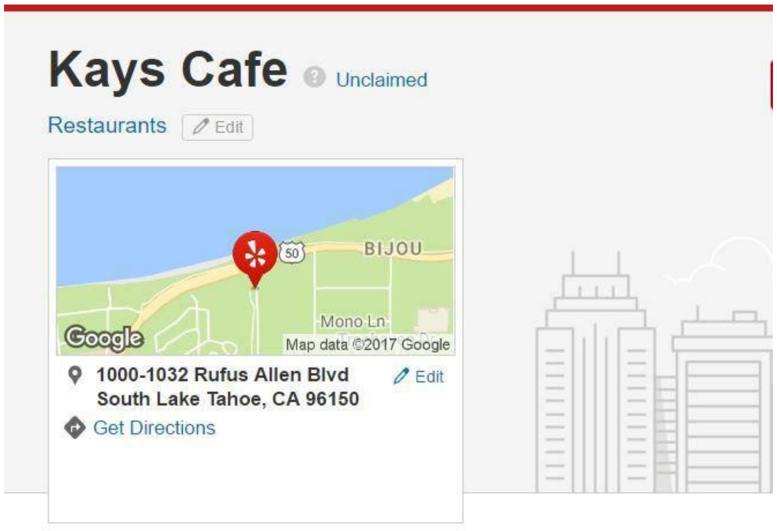
Was this review ...?
Useful Funny Cool

Figure 3.1: Label and review mismatch example

at the same time like Chinese or Thai. Probably the restaurant is a fusion restaurant, but multiple tagging of the main cuisine is misleading in such cases.

Given a photograph and review, we should be able to predict the label for the restaurant based on both the image and text features.

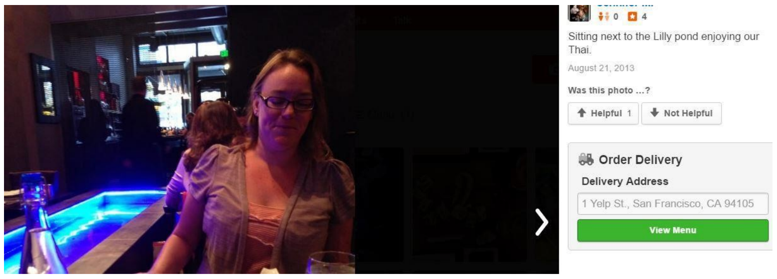
The input to the trained model in the pure classification task will be the set of features obtained from the user-review photos and text comments for a restaurant. The output from the model will be a unique predicted label for that restaurant. For example, if the set of possible labels are 1, 2, 3, 4, 5 for a restaurant business, the


Kays Cafe

Restaurants

1000-1032 Rufus Allen Blvd
South Lake Tahoe, CA 96150

Figure 3.2: Only the restaurant label


Sitting next to the Lilly pond enjoying our Thai.

August 21, 2013

Was this photo ...?

Order Delivery

Delivery Address

1 Yelp St., San Francisco, CA 94105

Figure 3.3: Good review1 with unrelated image

output of the model may be the predicted label (1) which means that label 1 applies to that business.

We need to extract the features from images and text reviews and make use of those features to train the model. So the proposal is to approach the problem in phases.

1. Initially, obtain the features of images and train the model to predict the label.

2. Then, obtain the features of text reviews to train the model and predict the

label.

3. Finally, combine approaches 1 and 2 to train the model with the combination of features from text and image.

The accuracy in predicting the label is most likely to increase in phase 3 where the text and image features are combined. The label classification is less likely to be apt in phase 1 or 2, since there could be images with insufficient information from comments or vice versa as shown in the Figure 3.4 or Figure 3.5.

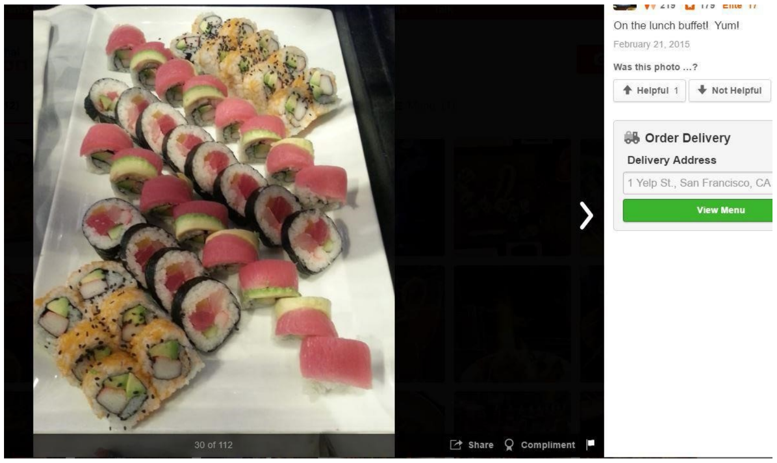
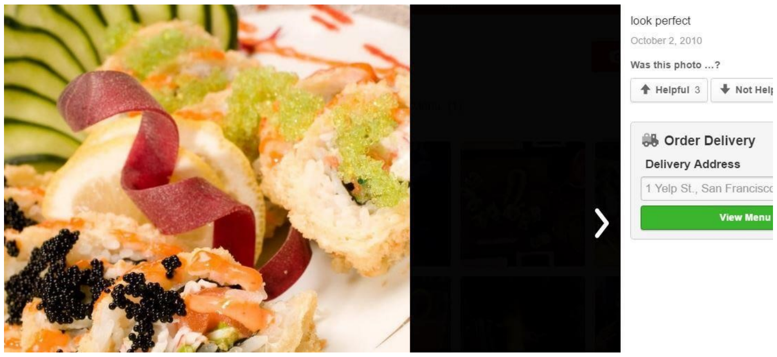

On the lunch buffet! Yum!
February 21, 2015

Figure 3.4: Good image1 with unrelated review

Figure 3.6 is the use case of the OCR algorithm, where the image is of textual data. The features obtained from the image in this case, would not be useful to predict the label but could give meaning text features if converted to textual format.

We approached the problem of restaurant label classification from Yelp images with the CaffeNet convolutional neural network architecture with transfer learning from a trained BVLC AlexNet and a custom ensemble approach. CaffeNet architecture from eight trained models like alexnet, googlenet, reference caffenet, SqueezeNet

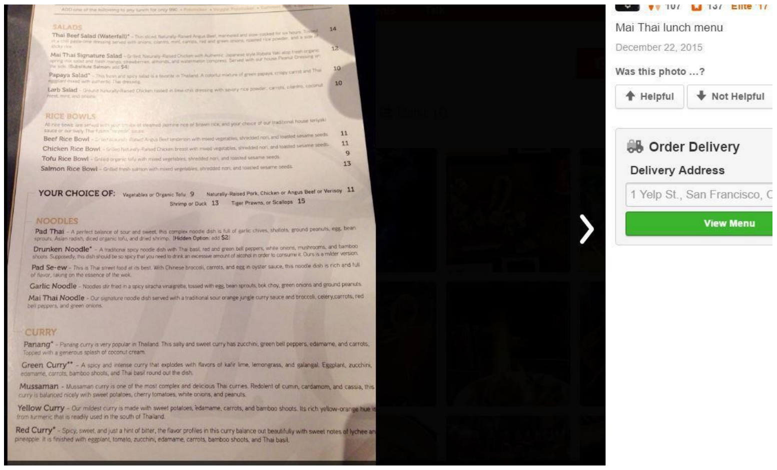


look perfect

October 2, 2010

Was this photo ...?

Figure 3.5: Good image2 with unrelated review



Mai Thai lunch menu

December 22, 2015

Was this photo ...?

Figure 3.6: Informative review2 with unrelated image

and VGG19 was used to extract (1000 or 4096) features of the 200K images based on the model in use. Extract features of Yelp reviews from various approaches will be discussed below.

We train the classifier based on the features obtained from text data, which includes the Yelp user reviews and the features obtained from Yelp review photographs. We classify the labels for new restaurants using the trained model.

3.1.1 Label classification from image feature extracted from pre-trained CNNs

Pre-trained convolutional neural network models that were originally trained on a large scale natural image dataset like ImageNet can be directly generalized to other datasets specific to other application domains. The main advantage of these pre-trained networks is that they have already learned a rich set of features from a wide range of natural images. Traditionally, machine learning with image datasets used to require a lot of effort in hand-picking important features, i.e. most of the work used to be focused on feature engineering.

Like we mentioned earlier, a downside of the neural networks is that they require a lot of data to be properly trained and demand a lot of computational resources and experimentation time. Fortunately, researchers from academia and industry have already trained such powerful CNN models with large-scale datasets such as ImageNet and have made the trained models (the learned parameters of the CNN) publicly available for other researchers to use in other relevant tasks. In this thesis, we use a pre-trained CNN model: Alexnet [25] obtained from Caffe library.

Like every other neural network, convolutional neural networks also have hierarchical and layered-based network architecture. The raw input images are fed from the lowest input layer and the network successively learns from general (eg. edges, color blobs) to more abstract (eg. cakes, pizza etc) concepts in images through a hierarchical representational learning process. The pre-trained models that we are using in our implementation were all trained originally on millions of natural images over more than twenty thousand categories for multi-class classification tasks, and these models have learned very general features (applicable to nearly any type of existing images) in the lower layers.

Since our dataset is very small in comparison to ImageNet, we were able to extract rich features for our images even from the higher-level layers (fully connected layers on the output end). Usually, once these CNN are used to convert the original images(user uploaded review images in our case) into a set of meaningful features, these features can then be used to train the new classifiers such as an extra tree classifier applied to a new task which the original CNN models were not trained for. In this thesis, one of our major goals is to use the extracted features using pre-trained models for a classification task.

Algorithm 1 Restaurant label classification with image features extracted from CNN-Alexnet.

```
1: procedure F(M1, TrainIMGs, InferenceIMGs )
       Input : M1 is the CNN model like BVLC ALexnet used for feature extraction. TrainIMGs are a set of images and the features extracted for these will be used as a training set to train classifier model M2, and InferenceIMGs is the other set of images and the features extracted for these will be used to validate the pre-trained classifier.
       Output : Accuracy score(S)
       Train Phase
2:     TrFeatures ← getfeatures (M1,TrainIMGS)
3:     TrM ← trainCM(M2,TrFeatures)
       Inference Phase
4:     IFeatures ← getfeatures (M1,InferenceIMGS)
5:     S ← ValidateCM(TrM,IFeatures)
6:     return S
7: end procedure
```

Algorithm1 details: The total restaurants are stratified shuffle split into train and test and corresponding total images, say T images, of these restaurants are split into x number of **TrainIMGs** and y number of **InferenceIMGs** such that:

$$\mathbf{x + y = T}$$

getFeatures method takes the images and CNN model like BVLC ALexnet to extract features. Initially in the training phase, we use the **CNN model M1** to extract **TrFeatures** which are the feature set extracted from TrainIMGS.

The trainCM method uses the TrFeatures obtained in step2 to train the classifier model M2 .

In the inference phase, we use the same getfeatures method to extract **IFeatures** which is the feature set extracted from InferenceIMGS.

