



Centre for Modeling and Simulation  
Savitribai Phule Pune University

Master of Technology (M.Tech.)  
Programme in Modeling and Simulation

Project Report

# **Business Category Inference from Yelp Data**

Bhakti Khude  
CMS1406

Academic Year 2015-16





Centre for Modeling and Simulation  
Savitribai Phule Pune University

## Certificate

This is to certify that this report, titled

**Business Category Inference from Yelp Data,**

authored by

**Bhakti Khude** (CMS1406),

describes the project work carried out by the author under our supervision during the period from January 2016 to June 2016. This work represents the project component of the Master of Technology (M.Tech.) Programme in Modeling and Simulation at the Center for Modeling and Simulation, Savitribai Phule Pune University.

**Yogesh Badhe,**  
Senior Technical Specialist  
Persistent Systems Limited  
Near Nal Stop, Erandwane  
Pune 411004 India

**Valadi Jayaraman,**  
Honorary Adjunct Professor  
Centre for Modeling and Simulation  
Savitribai Phule Pune University  
Pune 411007 India

**Mihir Arjunwadkar,**  
Associate Professor  
Centre for Modeling and Simulation  
Savitribai Phule Pune University  
Pune 411007 India

**Anjali Kshirsagar,**  
Director  
Centre for Modeling and Simulation  
Savitribai Phule Pune University  
Pune 411007 India





Centre for Modeling and Simulation  
Savitribai Phule Pune University

## Author's Declaration

This document, titled

**Business Category Inference from Yelp Data,**

authored by me, is an authentic report of the project work carried out by me as part of the Master of Technology (M.Tech.) Programme in Modeling and Simulation at the Center for Modeling and Simulation, Savitribai Phule Pune University. In writing this report, I have taken reasonable and adequate care to ensure that material borrowed from sources such as books, research papers, internet, etc., is acknowledged as per accepted academic norms and practices in this regard. I have read and understood the University's policy on plagiarism ([http://unipune.ac.in/administration\\_files/pdf/Plagiarism\\_Policy\\_University\\_14-5-12.pdf](http://unipune.ac.in/administration_files/pdf/Plagiarism_Policy_University_14-5-12.pdf)).

**Bhakti Khude**

CMS1406



# Abstract

Business category inference will categorize any new or existing business into relevant business categories, given its business information, review and image. Yelp data is a huge database having so many various facets. Much research has been limited to extracting sentiment and predicting star rating from review data. In this report, I have tried to utilize maximum possible database to extract various types of features. Text features are created using Bag of Words and Word vectors. Much of the effort is put into separating the garbage from the text data. Image features are extracted using pretrained Convolutional Neural Networks. Feature engineering is done to get features from time data. Three multi-label machine learning algorithms are studied, compared and contrasted. Finally an ensemble of classifiers is built to bring all the information together and predict the category. The final model gives an accuracy greater than 95% and F1 score of 0.96.





# Acknowledgments

All good things come to an end, and so this piece of work has also come to an end. Though only my name appears on the cover of this Project report, a great many people have contributed to its production and made my Master's experience the one that I will cherish forever.

First and foremost, I owe my deepest gratitude to Dr. Yogesh Badhe for allowing me to intern at the prestigious Persistent Systems Limited, Pune. Dr. Yogesh also served as my guide at Persistent. He inspite of being extraordinarily busy with his duties, always took out time to hear, guide and keep me on the correct path. Discussions with him were always fruitful whenever I got stuck in my work. He made possible all the resources that I needed to carry out this piece of work.

I am indebted to Dr. V. K. Jayaraman for providing all the skill sets and serving as my internal project guide. He is the one who introduced me to the world of machine learning which I thoroughly enjoy working in. Jayaraman Sir sets very high standards for his students and he encourages and guides them to meet those standards. He enforces strict validations on each result which is why I had to do a deep research and give reasoning on every step in my work. Because of his continuous encouragement, I pushed myself to meet his expectations.

I am also thankful to Dr. Mihir Arjunwadkar for being my internal guide. He layed the foundations of stochastic techniques and R programming for me. I am very thankful for his inputs during our weekly report sessions.

I extend my gratitude to all the other faculty members at CMS.