The ValidateCM method uses the IFeatures obtained in step4 to validate the trained model TrM . The output of this method is the accuracy score S, used for validating the performance of classifier.

3.1.2 Label classification through TF-IDF,LSTM and word2vec models based on textual features

Label classification can also be done based on the user-submitted reviews. We obtained the reviews for all the businesses and passed these reviews through models like TF-IDF, LSTM, that can be used to predict the label.

TF-IDF feature extraction

TF-IDF can be considered as a baseline approach for restaurant label classification in this thesis. In a large text corpus, some words will be high-frequency words (e.g. the, a, is in English) but carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier, those very frequent terms would shadow the frequencies of rarer yet more interesting terms. In order to re-weight the count features into floating point values suitable for usage by a classifier, it is very common to use the TF-IDF transform.

As discussed in Section 2.3 , Tf means term-frequency while TF-IDF means term-frequency times inverse document-frequency:

$$TF - IDF(t, d) = tf(t, d) \times idf(t)$$

The term frequency, the number of times a term occurs in a given document, is multiplied with idf component, which is computed as

$$idf(t) = log(1 + n_d)/1 + df(d, t)$$

where n_d is the total number of documents, and df(d,t) is the number of documents that contain term t. The resulting TF-IDF vectors are then normalized by the Euclidean norm:

$$v_{norm} = v/\sqrt{v_1^2 + v_2^2 + ...v_n^2}$$

This was originally a term weighting scheme developed for information retrieval (as a ranking function for search engine results) that has also found good use in document classification that we are utilizing to classify the reviews to their cuisine type. The feature size obtained through this method is the size of the vocabulary. These features of businesses are split into training and inference. The training features are input to a classifier model. The trained model is then tested with the inference features to predict the accuracy to evaluate the baseline model.

Algorithm2 details: The total restaurants are stratified shuffle split into train

Algorithm 2 Restaurant label classification with text features extracted from TF-IDF model.

```
1: procedure F(M1, TrainRevs, InferenceRevs )
       Input : M1 is the feature extraction method TF-IDF. InferenceRevs are a
   set of reviews and the features extracted for these will be used as a training set to
   train classifier model M2, and InferenceRevs is the other set of reviews and the
   features extracted for these will be used to validate the pre-trained classifier.
       Output : Accuracy score(S)
       Train Phase
2:     TrFeatures ← getfeatures (M1,TrainRevS)
3:     TrM ← trainCM(M2,TrFeatures)
       Inference Phase
4:     IFeatures ← getfeatures (M1,InferenceRevS)
5:     S ← ValidateCM(TrM,IFeaturess)
6:     return S
7: end procedure
```

and test and corresponding total reviews, say T reviews, of these restaurants are split into x number of **TrainRevs** and y number of **InferenceRevs** such that:

$\mathbf{x + y = T}$

The getFeatures method takes the reviews and TF-IDF model to extract features. The feature size obtained from TF-IDF is the size of the vocabulary. Initially, in the training phase, we use the **TF-IDF model M1** to extract **TrFeatures** which are the feature set extracted from TrainRevS.

The trainCM method uses the TrFeatures obtained in step2 to train the classifier model M2 extra tree classifier.

In the Inference phase, we use the same getfeatures method to extract **IFeatures** which are the feature set extracted from InferenceRevS.

The ValidateCM method uses the IFeatures obtained in step4 to validate the trained model TrM . The output of this method is the accuracy score S, used for validating the performance of the classifier.

LSTM feature extraction

Instead of simply taking a review and returning a label, an RNN also maintains internal memories about the world to help perform its classifications. The knowledge of the world can change pretty chaotically. This chaos means information quickly transforms and vanishes, and it's difficult for the model to keep a long-term memory. So what we'd like is for the network to learn how to update its beliefs, in a way that its knowledge of the world evolves more gently. Then this is through long short-term memory network. An RNN can overwrite its memory at each time step in a fairly uncontrolled fashion. An LSTM transforms its memory in a very precise way: by using specific learning mechanisms for which pieces of information to remember, which to update, and which to pay attention to. This helps it keep track of information over longer periods of time.

We adopt the LSTM methodology to obtain features to train the classifier. The challenge is to build a model that can classify multiple sentences of different lengths at the same time.

The user-written reviews are not all of the same length for all the businesses. But, we need fixed length sequences to pass through the model. To resolve this issue, we have considered a few alternatives.

1. First approach was to consider a fixed length for a sequence (ideally the length of the shortest sequence) and cut the longer sequences to this fixed length. The drawback of this approach was the loss of useful information. Since the sequence length was varying from as short as 40 words to as long as 200,000 words. This leads to the loss of useful information and was ruled out.

2. Second approach was since sequences are of different lengths, we cannot feed

them into a Tensorflow graph as is, so we create a different tensor for each. This is an inefficient and difficult approach.

3. Third approach was padding the shorter sequences to the length of the longest sequence. If we pad shorter sequences so that all sequences are the same length, then all sequences will fit into a single tensor. The drawback with this approach was again the variation in lengths, which lead to a lot of padding of shorter sequences, resulting in a loss of memory. This approach was not ideal in our case, where we had varying length sequences and large amounts of vocabulary.

4. The final approach was to consider approach 3 with better performance using the bucketing and padding approach that has been considered in this thesis.

General procedure Tensorflow [1] is used to construct an RNN that operates on batches of input sequences of variable lengths. We will use this RNN for the classification task. Initially, there has to be a column of data mentioning the size of each sequence (review). An RNN is then built that accepts batches of data from this column. A padded iterator is then constructed to pad zeroes to shorter sequences in the batch.

A note on PAD symbols: *For this model, the zero that we used to pad our input sequences which is the index of the UNK symbol (representing UNKnown words) in our vocabulary. In this case, what we pad with does not affect the outcome, and so I chose to keep it simple, without the need to introduce a special PAD symbol. For example, here we will be feeding in a length tensor that holds information about our input sequence lengths. The advantage of the approach shown here (zero-padding with no special PAD symbol) is that it generalizes better to sequences with multi-dimensional*

[1] https://www.tensorflow.org/

continuous input. In such cases, it does not really make sense to have a separate PAD symbol.

Now, a sequence classification model is constructed using the padded data that assigns a single label to an entire input sequence.

Improving training speed using bucketing:

For the network above, if we use a batch size of 256, for example, each example in the batch has a different length ranging from 8 to 300. As the maximum length for each batch is usually very close to 300, short sequences required a lot of padding (e.g., all sequences of length 8 in the batch are padded with up to 292 zeros). Given this dataset, each batch is padded with an average of over 300,000 zeros, or over 100 padding symbols per sample.

This leads to a lot of excess computation, and we can improve upon it by bucketing our training samples. If we select our batches such that the lengths of the samples in each batch are within, say, 5 of each other, then the amount of padding in a batch of 256 is bounded by 256 * 5 = 1280. This would make our worst case outcome more than three times as good as the previous average case outcome.

There are many ways one might implement this, but the key point to keep in mind is that we should not bias the order in which different sequence lengths are sampled any more than necessary to achieve bucketing. E.g., sorting our data by sequence length might seem like a good solution, but then each epoch would be trained on short sequences before longer sequences, which could harm results.

By comparing the difference in training speed, observed that this bucketing strategy speeds up training by about 30% and average padding / batch is improved by a factor of about 6.

Algorithm3 details: The total restaurants are stratified shuffle split into train

Algorithm 3 Restaurant label classification with LSTM model.

```
1: procedure F(M, paddedTrainSeq, paddedISeq )
       Input : TrainFeats are a set of features extracted for train reviews sequences
   TrainSeq. These will be used as a training set to train classifier model M, and
   InferenceFeats is the other set of features extracted from test reviews sequences
   InferenceSeq. These will be used to validate the pre-trained classifier model.
   Seqlen is a parameter that defines the sequence length
       Output : Accuracy score(S)
       Train Phase
2:     batchedTrainSeq ← batchingwithbucketing(TrainSeq,seqlen)
3:     paddedTrainSeq ← paddediterator(batchedTrainSeq,seqlen)
4:     TrM ← trainCM (M,paddedTrainSeq)
       Inference Phase
5:     batchedISeq ← batchingwithbucketing(InferenceSeq,seqlen)
6:     paddedISeq ← paddediterator(batchedInferenceSeq,seqlen)
7:     S ← ValidateCM(TrM,paddedISeq)
8:     return S
9: end procedure
```

and test. The corresponding total reviews, say T reviews, of these restaurants are split into x number of **TrainRevs** and y number of **InferenceRevs** such that:

$\mathbf{x + y = T}$

Vocabulary is generated for the reviews and an identification is associated to each word in the vocabulary. The sequence of words is converted to sequence of associated identifications from the vocabulary file. Thus, we generate **TrainSeq** which is sequence of identifications for TrainRevs. Similary **TestSeq** is generated for InferenceRevs.

Initially in the training phase **batchingwithbucketing** takes the TrainSeq and **seqlen** which is the sequence length (an int value representing the number of words in a sequence) as input to generate **batchedTrainSeq** which is batches of data using bucketing as mentioned in earlier section.

paddediterator is then constructed to pad zeroes to shorter sequences in batched-

TrainSeq by reviewing the seqlen to obtained **paddedTrainSeq**.

trainCM method takes the paddedISeq obtained in step 3 as input and trains the LSTM model M. In inference phase, a similar approach is used to obtain the paddedIseq which are the sequences of shorter reviews that are padded. **ValidateCM** method uses the paddedInferenceSeq obtained in step 6 to validate the trained model TrM. The output of this method is the accuracy score S, used for validating the performance of classifier.

Table 3.1: Comparing LSTM result with baseline approach

Model	Accuracy
TF-IDF	O.66
LSTM	0.86

As it is possible to see from the Table 3.1 LSTM outperforms the base line approach TF-IDF for our dataset and the main motivation is due to its ability to retain only the useful parts of a sentence. There is a comparable difference in the result from LSTM when compared to TF-IDF. This could possibly be due to the difference in the working of TF-IDF which does not have knowledge of context, instead it is just based on keywords, unlike LSTM which has remembrance power.

Table 3.2 shows a few examples from real-time data that bring a difference between the possible accuracy difference between TF-IDF and LSTM.

Table 3.2: Reviews that support the LSTM performance (context based) compared to TF-IDF(key-word based)

Actual cuisine type	Misleading Keyword	Review text
Japanese	Italian	While there are both traditional and non-traditional ramen dishes, as well as the carbonara. I was expecting souply udon with a slight hint of **italian** flavoring, but no. It was more of the opposite way around - pasta with a hint of tonkatsu ramen flavoring, which was still delish! It was really tasty. Pretty heavy, but really good and was also rich in flavor. I would go here again.
Italian	Chinese	I'm giving the food 5-stars. Outstanding. I had the home made stued pasta and a cannoli for desert. Delicious. Only small nit pick is that the restaurant decor is lacking. Between **chinese** bueffet and noodles is good,food makes it worth it!
Japanese	Italian	I like ayce but what happened to those **italian** dishes? chefs do serve high quality sushi though. Smaller amount of rice compared to other ayce sushi joint, which, im sure is a good news to ladies. Too bad, only one order of dessert, bc im still dreaming of that green tea pannacotta.