My friends helped me stay sane through these two years. Their support and care helped me overcome setbacks and stay focused on my master's study. I must thank my friends Sheetal Ghanwat, Shrikant Panchal and Anirudh Jonnalgadda for being there for me through all times. This journey would never have become so memorable without them. I also thank Suraj Deshmukh for helping me with images with his knowledge on image data.

Most importantly, none of this would have been possible without the love and encouragement of my family. My parents Dnyaneshwar Khude and Daya Khude have always been a constant source of love, concern, support and strength all these years. Thank you so much!



*To my parents and my friends, who have supported me through my journey*



# Contents

<b>Abstract</b>	<b>7</b>
<b>Acknowledgments</b>	<b>9</b>
<b>1 Introduction</b>	<b>15</b>
1.1 Problem Statement	16
1.2 Prior work	16
1.3 The Dataset	17
1.3.1 Business	17
1.3.2 Review	18
1.3.3 User	18
1.3.4 Check-in	19
1.3.5 Tip	19
1.3.6 Photos	20
<b>2 Literature Review: Multi Label Classification Methods</b>	<b>21</b>
2.1 Problem transformation methods	22
2.1.1 Label Powerset (LP)	23
2.1.2 Binary Relevance (BR)	23
2.1.3 Ranking by Pairwise Comparison (RPC)	24
2.1.4 Calibrated Label Ranking (CLR)	24
2.2 Algorithm adaptation methods	25
2.2.1 Tree Based Boosting	25
2.2.2 Lazy Learning	25
2.2.3 Neural Network	25
<b>3 Text Data Extraction and Preprocessing</b>	<b>27</b>
3.1 Exploratory Analysis	29
3.2 Feature Extraction	31
3.2.1 Textual features	31
3.2.2 Time features	33
3.2.3 Check-in features	34
3.2.4 User features	34
3.2.5 Missing value imputation	35
<b>4 Image Data Extraction and Preprocessing</b>	<b>37</b>
4.1 Introduction to Images	37
4.2 Convolutional layer	38
4.3 Pooling layer	39

<b>5</b>	<b>Models and Methods</b>	<b>41</b>
5.1	Vector Space Model	41
5.1.1	TF-IDF weighting	41
5.2	Artificial Neural Networks	42
5.2.1	Backpropagation (BPP) Algorithm	44
5.3	Word2vec Algorithm	45
5.3.1	Continuous Bag of Words (CBOW) model	46
5.3.2	Skip Gram (SG) Model	46
5.4	Decision Tree based Algorithms	46
5.5	Dimensionality Reduction: Singular Value Decomposition (SVD)	48
5.6	VGG16	49
5.7	K-Fold Cross Validation (CV) Method	50
5.8	Evaluation Measure Definitions	50
<b>6</b>	<b>Results and Discussion</b>	<b>55</b>
6.1	Business dataset	55
6.1.1	Data cleaned by R	55
6.1.2	Data cleaned by Python	57
6.2	Review dataset	58
6.2.1	Non-Elite dataset	58
6.2.2	Elite dataset	60
6.3	Tip dataset	61
6.3.1	Non-Elite dataset	61
6.3.2	Elite dataset	62
6.4	Image dataset	63
6.5	Ensemble of classifiers	63
<b>7</b>	<b>Conclusions and Future Work</b>	<b>67</b>
7.1	Conclusions	67
7.2	Future work	67
	<b>Bibliography</b>	<b>70</b>

# Chapter 1

## Introduction

[Yelp.com](#) is a crowd-sourced local business review and social networking site. It is very famous because of its active user community. The site has pages devoted for businesses such as restaurants, schools, shopping outlets, automobile and sports to name a few. Yelp users (Yelpers) share their experience by submitting a review or a tip on these businesses. Now they can even share photos to give a visual of the business being reviewed. This greatly helps the people to make a spending decision. Yelp has helped the businesses to increase their revenue by 6-10% with the reviews provided by Yelpers.