It is clearly evident from these examples that the keyword approach could be misleading in the classification task and an actual understanding of the word used in the context that is necessary to achieve good performance. The result of LSTM in pure classification task motivates us to use LSTM in the context of novelty detection also. So, in the next section, we explore LSTM in terms of novelty detection task.

Classification through Word2vec embedding

As discussed earlier, Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of texts and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

There are 2 approaches in an attempt to embed word vector into our model:

1. Use an embedding model trained explicitly on current corpus of text.
2. Use pre-trained word word2vec / Glove word vectors as inputs to the model.

In the first option, everything has to be learned from scratch. The second one is good, but, the model will be unnecessarily big with all the word vectors for words that are not frequently used. In our thesis, this embedding layer (word2vec) has been passed before LSTM. LSTM takes these sequences of vectors as input and processes them.

We have used pre-trained word2vec model and a model that is explicitly trained on our dataset. Both of these approaches are considered in 2 ways

1. with the model being able to change the embedding and

2. fixed embedding.

The approach of this model is similar to LSTM except that an embedding layer of word2vec is added to produce sequence of feature vectors which are passed through LSTM.

An embedding layer for word2vec is followed by LSTM layer. Stemming has been done on data before the evaluation and then both training and test are stemmed.Vectors were used to represent the words. The accuracy is measured to evaluate the performance of the model and is mentioned in detail in the next chapter. LSTM without word2vec but with its own embedding based on associating a vector of shape[state size] to each word (trained the LSTM and this vector at the same time) worked better than LSTM with word2vec embedding where each word is associated with its vector(whether pre-trained or the one created from scratch). The reason for this is believed as our own embedding has some form of supervised learning, since the data is trained with its ground-truth and the entire sequence of words in the review are considered. In the case of word2vec embedding, the feature vectors of words are basically related to a window of words around them but not necessarily, the whole sentence of words. Features are obtained from the last layer of this trained LSTM model and are used in novelty detection task. So, the embedding layer (our own embedding [vocabulary size, state size] or a word2vec embedding) is used and these sequence of vectors are passed through LSTM layer. Features are extracted from the last layer of LSTM and used in the classification task.

3.2 Novelty detection for unknown cuisines

Novelty detection as described in [53] is the task of detecting data that differs in some respect from the data that is available during training. By definition, novelties (or unknown observations) are not present in the training set and appear only on the test set.

In this thesis, we consider three commonly used effective novelty detection algorithms:

(i) Gaussian Mixture Models

(ii) One-class Support Vector Machines and

(iii) Isolation Forests

Please refer to Chapter 2.2 for a detailed description of each one of these methods.

Implementation of novelty detection involves taking the extracted feature representation from LSTM, autoencoder etc. And train each of these novelty detection algorithms using known cuisine samples (datasets detail and data preparation is explained in Chapter 4) as training data and finally compute the novelty score for the cuisine samples in the test data. These novelty detection algorithms return a score as output specifying the degree of novelty for data in test data.

We describe our novelty detection procedure in Algorithm 4. Algorithm 4 takes in input an LSTM model M1. Its last layer is the one that we decide to extract the features from. In addition, there are 2 types of reviews: 1. reviews from known cuisines 2. reviews from unknown cuisines.

Reviews of known cuisines are split into knowntraining and knownInference.

Reviews of novel cusines are all considered for inference. knowntraining cusines reviews are converted to **TrainSeq** using the vocabulary file as mentioned in the earlier section. Similarly knownInference and novel are combined to obtain **InferenceSeq**. The procedure is divided into two parts: training session and testing session.

Firstly, in training session, TrainSeq features are processed as mentioned in steps 3 and 4 to obtain **TrainFeats**. These features are extracted from the last layer of **LSTM model M1.**

These extracted features are used to train the **novelty detection models M2** like GMM, One Class SVM, Isolation Forest by calling TrainNDM.

After the novelty detection models M2 are trained, we proceed with the scoring session. The scoring session is responsible for extracting the InferenceFeats. Then scoring them is done with the trained novelty detection models. The output of this procedure is a score (S).

Two performance metrics - AUROC and Average Precision [18] are computed to validate the results.

Parameter tuning Standard novelty detection procedure as (i) Gaussian Mixture Models (ii) Isolation Forests (iii) One-class Support Vector Machines are all different types parametric algorithms. These parameters must be tuned to understand appropriate algorithmic parameter values for the specific application. Some tunable parameters for One-class SVM are choice of kernel, norm (L1 or L2) and kernel coefficient. For clustering based algorithms like Gaussian Mixture Models, the number of estimated cluster is a parameter that needs to be selected appropriately. Similarly, for the Isolation Forest, we need to decide on what type of and how many decision tree classifiers we want to apply.

In this chapter, we have described the methodology of all the models used in our experiments. We shall now proceed to Chapter 4, to perform an analysis of the results obtained by implementing this methodologies.

Algorithm 4 Novelty detection with LSTM model.

```
1: procedure F(M1, paddedTrainSeq, paddedInferenceSeq )
       Input : TrainFeats are a set of features extracted for train reviews sequences-
       TrainSeq using LSTM model M1. These will be used as a training set to train
       novelty detection model M2, and InferenceFeats is the other set of features
       extracted from test reviews sequences InferenceSeq. These will be used to validate
       the pre-trained novelty detection model. Seqlen is a parameter that defines the
       sequence length
       Output : Accuracy score(S)
       Train Phase
2:     batchedTrainSeq ← batchingwithbucketing(TrainSeq,seqlen)
3:     paddedTrainSeq ← paddediterator(batchedTrainSeq,seqlen)
4:     TrainFeats ← getfeatures (M1,paddedTrainSeq)
5:     TrM ← TrainNDM(M2,TrainFeats)
       Inference Phase
6:     batchedInferenceSeq ← batchingwithbucketing(InferenceSeq,seqlen)
7:     paddedInferenceSeq ← paddediterator(batchedInferenceSeq,seqlen)
8:     InferenceFeats ← getfeatures (M1,paddedInferenceSeq)
9:     S ← ValidateNDM(TrM,paddedInferenceSeq)
10:    return S
11: end procedure
```

CHAPTER 4

EXPERIMENT AND RESULT ANALYSIS

In this section, we discuss the experiments conducted and perform analysis of the results obtained.

4.1 Datasets and Preprocessing

Datsets: The datasets used in this thesis are obtained from Yelp Dataset Kaggle Challenge. The Yelp dataset released for the academic challenge contains information for 200K images and a mapping to 86k businesses. In total, the Yelp dataset[1] provides 85K businesses associated with 2.6M reviews. Consideration has been made on those businesses that have at least 1 image and 1 review associated with them, which turned out to be 41k businesses. Consideration has been made on those businesses with a restaurant category associated to them. There are 15691 businesses with a restaurant label. There are a total of 271 cuisines under the restaurants category of which the top 15 main cuisine types like Italian, Chinese, American, Japanese, Thai, Mexican etc. were considered.

Data preprocessing: Yelp images and text comment reviews are obtained from the Yelp JSON files. Implementation of Alexnet in deep learning library, Caffe [21], requires the input images to be preprocessed in strict ways. The CNN model being

[1]https://www.yelp.com

used in this thesis requires Yelp user-review images to be resized to 256x256 pixels resolution followed by image transposition to CxHxW and channel order transformation from RGB to BGR. Additionally, the raw pixel values of each image were also required to be scaled up by a factor of 255 before feeding the input images to the CNN networks.

The reviews required to be cleaned using the NLTK package which includes but is not limited to removing HTML Markup: using the BeautifulSoup Package. BeautifulSoup is a very powerful library. Python wrapper for Stanford CoreNLP py-corenlp that could help condense the textual features to most useful ones is used. The details of it have been discussed in Section 2.7

4.2 Experimental Setup

In order to create a feasible experimental environment for restaurant classification, distinction of yelp review comments and images were defined. Following variations of datasets include:

Classification task:

1. With 3 cuisine types: The restaurant businesses under 3 main cuisine labels have been considered. All the reviews and images under these 3 restaurant businesses are stratified shuffle split into training and test to evaluate our defined classifier models.

2. With 5 cuisine types: The restaurant businesses under 5 main cuisine labels have been considered. All the reviews and images under these 5 restaurant businesses are stratified shuffle split into training and test to evaluate our defined classifier models.

3. With 15 cuisine types: The restaurant businesses under all 15 main cuisine labels were considered. All the reviews and images under these 15 restaurant businesses are stratified shuffle split into training and Inference to evaluate our defined classifier models.

In order to create a feasible experimental environment for novelty detection, the distinction of Yelp review comments to be considered as known (normal) and unknown (novel) were defined. By using the review dataset described in the previous section, following variations of our datasets have been created: **Novelty detection:**

1. With 2 unknown main cuisine types:Out of 15 main cuisines, 2 main cuisines are considered unknown and 13 cuisines are considered known.

2. With unknown fusion cuisine types:All 15 main cuisines are considered known, and fusion cuisines are considered unknown.

These variants of datasets provide us the way to make sure that our feature extraction procedure is not biased to any class of dataset and is generalizable to any type of dataset. This was done by applying (training and validating) these variants of dataset with our novelty detection algorithms.

Once each variant (dataset context) is properly defined, a 5-folds cross-validation is done[18] to evaluate the quality of our approach and take an average of this result.

4.3 Results and Discussion

4.3.1 Restaurant label classification:

Feature selection and extraction Caffe models were used to extract features from Yelp images and output to .pkl files. We have obtained the statistics of features like average, standard deviation, minimum and maximum for all the images in each of the 41k businesses. The final matrix for image features is of shape 41658 * 4000, which is 41,658 businesses as rows and corresponding 4000 features as columns.

To obtain textual features, Yelp reviews cannot be fed directly to the algorithms since they expect numerical feature vectors. For extracting numerical features from the text, we have to group together all the reviews associated with a particular business. We have to tokenize the strings to collect unigrams and give an integer for each unique token excluding punctuations and white spaces. We have to count the number of occurrences of tokens in each business and then normalize the tokens and weigh them with diminishing importance tokens that occur in the majority of samples. The corpus of all the textual data features is represented in the form a matrix similar to the image features matrix, with 1 row per business and 1 column per unique token occurring in the corpus. Consider condensing this matrix by cleaning the reviews and discard unimportant or not so useful reviews. Using NLTK package [28] to lemmatize the words in the dictionary and discard words that appear less frequently, Since words that appear infrequently are less useful for prediction. We use scikit shuffle split to randomly split the businesses into 80 training set and 20 inference set or suitable split to increase the performance. To reduce the dimension of the feature matrix, can apply principal component analysis (PCA) to the inputs of both the training set and the Inference set. Train and Inference the models on these training sets and test sets in order to see how the number of features influence the results for different learning algorithms.

Clustering

For better results we considered using a clustering algorithm such as K-means or agglomerative clustering to check which gives better results. As discussed in [10], the k-means algorithm is a powerful clustering algorithm. Input the businesses images features and divide them into clusters. Group the clusters of same businesses and for those businesses, which do not have images in a particular cluster, assign zero. The final matrix consists of businesses as the number of rows and features multiplied with the number of clusters as the number of columns.

Such matrices are found for both training and test data.

Classification

For the classification task obtain the corresponding labels of training and test datasets. Consider a classifier like the logistic regression or extra tree classifier. Based on ensemble methods [12, 7], in [73], Extra-Trees was presented which leads to a significant improvement of precision, and has various algorithmic advantages, in particular, reduced computational complexity with respect to classical trees and other ensemble methods. This model is utilized in our thesis.

We loaded the matrices of Image features as Xtrain and Xtest and the corresponding labels as ytrain and ytest to the classifier model and predict the labels for the businesses based on image features. We compared the predicted and actual values for labels to obtain the evaluation metrics such as accuracy, F1score and confusion matrix. Repeat the same procedure for text features as well.

We ignored the businesses that only have the image but no review and vice-versa. We obtained the businesses that have both images and reviews. We combined the features of image data and text data for such businesses and obtained a big matrix of businesses as rows and features of images and text together as columns. We obtained Xtrain , Xtest, ytrain, ytest for this matrix as mentioned above and formulated the evaluation metrics.

CaffeNet architecture from eight trained models like alexnet, googlenet, reference caffenet, SqueezeNet and VGG19 was used to extract 1,000 and 4,096 features depending on the model in use for the 200K images. Statistics for these features like the average, standard deviation, minimum and maximum value were obtained. Image feature statistics were obtained and were sorted based on the businesses.

Association between these 41k businesses and their corresponding labels have been found. There are 920 labels associated with these 41k businesses. A matrix of businesses and corresponding feature statistics of size 41658 * 4000 was obtained.

Classification task on 3 resturants

Table 4.1: Labels and number of businesses associated with it

Labels	BusinessCount
Chinese	880
Italian	1185
Japanese	589

Consideration has been made on a sample of the data obtained above with three main cuisines. That were initially Italian, Chinese and Japanese.

I shall discuss in detail the procedure followed with three cuisines. The same procedure was followed with five and fifteen cuisines as well. So, I shall directly get to the results in the latter case and discuss in detail the procedure for the three cuisines below.

The first step is to find the number of businesses associated with these cuisine labels and considered only the ones that are unique to the label, as shown in Table 4.1 which are altogether 2,654.

The number of images associated with these businesses were found as shown in Table 4.2 which are altogether 17,361 images. Currently, only the alexnet model was considered with 1,000 features output with ReLu and softmax layers. The features for these 17,361 images were obtained.

Scikit stratified split with the split as 80:20 for training and test dataset was done on these 2,654 businesses. 2123 businesses went under xtrain and 531 businesses into xtest.

Table 4.2: Labels and number of images associated with it

Labels	BusinessCount
Chinese	4698
Italian	6709
Japanese	7033

For restaurant label classification task, extensive experimentation has been done by considering the below scenarios:

1. Restaurant label classification based on image features obtained from Alexnet model.

2. Restaurant label classification based on image features obtained from fine-tuned Alexnet model.

3. Restaurant label classification based on text features obtained from the baseline TF-IDF model.

4. Restaurant label classification based on combination of image features obtained from CNN and text features obtained from TF-IDF.

5. Restaurant label classification based on text features obtained from LSTM model.

6. Restaurant label classification based on combination of image features obtained from CNN and text features through LSTM model.

7. Restaurant label classification based on text features obtained from pre-trained word2vec with fixed embedding.

8. Restaurant label classification based on text features obtained from pre-trained word2vec by allowing model to change the embedding.

9. Restaurant label classification based on text features obtained from word2vec model explicity trained on our dataset with fixed embedding.

10. Restaurant label classification based on text features obtained from word2vec model explicity trained on our dataset with the model able to change the embedding.

11. Restaurant label classification based on image features obtained from LSTM.

12. Restaurant label classification based on combination of image features and text features obtained from LSTM model.

label classification based on image features 13,854 images corresponding to 2,123 businesses were into the training dataset and 3,507 images corresponding to 531 test data businesses were into the test dataset. K-means and agglomerative models were used to divide these 13,854 training images into 10 clusters. The same model was used to obtain 10 clusters for 3,507 test images.

Agglomerative clustering with affinity cosine and manhattan were used. The clusters were then grouped based on images in each business. For those businesses that do not have images in a particular cluster are assigned zero. The final matrix(10000 * 2123) with 1000 features * 10 cluster as columns and the businesses that are represented by the rows is obtained for the training data and is called Xtrain. Similarly the matrix for the test data is obtained and is called Xtest (10000 * 531). Corresponding labels are obtained for Xtrain as ytrain and Xtest as ytest. The scikit extra tree classifier model is trained with the training dataset and the labels for test data is predicted. Multiple iterations were performed on the classifier with varying depths and `n_estimators` (number of trees). Similarly logistic regression model was trained with Xtrain data and labels for Xtest were predicted.

The evaluation metrics accuracy was obtained for extra tree classifier with varying parameters. The Table 4.3 displays the best of all evaluation metrics.

Table 4.3: Evaluation metrics for predicting labels based on image features

Model	Accuracy
Extra tree classifier	0.641468926554

label classification based on text features obtained from TF-IDF:

The reviews for each of the 2,123 businesses under xtrain were obtained and grouped together and reviews for each of the 531 businesses under xtest were also grouped together. The huge corpus of text data consisting of 2,654 businesses is stored into pandas [13] dataframe and converted into 2654x108231 sparse matrix, where 108,231 columns include the unique tokens obtained after the scikit tokenization by TF-IDFVectorizer . CountVectorizer method is used on the matrix to obtain the count of tokens. The numpy array of count tokens is then normalized by TF-IDFTransformer to obtain the features of text reviews using the TF-IDF term weighting. Tf means term-frequency while tfidf means term-frequency times inverse document-frequency [21]. The normalized data is then split for Xtrain and Xtest based on the training and test business dataset.

The final matrix(108231 * 2123) with 108,231 features as columns and the businesses are represented by the rows is obtained for the training data and is called Xtrain. Similarly the matrix for test data is obtained and is called Xtest (108231 * 531).

Corresponding labels are obtained for Xtrain as ytrain and Xtest as ytest. The scikit extra tree classifier model is trained with the training dataset and the labels for test data is predicted. Multiple iterations were performed on the classifier with varying depths and `n_estimators` (number of trees). Similarly, logistic regression model was trained with Xtrain data and labels for Xtest were predicted.

The evaluation metrics accuracy [16] was obtained for extra tree classifier with varying parameters. The Table 4.4 displays the best of all evaluation metrics [20].

Table 4.4: Evaluation metrics for predicting labels based on text features

Model	Accuracy
Extra tree classifier	0.661016949153

label classification based on combination of both text and image features The matrices of image features for training dataset (10000 * 2123) and test dataset (10000 * 531) are correspondingly merged with the matrices of text features for training (108231 * 2123) and test (108231 * 531) to obtain the final matrices for training dataset Xtrain (118231*2123) and test dataset Xtest (118231*531).

Corresponding labels are obtained for Xtrain as ytrain and Xtest as ytest. The scikit extra tree classifier model is trained with the training dataset and the labels for test data is predicted. Multiple iterations were performed on the classifier with varying depths and `n_estimators` (number of trees). Similarly logistic regression model was trained with Xtrain data and labels for Xtest were predicted. The evaluation metrics accuracy was obtained for extra tree classifier with varying parameters. The Table 4.5 displays the best of all evaluation metrics.

Table 4.5: Evaluation metrics for predicting labels based on combined features

Model	Accuracy
Extra tree classifier	0.702605047081

Observing the evaluation metrics from the preliminary result merging text and image suits can improve the prediction of labels compared to predicting the labels based on only image or text.

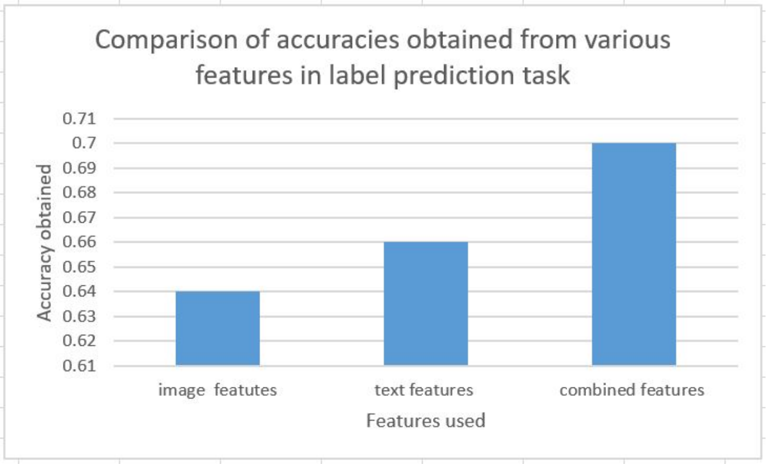


Figure 4.1: Comparison of accuracies obtained from various features in label classification task

Restaurant label classification based on text features through LSTM The same stratified split businesses are used in LSTM model to make the comparison accurate.

Initially, the vocabulary for training data is found. The training data is passed through LSTM using batching and bucketing with zero padding as mentioned in earlier sections.

The vocabulary size for three restaurants is found to be: 55695

The best result obtained on LSTM with 3 cuisines is 0.86

The major difficulty was dealing with such a huge amount of data. The vocabulary was considerably high, given the fact that there were only 3 restaurants considered.

label classification based on text features through word2vec

Distributed word vectors created by the Word2Vec algorithm were used. Word2vec, published by Google in 2013, is a neural network implementation that learns distributed representations for words. Other deep or recurrent neural network architectures had been proposed for learning word representations prior to this. But the major problem with these was a long time required to train the models. Word2vec learns quickly relative to other models.

From pre-trained model

Google's Word2Vec is a deep-learning inspired method that focuses on the meaning of words. Word2Vec attempts to understand the meaning and semantic relationships among words. It works in a way that is similar to deep approaches, such as recurrent neural nets or deep neural nets but is computationally more efficient. Each word is represented in a 300 feature vector. The vocabulary of our training data is mapped to their corresponding word vectors from this pre-trained model. These sequence of vectors are then passed through LSTM for the classification task.

The best accuracy obtained through this procedure is 0.42 with fixed embedding. By allowing the model to change the embedding, the result came down to 0.38.

Glove pre-trained 50 feature vector was also used as well and obtained identical results.

From the model trained on our dataset Our own word2vec embedding based on the vocabulary in reviews was created. Gensims word2vec:[2] can be used to create our own word2vec model. It expects a sequence of sentences as its input. Each sentence has a list of words. The input is a large corpus of text.