Yelp hosts a [Yelp Dataset Challenge](#) where they open their data and challenge to come up with innovative data science research. Many research papers have been written from the previous rounds. This challenge is open for 6 months. Currently, they are holding the 7th round which began on 1st January 2016 and ends on 30th June 2016. This dataset is a trove for aspiring data scientists like me to work on real time data. It consists of five text files and a bunch of images which contain information about local businesses in 10 cities across 4 countries. I will discuss the dataset in detail in the further section.

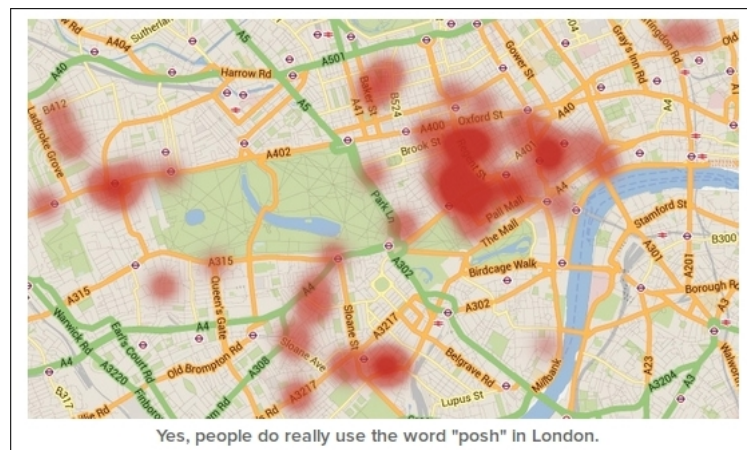


Figure 1.1: Cultural trends [1]

There are several possible insights from this rich dataset such as: Information from various cities across different countries can be used to find cultural trends (Figure 1.1) between different countries like food habits. With the help of geographical data we can analyse which businesses are succeeding in which areas. If a new business has to be set up it should be in which area so that it will have less competitors. Checkin data can be used to find busiest timings of a business.

It can also be used in recommendation systems to recommend best places to visit. Sentiment analysis, a popular machine learning technique can be utilised on the review and tip data to understand user opinion for a particular business.

## 1.1 Problem Statement

The aim of this project is to automatically classify new business into relevant categories given the business attributes, reviews, tips and images. Business categorization informs users about what a business does, determines search results and improves search relevance. For example, “Ambrosia Bakery” Figure 1.2 is categorized as “Bakeries, Sandwiches”. But many businesses and newly added businesses do not have categories but they have clues which can be used to infer categories. Text mining had always been my area of interest and with images it gave me a new field to explore and come up with a model using both.