Word2vec accepts several parameters that affect both training speed and quality. One of them is for pruning the internal dictionary. Words that appear only once or twice in a billion-word corpus are probably uninteresting typos and garbage. In addition, there is not enough data to make any meaningful training on those words, so it is best to ignore them. Another parameter is the size of the NN layers, which correspond to the degrees of freedom the training algorithm has. Bigger size values require more training data but can lead to better (more accurate) models. Reasonable values are in the tens to hundreds. The last of the major parameters is for training parallelization, to speed up training using ”workers”. These models can be stored and loaded. The result did not drastically improve, even by training our own word2vec model. The best accuracy achieved is 0.48 with fixed embedding. By allowing the model to change it, the result obtained is 0.43

Restaurant label prediction from LSTM through image features Similar to text word2vec sequences, a sequence of image features from CNN was obtained and passed through LSTM. The result obtained is 0.39. By fine-tuning the model, and generating new image features from CNN and using these features slightly improved the result.

These results aid in realizing that text features are better at classification task

[2]https://radimrehurek.com/gensim/models/word2vec.html

than image features for this particular dataset. Also, especially, LSTM without word2vec embeddings performs better than with pre-trained word2vec or our own trained word2vec methods. LSTM without word2vec but with its own embedding based on associating a vector of shape[state size] to each word (trained the LSTM and this vector at the same time) worked better than LSTM with word2vec embedding where each word is associated with its vector (whether pre-trained or the one created from scratch). The reason for this is believed as, our own embedding has some form of supervised learning, since the data is trained with its ground-truth and the entire sequence of words in a review are considered. In the case of word2vec embedding, the feature vectors of words are basically related to a window of words around them but not necessarily the whole sentence of words. Since the context for classification task is better understood with a simple sequence of words passed through LSTM rather than by semantic relation between words.

Example of `confusion_matrix` for 3 cuisines from one of the five folds is mentioned below for a basic idea:

```
[[240 78 56]
[ 3 54 1]
[ 4 40 55]]
```

Table 4.6 portrays all the results for 3 cuisines for a better understanding and easy comparison.

Table 4.6: Results for 3 cuisines

etc CNN imgf	0.64
etc TF-IDF textf	0.66
etc CNN imgf and TF-IDF textf	0.70
LSTM W/O w2v: textf	**0.86**
LSTM with PTW2v fixed: textf	0.42
LSTM with PTW2v changing: textf	0.38
LSTM with DTW2v fixed: textf	0.48
LSTM with DTW2v changing: textf	0.43
LSTM imgf from CNN	0.39
LSTM fine-tuned imgf from CNN	0.40

Notations:

etc: extra tree classifier

imgf: image features

textf: text features

W/O: without

w2v : word2vec

PTW2V: pre-trained word2vec

DTW2V: word2vec explicity trained on our dataset

Classification task on 5 restaurants

Similar experiments were conducted by increasing the cuisines from 3 to 5. Since, the procedure in detail has been learnt with 3 restaurant classification, we shall directly look at the results obtained with 5 cuisines as shown in Table 4.7.

Example of `confusion_matrix` for 5 cuisines from one of the five folds is mentioned below for an understanding:

```
[[282 1 0 2 0]
[ 3 110 0 1 0]
[ 2 8 160 1 0]
[ 2 0 0 343 0]
[ 2 6 0 0 58]]
```

Table 4.7: Results for 5 cuisines

etc with TF-IDF(review features)	0.94
etc with CNN(image features)	0.62
Combination of CNN image and TF-IDF text features	0.76
LSTM without word2vec	**0.97**
LSTM with pre-trained word2vec fixed	0.41
LSTM with pre-trained word2vec changing	0.33
LSTM with dataset-trained word2vec fixed	0.43
LSTM with dataset-trained word2vec changing	0.38
LSTM (image features)	0.48
LSTM (fine-tuned image features)	0.51

LSTM outperforms TF-IDF in this case as well. LSTM without Word2Vec gives the best performance compared to any other model. word2vec did not yield better results than LSTM alone, as we have seen with three restaurants. The conclusion is that the image features are not very great in predicting the label even after fine-tuning the model. Also, concluding that LSTM alone outperforms any other method in the classification task.

Compared to three restaurants the improvement is that there was a dataset with awkward sequence lengths that made even a bucketed approach inefficient. For example, there could be lots of very short sequences of lengths 1, 2 and 3. Alternatively, there might be a few very long sequences among our shorter ones; we want to propagate the internal state forward through time for the long sequences, but do not have enough of them to train efficiently in parallel. One solution in both of these scenarios is to combine short sequences into longer ones. This has been implemented with five restaurant data.

Classification task on 15 restaurants

All the main cuisine types for the data classification task were considered. It leads to a huge amount of data and a large amount of vocabulary. A major concern when dealing with 15 restaurants undoubtedly was the size of the data. The dimensional reduction had been done and had to deal with segmentation errors and memory overflow. It was time consuming and required tedious experiments. The results can be found at Table 4.9.

An example of `confusion_matrix` for 15 cuisines from one of the five folds is mentioned below to aid understanding:

Table 4.8: Results for 15cuisine confusion matrix from one of the five folds

18	8	1	113	2	142	0	0	0	4	0	0	0	1	3
18	5	0	80	1	110	0	0	0	1	0	1	0	2	1
12	2	1	46	0	77	0	0	0	0	1	0	0	0	0
12	2	1	46	0	77	0	0	0	0	1	0	0	0	0
30	12	2	124	0	216	0	0	0	5	3	2	0	3	4
6	4	0	24	0	42	0	0	0	0	1	0	0	0	1
35	9	1	153	3	239	0	0	2	3	1	2	1	0	1
2	2	0	16	0	20	0	0	0	0	0	0	1	0	0
0	0	0	2	0	1	0	0	0	0	0	0	0	0	0
20	6	0	50	1	70	0	0	0	0	0	0	0	1	0
5	1	0	31	1	35	0	0	0	0	1	0	1	1	0
2	2	0	19	1	41	0	0	0	0	0	0	0	0	0
5	0	0	30	0	37	0	0	0	1	1	0	0	0	0
9	4	1	27	0	47	0	0	0	2	2	0	0	1	1
4	1	0	19	0	27	0	0	1	0	0	0	0	0	0
6	1	0	15	0	36	0	0	0	0	0	0	0	0	0

The major concern with 15 cuisines was dealing with such a huge amount of data. The vocabulary was so large and the training models took lots of time. Dimensionality reduction did not improve the result in this case. The batch size was decreased to avoid memory issues. Also, the data was so imbalanced. Multiple solutions were implemented to find which one works the best of all. Though no new algorithm is being introduced, dealing with such an unbalanced dataset and performing extensive experimentations, as well as spending hours together on refining the approaches were

Table 4.9: Results for 15 cuisines

TF-IDF (review features)	0.17
CNN image features	0.0972345
LSTM review without word2vec	0.3075729927
LSTM review with weights(penalizing error)	**0.33667**
LSTM with pre-trained word2vec fixed	0.0457875457
LSTM with pre-trained word2vec changing	0.0345289075
LSTM with self-trained word2vec fixed	0.051938317
LSTM with self-trained word2vec changing	0.048345677
LSTM image features	0.10
LSTM fine-tuned image features	0.11
LSTM review undersampling without word2vec	0.08
LSTM review oversampling without word2vec	0.23

the major efforts this thesis needed. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally.

Undersampling worsened the result, most likely due to the loss of useful information. Oversampling somewhat improved the result. But applying weights to penalize error works best with this data. As one can see from the above table, the best result with 15 cuisines is also from the LSTM model for text features without word2vec with applied weights yielding an accuracy of 0.33, which is good, considering the fact that the number of cuisines in this case is as high as 15.

4.3.2 Unknown cuisine novelty detection

Novelty detection with unknown cuisines is initially dealt with considering 2 of the main cuisines as unknown cuisines. The same approach is applied by considering fusion cuisines as unknown cuisines. The details of the results obtained in both cases are found below.

Data of the known main cuisines has been used only in the training phase of the novelty detection, but in the test phase, the fusion cuisines data has also been used to validate the models, i.e. the ability to discriminate between known and novel cuisines. This makes it a semi-supervised task.

Novelty detection with 2 unknown main cuisines

There are a total of 15 cuisines. Consider 2 cuisines as unknown cuisines and the other 13 cuisines as known ones. 13 known cuisines are split into knowntrain and knowntest. The 2 unknown cuisines are considered as unknown test and combine knowntest and unknowntest to obtain the finaltest set. Obtain the features for knowntrain from TF-IDF baseline approach. Also from the last layer of the LSTM model. The same model is used to extract features for finaltest. The novelty detection task is done by training the novelty detection algorithms on knowntrain features, which have only known data, and evaluating the model using finaltest features, which have both the known and unknown data .

The auroc result obtained from the TF-IDF approach for GMM, in this case, is 0.61 whereas the result obtained from the LSTM GMM is 0.99.

Novelty detection algorithm results from LSTM features with 2 unknown cusines

Table 4.10: Novelty detection algorithm results from LSTM features with 2 unknown main cuisines.

Novelty Detection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.990119562225	0.992921066081
One Class SVM	0.992341358709	0.993671796934
Isolation Forest	0.985908472029	0.989949392696

These are amazing results since the AUROC and AveragePrecision scores are 0.990 and 0.992 respectively.

Novelty detection algorithm results from autoencoder features with 2 unknown main cuisines. The features are extracted from the middle layer of autoencoder, which captures the embedding of input layer into lower dimensional space and uses these features in novelty detection of unknown cuisines. The result obtained from autoencoder is found in Table 4.11.

Table 4.11: Novelty detection results from autoencoder features with 2 unknown main cuisines

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.601874344342	0.414788990628
One Class SVM	0.591034582294	0.438294859303
Isolation Forest	0.589025464758	0.408273728273

The best result is 0.628 auroc obained from One Class SVM with an average precision 0.67.

Novelty detection of 2 unknown main cuisines with autoencoder reconstruction error

Autoencoder has the ability to effectively reconstruct the examples that have the similar statistical properties in the original feature space, thus obtaining the smaller reconstruction errors for known types of objects. An autoencoder tends to obtain higher reconstruction errors for the novel or unknown cuisines. Similar to any novelty detection algorithm, autoencoder is also trained with only the known examples of training data. Once the optimized embeddings are learned using the training data, the reconstruction errors are computed for all the known and novel data in the test set. The higher the reconstruction error, the higher is the chance of that data point to be unknown.

Table 4.12: Novelty detection algorithm results with 2 unknown main cuisines from autoencoder reconstruction error

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.85184076632	0.825355185297
One Class SVM	0.821761269932	0.812867560746
Isolation Forest	0.834689012973	0.83165725464

This result is better than the result obtained directly from autoencoder features.