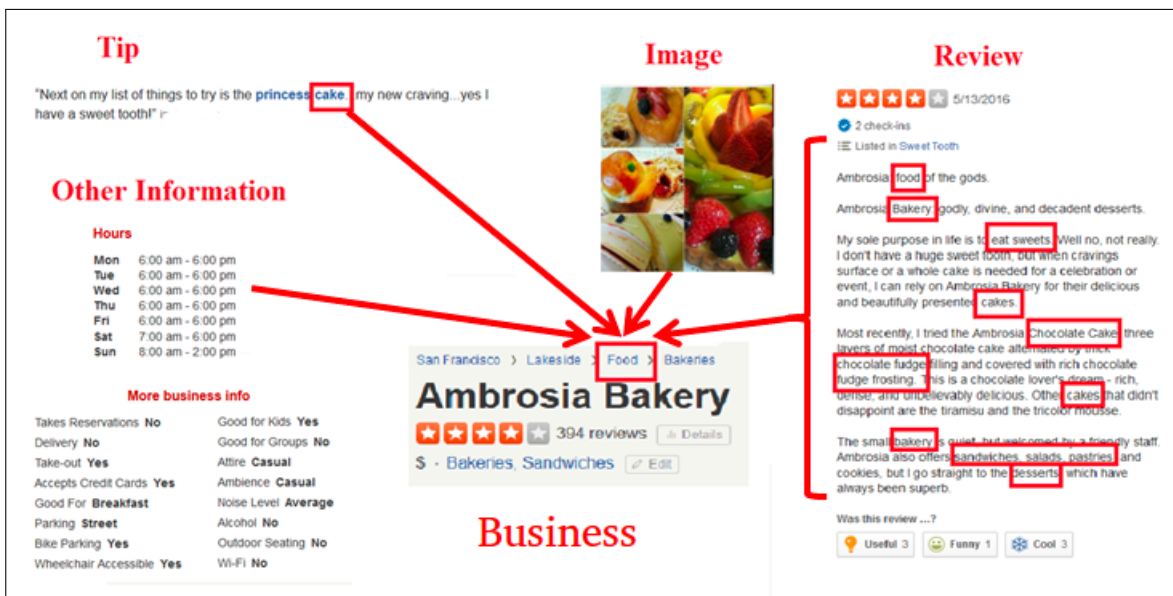


Figure 1.2: Business Categorization

## 1.2 Prior work

A lot of work has been done to utilize the data in a number of ways. As mentioned earlier, there have been very good papers that have come out of these challenges. I will talk about some of these papers and the different ways in which Yelp data has been used for interesting data analyses. There are different categories of these papers, some focus on the Yelp social network, some focus on getting utility out of the review text by topic analyses, and some other try to discover trends.

1. *Felix W.* [3] focussed on understanding the utility of social understanding systems. He studies how recommendations diffuse through the network and proposes an algorithm for social networks that will optimize the utility of an individual by making connections leading to good recommendations, and the utility of the network in effective information propagation at the same time.



2. *Hood et al.* [4] looks at a very comprehensive set of features to predict a star rating for a business. The features involve time-dependent features including number of reviews, number of days since last review; text-based features including sentiment analysis of the keywords in each category; and user-clustering features, including number of reviews the user has written.
3. *Huang et al.* [5] discovered latent subtopics from review texts using online Latent Dirichlet Allocation (LDA), and predicted star ratings for each of the latent topics. They describe that a 3 star rating for a restaurant might actually be a result of combination of 4 star rating for food, and a 2 star rating for the ambience and service.
4. *McAuley et al.* [6] find hidden factors and topics in the text to justify the rating accompanying the review.

The dataset has been included with images for the first time. None of the prior works have worked with image data. I am trying to build a system which will consider both text and image data in order to categorize a business into its relevant categories.

## 1.3 The Dataset

There are 6 separate JSON objects containing information about business, review, user, check-in, tip and photos. The data distribution is as follows:

- 2.2M reviews and 591K tips by 552K users for 77K businesses
- 566K business attributes, e.g. hours, parking availability, ambience
- Social network of 552K users for a total of 3.5M social edges
- Aggregated check-ins over time for each of the 77K businesses
- 200,000 pictures from the included businesses

### 1.3.1 Business

---

```
{
  'type': 'business',
  'business_id': (encrypted business id),
  'name': (business name),
  'neighborhoods': [(hood names)],
  'full_address': (localized address),
  'city': (city),
  'state': (state),
  'latitude': latitude,
  'longitude': longitude,
  'stars': (star rating, rounded to half-stars),
  'review_count': review count,
  'categories': [(localized category names)]
  'open': True / False (corresponds to closed, not business
    hours),
  'hours': {
    (day_of_week): {
```

```

        'open': (HH:MM),
        'close': (HH:MM)
    },
    ...
},
'attributes': {
    (attribute_name): (attribute_value),
    ...
},
}

```

---

Each business has an encrypted business id, name, geographical features, whether it is still open or shutdown, categories, average star rating given by reviewers, total review count, open and close timings and some special attributes like accepting credit cards, parking availability, wifi and so on.

### 1.3.2 Review

---

```

{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}

```

---

Each review consists of review text, star rating possibly justifying the text, and votes like useful received by the review from other users.

### 1.3.3 User

---

```

{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}

```

---

Each user has a user id, name, reviews he has given, average star rating across all reviews, his friends on Yelp, **Elite** status, number of years active on Yelp and so on. Yelp’s primary claims about their Elite user’s are:

- It states that its Elite users have high connectivity, which means that they are connected with many other users and interact often with members of their Yelp community.
- It claims that its Elite users make up the “true heart of the Yelp community”
- It claims that its users have high contribution, which means that the user has made a large impact on the site with meaningful and high-quality reviews [2]

#### 1.3.4 Check-in

---

```
{
  'type': 'checkin',
  'business_id': (encrypted business id),
  'checkin_info': {
    '0-0': (number of checkins from 00:00 to 01:00 on all
            Sundays),
    '1-0': (number of checkins from 01:00 to 02:00 on all
            Sundays),
    ...
    '14-4': (number of checkins from 14:00 to 15:00 on all
            Thursdays),
    ...
    '23-6': (number of checkins from 23:00 to 00:00 on all
            Saturdays)
  }, # if there was no checkin for a hour-day block it will not
    be in the dict
}
```

---

Each checkin gives view of all the checkins for a business for every hour of the day, for every day of the week. This gives an idea about what are the busiest times for the business.

#### 1.3.5 Tip

---

```
{
  'type': 'tip',
  'text': (tip text),
  'business_id': (encrypted business id),
  'user_id': (encrypted user id),
  'date': (date, formatted like '2012-03-14'),
  'likes': (count),
}
```

---

Tips are small but useful piece of advice that a user leaves about the business.

### 1.3.6 Photos

---

```
[
  {
    "photo_id": (encrypted photo id),
    "business_id" : (encrypted business id),
    "caption" : (the photo caption, if any),
    "label" : (the category the photo belongs to, if any)
  },
  { ... }
]
```

---



Figure 1.3: Photo linking with business example

---

```
{
  "photo_id": "-2G-S95e-6Q5l0XgRxqaZA",
  "business_id": "UUD-BLty8HGzqj5cR0o34g",
  "caption": "10 toppings few classic yogurt flavors.",
  "label": "food"
}
```

---

This is a file which links each photo in the photo archive to its corresponding business in the dataset, it also has other attributes like caption if present and label which gives the type of content if present. Figure 1.3 shows an example photo linking with its business in the dataset.

## Chapter 2

# Literature Review: Multi Label Classification Methods

Multi-label Learning is a form of supervised learning where the classification algorithm is required to learn from a set of instances, each instance can belong to multiple classes and so after be able to predict a set of class labels for a new instance. There exists a wide range of applications for multi-labelled predictions, such as text categorization, semantic image labeling, gene functionality classification etc. and the scope and interests increasing with modern applications [7].

In machine learning, single label classification is a common learning problem. Generally, the goal is to learn from a set of instances, each associated with a unique class label from a set of mutually exclusive class labels  $\mathcal{L}$ . Depending on the total number of mutually exclusive classes in  $\mathcal{L}$ , the problem can be identified as binary classification (when  $|\mathcal{L}| = 2$ ) or multi-class classification (when  $|\mathcal{L}| > 2$ ) problem. Unlike these problems, multi-label classification allows the instances to be connected with more than one class. For example, a text document that talks about scientific contributions in medical science can belong to both science and health category, genes may have multiple functionalities (e.g. diseases) causing them to be connected with multiple classes, an image that records a field and fall colored trees can belong to both field and fall foliage categories, a movie can belong to action, crime and drama categories at the same time, an email message can be associated with both work and research project. That is, the objective in multi-label classification is to learn from a set of instances where each instance can belong to one or more classes in  $\mathcal{L}$  [7].

Multi-label Learning is a form of supervised learning. The classification algorithm is required to learn from a set of instances, each instance can belong to multiple classes and be able to predict a set of class labels for a new instance. There exists a wide range of applications for multi-labelled predictions, such as text categorization, semantic image labeling, gene functionality classification etc. and the scope and interestis increasing with modern applications [7].

The key challenge of learning from multi-label data is the enormous size of output space, i.e. the number of label sets grows exponentially as the number of class labels increases. For example, for a label space with 20 class labels ( $k = 20$ ), the number of possible label sets would exceed one million (i.e.  $2^{20}$ ) [8].

**Notations** Let  $\mathcal{X}$  be an instance space,  $\mathcal{L} = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$  be a finite set of class labels. An instance (or an example)  $\mathbf{x} \in \mathcal{X}$ , represented in terms of features vector  $\mathbf{x} = (x_1, x_2, \dots, x_m)$ , is (non-deterministically) associated with a subset of labels  $L \in 2^{\mathcal{L}}$ . if we call this set  $L$  be

the set of relevant labels of  $\mathbf{x}$ , then we could call the complement  $\mathcal{L}$  to be the set of irrelevant labels of  $x$ . Lets denote the set of relevant labels  $L$  with a binary vector  $\mathbf{Y} = (y_1, y_2, \dots, y_k)$ , where  $y_i = 1 \iff \lambda_i \in L$ .  $\mathcal{Y} = \{0, 1\}^k$  is the set of all such possible labelings [7].