Table 4.13: Auroc score for novelty detection with 2 unknown main cuisines - LSTM vs LSTMautoencoder(ae) vs reconstruction error(rce)

NoveltyDetection Algorithm	AUROC LSTM	AUROC LSTM ae	AUROC rce
Gaussian Mixture Model	0.990119562225	0.601874344342	0.85184076632
One Class SVM	0.992341358709	0.591034582294	0.821761269932
Isolation Forest	0.985908472029	0.589025464758	0.834689012973

Table 4.14: Average precision(AP) score for novelty detection with 2 unknown main cuisines - LSTM vs LSTMautoencoder vs LSTM reconsruction error

NoveltyDetection Algorithm	AP LSTM	AP LSTM ae	AP rce
Gaussian Mixture Model	0.992921066081	0.414788990628	0.825355185297
One Class SVM	0.993671796934	0.438294859303	0.812867560746
Isolation Forest	0.989949392696	0.408273728273	0.83165725464

Novelty detection with unknown fusion cuisines

All 15 main cuisines are considered as known and fusion cuisines as unknown data.

Novelty detection of fusion cuisines with LSTM features

The results obtained from the novelty detection algorithm of fusion cuisines with features obtained from the last rnn layer of the LSTM are discussed here.

Table 4.15: Novelty detection algorithm results for fusion cuisines with LSTM features

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.855935506451	0.859159097658
One Class SVM	0.728788505491	0.827839895438
Isolation Forest	0.716434135658	0.809050568295

The best result is 0.85 auroc obtained from Gaussian Mixture Models with an average precision score of 0.86.

Novelty detection of fusion cuisines with auto encoder features

The result obtained from novelty detection algorithm of fusion cuisines with features obtained from the autoencoder are discussed here.

Table 4.16: Novelty detection algorithm results for fusion cuisines from the autoencoder features

Novelty Detection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.622093362393	0.655705091671
One Class SVM	0.62871134868	0.677145963965
Isolation Forest	0.617121918368	0.66685578649

Novelty detection of fusion cuisines with autoencoder reconstruction error

Table 4.17: Novelty detection algorithm results for fusion cuisines from autoencoder reconstruction error

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.651250762288	0.625355185297
One Class SVM	0.621761269932	0.629047602555
Isolation Forest	0.634689012976	0.63165725465

This result is better than the result obtained directly from the autoencoder features.

Table 4.18: AUROC score for novelty detection with fusion cuisines LSTM vs LSTM autoencoder vs reconstruction error vs doc2vec

Model	LSTM	LSTM ae	rce	doc2vec
GMM	0.855935506451	0.622093362393	0.651250762288	0.50998531104
OneClassSVM	0.728788505491	0.62871134868	0.621761269932	0.51281439412
Isolation Forest	0.716434135658	0.617121918368	0.634689012976	0.508125328115

Table 4.19: Average precision score for novelty detection with fusion cuisines LSTM vs LSTM autoencoder vs reconstruction error vs doc2vec

Model	LSTM	LSTM ae	rce	doc2vec
GMM	0.859159097658	0.655705091671	0.625355185297	0.517519906488
OneClassSVM	0.827839895438	0.677145963965	0.629047602555	0.531196110004
Isolation Forest	0.809050568295	0.66685578649	0.63165725465	0.518403501311

Considering only 10% of fusion cuisines

In the earlier case, 15 main cuisines were considered as known and stratified shuffle split them into 90% training and 10% test1 and considered all of the fusion cuisines(unknown) as test2 and merged test1 and test2 to obtained the test set.

Now, we consider only 10% of fusion cuisines and unknown test2 and merge it with test1. This analysis is done to prepare the fusion cuisines according to the distribution of main cuisines.

The result obtained in this scenario from LSTM features, autoencoder and reconstruction error is shown in Table 4.20 , Table 4.21 and Table 4.22, respectively.

Table 4.20: Novelty detection algorithm results from LSTM features considering only 10% of unknown fusion cuisine data

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.810594739	0.9804579235359
One Class SVM	0.527839128192	0.94124127907
Isolation Forest	0.601979818285	0.948524537359

Table 4.21: Novelty detection algorithm results from auto encoder features considering only 10% of unknown fusion cuisine data

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.422010765345	0.905559954599
One Class SVM	0.537321624588	0.935497512444
Isolation Forest	0.481180484837	0.915297981663

Table 4.22: Novelty detection algorithm results from reconstruction error considering only 10% of unknown fusion cuisine data

NoveltyDetection Algorithm	AUROC	AvgPrecision
Gaussian Mixture Model	0.455456648907	0.934572579796
One Class SVM	0.543345667788	0.953246467853
Isolation Forest	0.513469747435	0.928953644654

Table 4.23: AUROC score for novelty detection with fusion cuisines(10% data) LSTM vs LSTM autoencoder vs reconstruction error

Model	LSTM	LSTM ae	rce
GMM	0.810594739	0.422010765345	0.455456648907
OneClassSVM	0.527839128192	0.537321624588	0.543345667788
Isolation Forest	0.601979818285	0.481180484837	0.513469747435

Table 4.24: Average precision score for novelty detection with fusion cuisines(10% data) LSTM vs LSTM autoencoder vs reconstruction error

Model	LSTM	LSTM ae	rce
GMM	0.9804579235359	0.905559954599	0.934572579796
OneClassSVM	0.94124127907	0.935497512444	0.953246467853
Isolation Forest	0.948524537359	0.915297981663	0.928953644654

Looking at the result of the classification task, comparing TF-IDF and LSTM without word2vec and with word2vec, LSTM without word2vec embedding outperformed every other model as shown in Figure 4.2

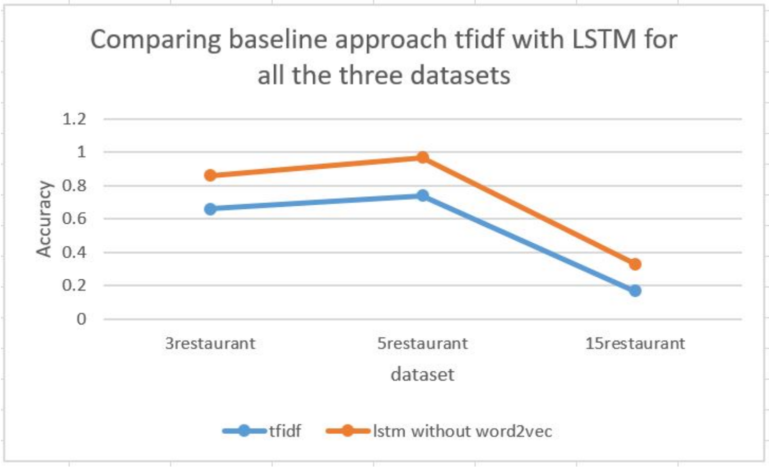


Figure 4.2: Comparing baseline approach TF-IDF with LSTM for all the three datasets in classification task

The Gaussian Mixture model result for 2 unknown cuisines, fusion cuisine and fusion cuisine with only 10% data in novelty detection task is mentioned in Figure 4.3 , 4.4 and 4.5.

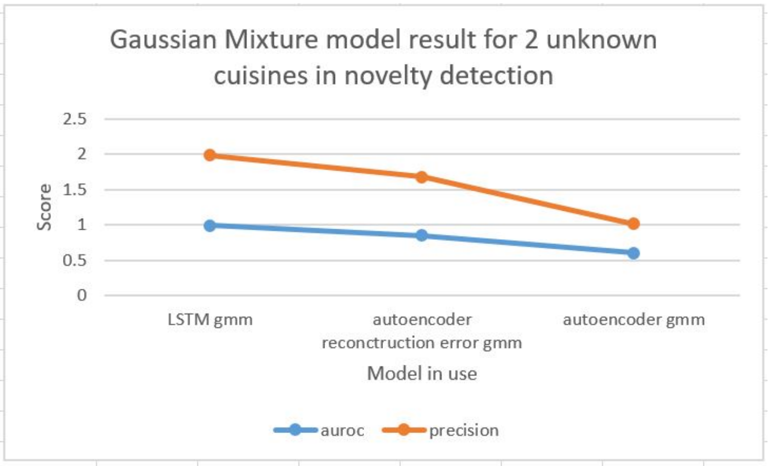


Figure 4.3: Gaussian Mixture model result for 2 unknown cuisines in novelty detection task

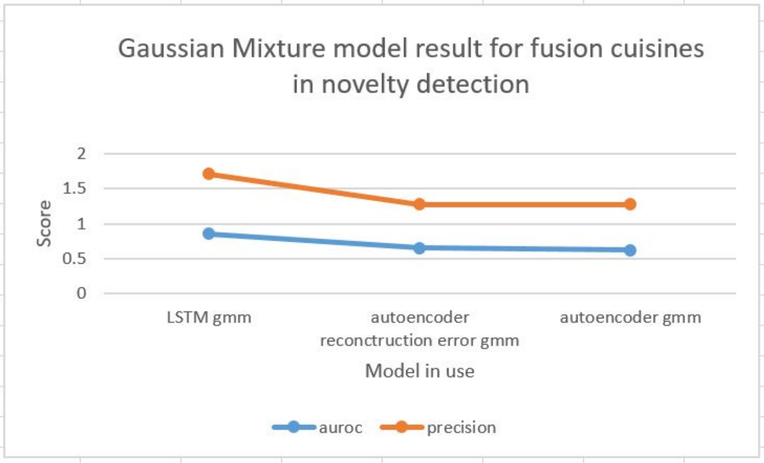


Figure 4.4: Gaussian Mixture model result for fusion cuisines in novelty detection task

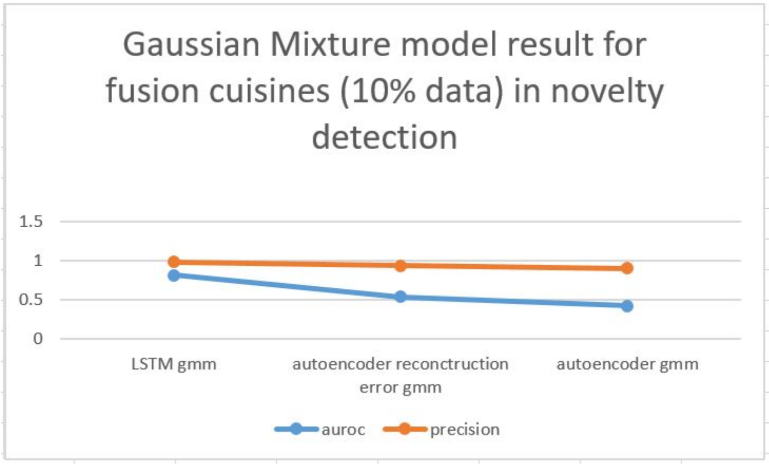


Figure 4.5: Gaussian Mixture model result for fusion cuisines in novelty detection task

Let us look at the auroc score and average precision score obtained from the 3 novelty detection algorithms by all 3 datasets for various models in use.