**Definition Multi-label Learning (MLL):**

Given a training set,  $S = (\mathbf{x}_i, \mathbf{Y}_i)$ ,  $1 \leq i \leq n$ , consisting  $n$  training instances,  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{Y}_i \in \mathcal{Y})$  i.i.d (independent and identically distributed) drawn from an unknown distribution  $D$ , the goal of the multi-label learning is to produce a multi-label classifier  $h : \mathcal{X} \rightarrow \mathcal{Y}$  (in other words,  $h : \mathcal{X} \rightarrow 2^{\mathcal{L}}$ ) that optimizes some specific evaluation function (i.e. loss function) [7].

Multi Label Classification techniques belong to two categories:

- 1) Problem transformation methods
- 2) Algorithm adaptation methods

1) Problem transformation methods: This category of algorithms attempt multi-label learning problem by transforming it into other well-know learning algorithms. Such algorithms include Binary Relevance, Ranking by Pairwise Comparison, Stacked Aggregation, Classifier Chains which transform the task of multi-label learning into the task of binary classification, Calibrated Label Ranking which transforms the task of multi-label learning into the task of label ranking, and Label Powerset, Random  $k$ -labelsets which transforms the task of multi-label learning into the task of multi-class classification [8].

2)Algorithm adaptation Methods: This category of algorithms attempt multi-label learning problem by adapting popular learning techniques to deal with multi-label data. Such algorithms include ML- $k$ NN adapting lazy learning techniques, ML-DT remodel decision tree techniques, rank-SVM adapting kernel techniques and BP-MLL remodel neural network techniques [8].

## 2.1 Problem transformation methods

There exists a variety of simple problem transformation methods which convert multi-label data in a way so that established classification algorithms (i.e. binary classifiers) can be applied. Consider an example multi-label data in Table 2.1. There are four instances belong to at least one of the four classes,  $\lambda_1, \lambda_2, \lambda_3, \lambda_4$  [7].

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

Table 2.1: Example Multi-label dataset [7]

The copy transformation method substitutes each example  $(x_i, Y_i)$  with  $|Y_i|$  examples  $(x_i, \lambda_i)$ , for each  $\lambda_j \in Y_i$ . An add-on to this is to use an weight of  $\frac{1}{|Y_i|}$  to each of these newly generated instances. This is called dubbed copy-weight method [7].

For each instance, the select family of transformation methods substitutes  $Y_i$  by one of its members. Depending on how this one member is selected, there can be various versions, such as, select-min (select least frequent), select-max (select most frequent), and select-random (randomly selected) [7].

Finally, ignore transformation just frops the multi-label instances and does the training with single label instances only [7].

Ex.	Label	Ex.	Label	Weight								
1a	$\lambda_2$	1a	$\lambda_2$	0.50	Ex.	Label	Ex.	Label	Ex.	Label	Ex.	Label
1b	$\lambda_3$	1b	$\lambda_3$	0.50								
2	$\lambda_1$	2	$\lambda_1$	1.00								
3a	$\lambda_1$	3a	$\lambda_1$	0.33								
3b	$\lambda_2$	3b	$\lambda_2$	0.33	1	$\lambda_2$	2	$\lambda_1$	2	$\lambda_1$	2	$\lambda_1$
3c	$\lambda_3$	3c	$\lambda_3$	0.33								
4a	$\lambda_2$	4a	$\lambda_2$	0.50								
4b	$\lambda_4$	4b	$\lambda_4$	0.50								
(a)		(b)			(c)		(d)		(e)		(f)	

Table 2.2: Transformed multi-label data: (a) copy (b) dubbed copy-weight (c) select-max (d) select-min (e) select-random (f) ignore [7]

Table 2.2 shows datasets generated by these approaches. But these methods is are going to have lower prediction performance because of the loss of instances [7].