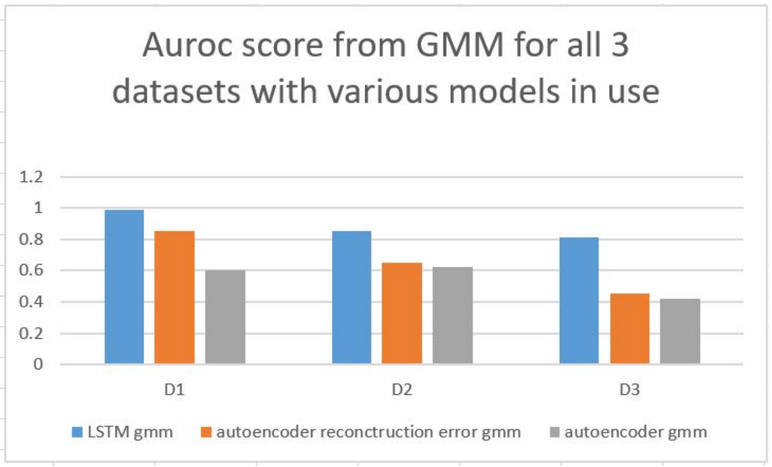


Figure 4.6: AUROC score from GMM for all 3 datasets with various models in use

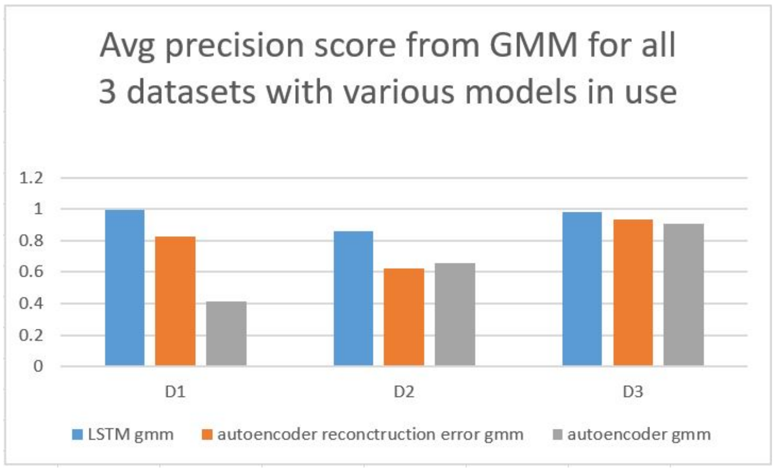


Figure 4.7: Avg precision score from GMM for all 3 datasets with various models in use

The three datasets refers to:

D1 = Novelty detection with 2 unknown main cuisines

D2 = Novelty detection with novel cuisines

D3 = Novelty detection with 10% novel cuisines

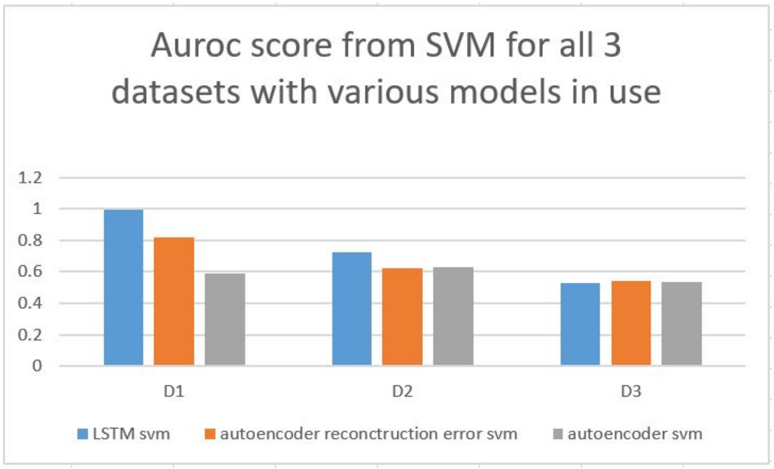


Figure 4.8: AUROC score from One Class SVM for all 3 datasets with various models in use

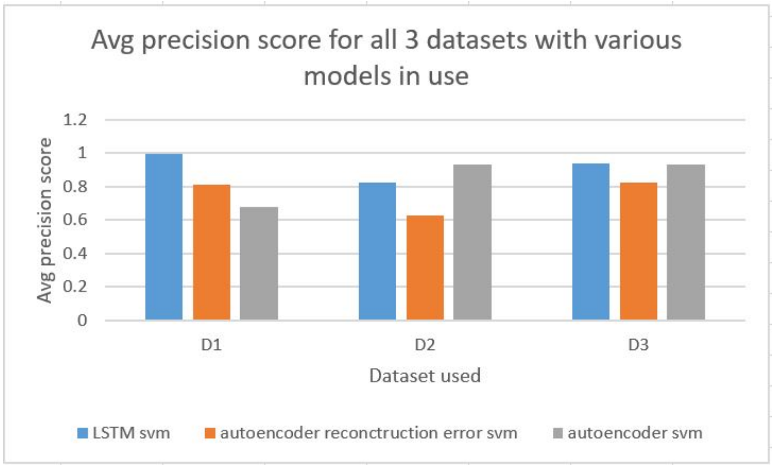


Figure 4.9: Average Precision from One Class SVM for all 3 datasets with various models in use

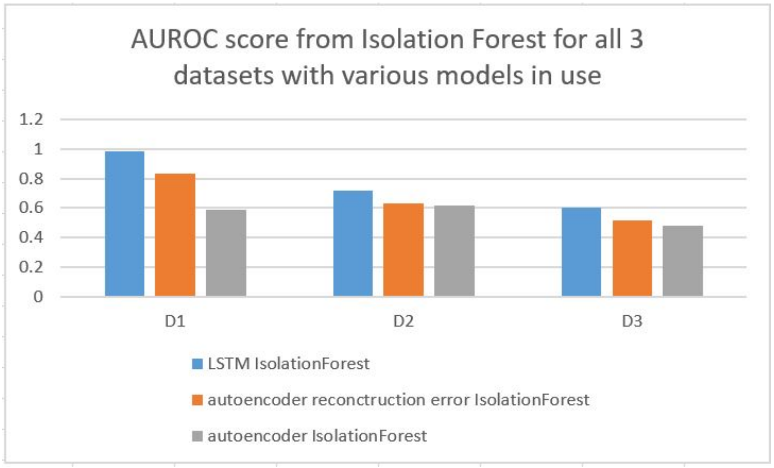


Figure 4.10: AUROC score from Isolation Forest for all 3 datasets with various models in use

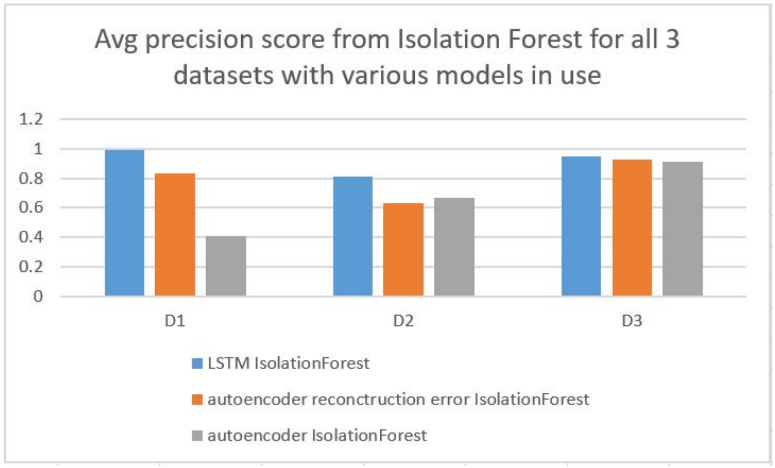


Figure 4.11: Average Precision from Isolation Forest for all 3 datasets with various models in use

The auroc score from the autoencoder reconstruction error is found to be better than LSTM in case of One Class SVM and Isolation Forest with dataset3 - obtained by considering 10% novel cuisines.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, a new problem of identifying the unknown type of cuisines was defined. This problem has a high impact on Yelp crowd sourcing. Restaurant label classification using baseline TF-IDF approach and LSTM approach was proposed. LSTM outperformed TF-IDF with a score of 0.97. This is likely due to the remembrance power of LSTM whereas TF-IDF is not context-based; instead, it is key-word based method could lead to misleading results in some cases, as discussed in earlier chapters. This analysis motivated us in approaching the novel cuisine identification problem using LSTM. Our novelty detection approach is using LSTM features, autoencoder features and reconstruction error. Standard novelty detection algorithms were applied to each. Results obtained are: 0.855 of AUROC score and 0.859 of Average Precision score for the unknown fusion cuisine novelty detection problem using the LSTM model with GMM.

Future Work: Our work applied the LSTM and autoencoder methodologies in novelty detection and we believe that there is room for improvement. The current model used in this thesis is semi-supervised and is very promising, but we also plan to approach this problem without the labels at all, that is, investigating more on the unsupervised approach. We would like to adopt the competitive learning approach to extract features from the LSTM automatically for the novelty detection task.

We also need to improve the time of training, since currently our model is slow and needs improvement in dealing with such a huge amount of imbalanced data.

By fusion cuisines, we just mean a combination of some main cuisines, but our model does not deal with the percentage of different main cuisines in fusion restaurants, which could be an interesting possibility to explore in the context of classification when there is more than one label.

Another possibility for the extension of our work is to use review images in novelty detection task. We can fine-tune the ensemble of CNN models. The ensemble of CNN models has also achieved state-of-the-art accuracy scores in various image classification tasks. By concatenating different CNN networks such as Alexnet and VGG19, and training them efficiently we might produce more representative CNN models which, in turn, might produce better feature representations of images that eventually improve the performance of standard novelty detection algorithms and restaurant label classification.

We could also explore our proposed approach in other application domains like forensic documents and patient health reports.

Also, we partially experimented with autoencoder neural networks using only the reconstruction error. We also look forward to using stronger denoising and variational autoencoders, which have gained success recently.

REFERENCES

[1] Ahmed Abbasi, Hsinchun Chen, and Arab Salem. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)*, 26(3):12, 2008.

[2] Ahmed Abbasi, Stephen France, Zhu Zhang, and Hsinchun Chen. Selecting attributes for sentiment classification using feature relation networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(3):447–462, 2011.

[3] Deepak Agarwal. Detecting anomalies in cross-classified streams: a bayesian approach. *Knowledge and information systems*, 11(1):29–44, 2007.

[4] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[5] Muhammad Zubair Asghar, Aurangzeb Khan, Shakeel Ahmad, and Fazal Masud Kundi. A review of feature extraction in sentiment analysis. *Journal of Basic and Applied Scientific Research*, 4(3):181–186, 2014.

[6] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[7] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):105–139, 1999.

[8] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[9] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[10] Hans-Hermann Bock. Clustering methods: a history of k-means algorithms. *Selected contributions in data analysis and classification*, pages 161–172, 2007.

[11] Richard J Bolton, David J Hand, et al. Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, pages 235–255, 2001.

[12] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[13] Lukas Cavigelli, Dominic Bernath, Michele Magno, and Luca Benini. Computationally efficient target classification in multispectral image data with deep neural networks. *arXiv preprint arXiv:1611.03130*, 2016.

[14] C Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970.

[15] Lei Clifton, David A Clifton, Peter J Watkinson, and Lionel Tarassenko. Identification of patient deterioration in vital-sign data using one-class support vector machines. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pages 125–131. IEEE, 2011.

[16] Lei A Clifton, Hujun Yin, David A Clifton, and Yang Zhang. Combined support vector novelty detection for multi-channel combustion data. In *Networking, Sensing and Control, 2007 IEEE International Conference on*, pages 495–500. IEEE, 2007.

[17] Lei A Clifton, Hujun Yin, and Yang Zhang. Support vector machine in novelty detection for multi-channel combustion data. In *International Symposium on Neural Networks*, pages 836–843. Springer, 2006.

[18] colah. Understanding lstm, 2015.

[19] Dragoş Datcu and Léon JM Rothkrantz. Emotion recognition using bimodal data fusion. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, pages 122–128. ACM, 2011.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[21] Kavita Ganesan and Chengxiang Zhai. Opinion-based entity ranking. *Information retrieval*, 15(2):116–150, 2012.

[22] Pedro Garcia-Teodoro, J Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.

[23] Andrew B Gardner, Abba M Krieger, George Vachtsevanos, and Brian Litt. One-class novelty detection for seizure analysis from intracranial eeg. *Journal of Machine Learning Research*, 7(Jun):1025–1044, 2006.

[24] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[26] David Hardoon and Larry M Manevitz. fmri analysis via one-class machine learning techniques. 2005.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[28] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1):177–196, 2001.

[29] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[30] Yangqing Jia and Changshui Zhang. Instance-level semisupervised multiple instance learning. In *AAAI*, pages 640–645, 2008.

[31] Hirokatsu Kataoka, Kenji Iwata, and Yutaka Satoh. Feature evaluation of deep convolutional neural networks for object recognition and detection. *arXiv preprint arXiv:1509.07627*, 2015.

[32] Soo-Min Kim and Eduard Hovy. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 483–490. Association for Computational Linguistics, 2006.

[33] Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *EMNLP-CoNLL*, volume 7, pages 1065–1074, 2007.

[34] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, 2006.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[37] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Karel Lenc and Andrea Vedaldi. R-cnn minus r. *arXiv preprint arXiv:1506.06981*, 2015.

[40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[41] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[42] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[43] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618. ACM, 2003.

[44] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[45] Erik Marchi, Fabio Vesperini, Florian Eyben, Stefano Squartini, and Björn Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1996–2000. IEEE, 2015.

[46] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[47] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pages 746–751, 2013.

[48] Keiller Nogueira, Otávio AB Penatti, and Jefersson A dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017.

[49] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[50] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1777–1784, 2013.

[51] Verónica Pérez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. Utterance-level multimodal sentiment analysis. In *ACL (1)*, pages 973–982, 2013.

[52] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

[53] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[54] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[55] Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE, 2016.

[56] Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer, 2001.

[57] A Rabaoui, H Kadri, and N Ellouze. New approaches based on one-class svms for impulsive sounds recognition tasks. In *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 285–290. IEEE, 2008.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[59] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[60] Pieter Bloemerus Ruthven. Contextual profiling of homogeneous user groups for masquerade detection. Master's thesis, 2014.

[61] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, et al. Fully convolutional neural network for fast anomaly detection in crowded scenes. *arXiv preprint arXiv:1609.00866*, 2016.

[62] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[63] Mark Sanderson, D Christopher, Hinrich Manning, et al. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100, 2010.

[64] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[65] Bjorn Schuller. Recognizing affect from linguistic information in 3d continuous space. *IEEE Transactions on Affective computing*, 2(4):192–205, 2011.

[66] Shahaboddin Shamshirband, Nor Badrul Anuar, Miss Laiha Mat Kiah, and Ahmed Patel. An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique. *Engineering Applications of Artificial Intelligence*, 26(9):2105–2127, 2013.

[67] Johan Sigholm and Massimiliano Raciti. Best-effort data leakage prevention in inter-organizational tactical manets. In *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pages 1–7. IEEE, 2012.

[68] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[69] Diveesh Singh and Pedro Garzon. Using convolutional neural networks and transfer learning to perform yelp restaurant photo classification.

[70] Niranjan Subrahmanya and Yung C Shin. Sparse multiple kernel learning for signal processing applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):788–798, 2010.

[71] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[72] Sutapat Thiprungsri and Miklos A Vasarhelyi. Cluster analysis for anomaly detection in accounting data: An audit approach. *International Journal of Digital Accounting Research*, 11, 2011.

[73] Louis Wehenkel, Damien Ernst, and Pierre Geurts. Ensembles of extremely randomized trees and some generic applications. *Proceedings of Robust Methods for Power System State Estimation and Load Forecasting*, 2006.

[74] wiki. tfidf.

[75] yelp. Yelp kaggle challenge, 2015.

[76] Dit-Yan Yeung and Yuxin Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, 36(1):229–243, 2003.

[77] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[78] Cha Zhang, John C Platt, and Paul A Viola. Multiple instance boosting for object detection. In *Advances in neural information processing systems*, pages 1417–1424, 2006.

[79] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635, 2013.

[80] Haiping Zhang, Zhengang Yu, Ming Xu, and Yueling Shi. Feature-level sentiment analysis for chinese product reviews. In *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, volume 2, pages 135–140. IEEE, 2011.

[81] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[82] Ling Zhuang and Honghua Dai. Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of Computers*, 1(7):32–40, 2006.

APPENDIX A

EVALUATION METRICS

ROC Curve

ROC or receiver operating characteristic curve is a two-dimensional curve, which shows the performance of binary classification under varying discrimination threshold17 . It is the plot of true positive rate (TPR) against the false positive rate (FPR) at different threshold values. In a classifier, accuracy is sensitive to the imbalance in classes. For example, in 100 data samples, if 80 of them are positive and 20 are negative, the classifier results in an accuracy of 80 percent at least. This means that the accuracy is only showing the distribution of classes in the dataset. With 80 percent data as positive, the probability of getting a positive sample is already 80 percent. The ROC curve is insensitive to this class imbalance.

Figure A.2 represents the sample ROC curve with an area of 0.78. The luck line shown in the diagram represents any classifier with random performance level. This is a baseline representing the performance level of the classifier. ROC curve in the upper left corner represents a good classification, while the ROC curve in the lower right corner represents poor classification.

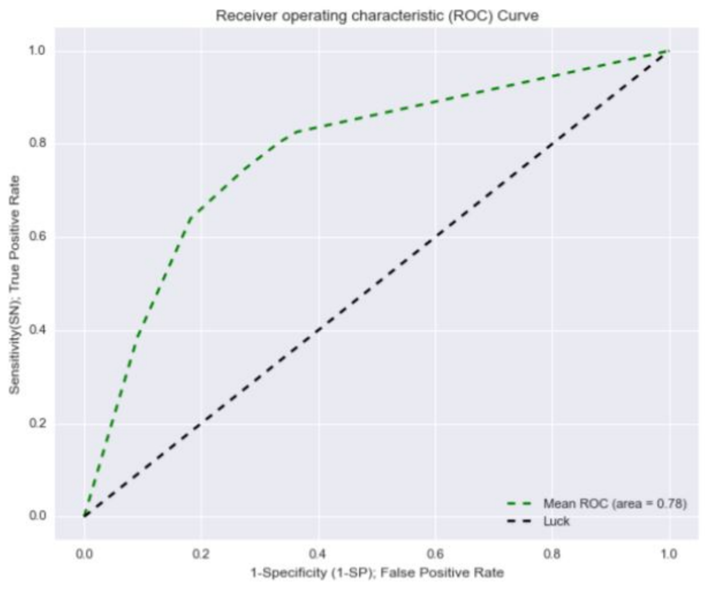


Figure A.1: Sample ROC Curve

[1].

AUROC (Area under the ROC curve)

The area under the ROC curve represents the probability that the classifier ranks a randomly chosen positive sample higher than the randomly chosen negative sample18 . The AUROC of the excellent classifier is 1. The dotted blue line in Figure A.1 has an area of 0.5. Therefore, any random predictor has AUROC of 0.5, which is used as a baseline in identifying the usefulness of the model.

Confusion Matrix

Binary classification produces four possible outcomes.

1. True Positive (TP): Positive samples predicted as positive

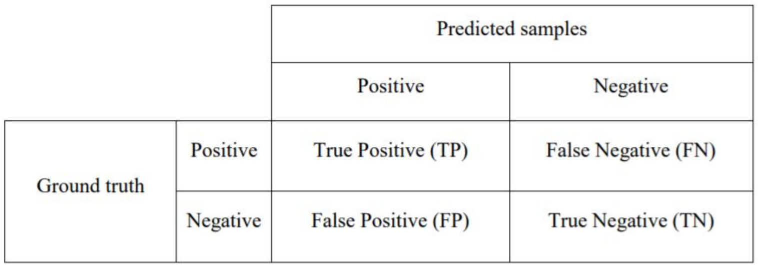
2. False Positive (FP): Negative samples predicted as positive

[1]https://en.wikipedia.org/wiki

3. True Negative (TN): Negative samples predicted as negative

4. False Negative (FN): Negative samples predicted as positive

A 2 by 2 table showing these four outcomes of the binary classification is called confusion matrix. Table below shows the confusion matrix with four possible outcomes.



		Predicted samples	
		Positive	Negative
Ground truth	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure A.2: Confusion Matrix

APPENDIX B

LIBRARY AND MODELS

Caffe

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. Yangqing Jia created the project during his Ph.D at UC Berkeley. Caffe is released under the BSD 2-Clause license.

Why Caffe

Expressive architecture encourages application and innovation. Models and optimization are defined by configuration without hard-coding. Switch between CPU and GPU by setting a single flag to train on a GPU machine then deploy to commodity clusters or mobile devices.

Extensible code fosters active development. In Caffes first year, it has been forked by over 1,000 developers and had many significant changes contributed back. Thanks to these contributors the framework tracks the state-of-the-art in both code and models.

Speed makes Caffe perfect for research experiments and industry deployment. Caffe can process over 60M images per day with a single NVIDIA K40 GPU*. Thats 1 ms/image for inference and 4 ms/image for learning and more recent library versions and hardware are faster still. We believe that Caffe is among the fastest convnet implementations available.

Community: Caffe already powers academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Join our community of brewers on the caffe-users group and Github.

ImageMagick

ImageMagick is a free and open-source software suite for displaying, converting, and editing raster image and vector image files. It can read and write over 200 image file formats.

ImageMagick was created in 1987 by John Cristy when working at DuPont, to convert 24-bit images (16 million colors) to 8-bit images (256-color), so they could be displayed on most screens. It was freely released in 1990 when DuPont agreed to transfer copyright to ImageMagick Studio LLC, still currently the project maintainer organization.

In May 2016, it was reported that ImageMagick had a vulnerability through which an attacker can execute arbitrary code on servers that use the application to edit user-uploaded images.[5] Security experts including CloudFlare researchers observed actual use of the vulnerability in active hacking attempts

scikit

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random Forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Natural Language ToolKit:NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook. NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

Tensorflow

TensorFlow is an open-source software library for machine learning across a range of tasks. It is a symbolic math library and also used as a system for building and training neural networks to detect and decipher patterns and correlations, analogous to human learning and reasoning. It is used for both research and production at Google, often replacing its closed-source predecessor, DistBelief. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on 9 November 2015.