Instance	Label Set
1	$\lambda_{2,3}$
2	$\lambda_1$
3	$\lambda_{1,2,3}$
4	$\lambda_{2,4}$

Table 2.3: Transformed data using Label Powerset method [7]

### 2.1.1 Label Powerset (LP)

Label Powerset (LP) is an effortless method that considers each unique set of labels in a multi-label training data as one class in the new transformed data. Therefore, the new transformed problem is a single label multi-class classification task. For a new instance, LP predicts the most probable class which actually is a collection of classes in the original data. It is also possible to produce a ranking of labels using LP, given the classifier can predict a probability distribution over all newly formed classes. Table 2.3 shows the dataset transformed using LP method. Notice that the computational complexity of LP is upper bounded by  $\min(n, 2^k)$ , where  $n$  is the total number of data instances and  $k$  is the total number of classes in the training data (before transformation). In practice complexity would be much less than  $2^k$ , still for large values of  $n$  and  $k$  this can be an issue. The other problem with this approach is that, a large number of classes (set of a labels in the original data) would be associated with very few examples and that would also cause extreme class imbalance problem for learning [7].

The Pruned Problem Transformation method addresses the second problem above by pruning away the label sets that occur less than a user-defined threshold and often replacing them by introducing disjoint subsets of these labels sets that are more frequent (greater than the threshold) in the data.

### 2.1.2 Binary Relevance (BR)

Binary Relevance (BR) is one of the most popular approaches as a problem transformation method that actually creates  $k$  datasets ( $k = \mathcal{L}$ , total number of classes), each for one class label and trains a classifier on each of these datasets. Each of these datasets contains the same number of instances as the original data, but each dataset  $D_{\lambda_j}, 1 \leq j \leq k$  positively labels

instances that belong to class  $\lambda_j$  and negative otherwise. Table 2.4 shows the example dataset transformed for BR.

Ex.	Label	Ex.	Label	Ex.	Label	Ex.	Label
1	$\neg\lambda_1$	1	$\lambda_2$	1	$\lambda_3$	1	$\neg\lambda_4$
2	$\lambda_1$	2	$\neg\lambda_2$	2	$\neg\lambda_3$	2	$\neg\lambda_4$
3	$\lambda_1$	3	$\lambda_2$	3	$\lambda_3$	3	$\neg\lambda_4$
4	$\neg\lambda_1$	4	$\lambda_2$	4	$\neg\lambda_3$	4	$\lambda_4$

(a)                      (b)                      (c)                      (d)

Table 2.4: Transformed data sets produced by Binary Relevance (BR) method [7]

Once these datasets are ready, it is easy to train one binary classifier for each. For any new instance, BR outputs the union of the labels  $\lambda_j$  that are positively predicted by the  $k$  classifiers. While BR has been used in many practical applications, it has been widely criticized for its indirect assumption of label independence which might not hold in the data [7].

### 2.1.3 Ranking by Pairwise Comparison (RPC)

Ranking by Pairwise Comparison (RPC) transforms the multi-label dataset into  $\binom{k}{2}$  binary label datasets, one for each pair of labels,  $\lambda_i, \lambda_j$ ,  $1 \leq i < j \leq k$ . Each dataset retains the instances from the original dataset that belong to atleast one of the corresponding labels but not both (Table 2.5). A binary classifier is then trained on each of these datasets. Also, given a new instances, it is easy to obtain a ranking of labels by first invoking all these binary classifiers and then counting their votes for each label [7].

Ex.	Label	Ex.	Label	Ex.	Label	Ex.	Label	Ex.	Label	Ex.	Label
1	$\lambda_{\neg 1,2}$	1	$\lambda_{\neg 1,3}$	2	$\lambda_{1,\neg 4}$	4	$\lambda_{2,\neg 3}$	1	$\lambda_{2,\neg 4}$	3	$\lambda_{3,\neg 4}$
2	$\lambda_{1,\neg 2}$	2	$\lambda_{1,\neg 3}$	3	$\lambda_{1,\neg 4}$			3	$\lambda_{2,\neg 4}$	4	$\lambda_{\neg 3,4}$
4	$\lambda_{\neg 1,2}$	4	$\lambda_{\neg 1,\neg 3}$	4	$\lambda_{\neg 1,4}$						

(a)                      (b)                      (c)                      (d)                      (e)                      (f)

Table 2.5: Datasets transformed by RPC method [7]

### 2.1.4 Calibrated Label Ranking (CLR)

Calibrated Label Ranking (CLR) is an extension of RPC, introducing an additional label to the original label set, which can be interpreted as a “neutral breaking point” (often called calibration label,  $\lambda_0$  and can be thought as a split point between relevant and irrelevant labels. Thus a calibrated ranking,

$$\lambda_{i1} \succ \lambda_{i2} \dots \dots \succ \lambda_{ij} \succ \lambda_0 \succ \lambda_{ij+1} \succ \dots \dots \succ \lambda_{ik}$$

clearly is a ranking of the labels (ignore the calibration label  $\lambda_0$ ) and at the same time creates a bipartition of relevant ( $\lambda_{i1} \dots \dots \lambda_{ij}$ ) and irrelevant ( $\lambda_{ij+1} \dots \dots \lambda_{ik}$ ) labels. Each example that is annotated with a particular label, clearly is a positive example for that label and is treated as a negative example for the calibration label. Each example that is not annotated with a label is clearly a negative example for that label and is treated as a positive example for the calibration label. Thus a Binary Relevance (BR) classifier can then be employed to discriminate between the calibrated label and each of the other labels. Intuitively, while applied to the dataset in



Table 2.1, CLR would work on both the data in Table 2.4 and Table 2.5, and the latter one is for the calibration label [7].

## 2.2 Algorithm adaptation methods

Clare *et. al* in [9] uses C4.5 algorithm for multi-label data with the modified entropy definition:

$$Entropy = - \sum_{j=1}^k \{ (P(\lambda_j) \log P(\lambda_j) + (1 - P(\lambda_j)) \log(1 - P(\lambda_j))) \}$$

where,  $P(\lambda_j)$  = probability of class  $\lambda_j$ . This allows to estimate the uncertainty in terms of number of bits in multi-label setting. This modified method also allows multiple labels at the leaves [7].

### 2.2.1 Tree Based Boosting

AdaBoost.MH and AdaBoost.MR are two simple extensions of AdaBoost for multi-label data where the first tries to minimize hamming loss and the second tries to find a hypothesis with optimal ranking. In AdaBoost.MH, examples are presented as example-label pairs and in each iterations increases the weights of misclassified example-label pairs. In contrast, AdaBoost.MR works on pairs of labels for any instance and in each iteration increases the weights of the example with misordered label pairs. With a goal to produce better human readable classification rules (from trees) [7].

### 2.2.2 Lazy Learning

ML- $k$ NN (i.e.  $k$  Nearest Neighborhood ( $k$ NN)) is based on BR, but to find the label set for a given test instance it uses maximum a posteriori (MAP), based on prior and posterior probabilities of for each  $k$  nearest neighbor label. For each instance, ML- $k$ NN first finds its  $k$  nearest neighbors in the training set and then formulates the problem as a MAP problem below:

$$Z_t^\lambda = \underset{b \in \{0,1\}}{\operatorname{argmax}} P(H_b^\lambda | E_{C_t(\lambda)}^\lambda), \quad l \in \mathcal{L}$$

where  $Z_t^\lambda$  = predicted label set for instance  $t$ ,  $H_1^\lambda(t)$  and  $H_0^\lambda(t)$  be the events that  $t$  has label  $\lambda$  and does not have label  $\lambda$  respectively.  $E_i^\lambda$  is the event that exactly  $j$  instances of  $k$  nearest neighbor of test instance  $t$  has label  $\lambda$  and,  $C_t^\lambda$  is the membership vector that counts the number of neighbors of  $t$  belonging to class  $\lambda$ . This can be re-written as:

$$Z_t^\lambda = \underset{b \in \{0,1\}}{\operatorname{argmax}} P(H_b^\lambda) P(E_{C_t(\lambda)}^\lambda | H_b^\lambda), \quad l \in \mathcal{L}$$

and can be directly estimated from the training data [7].

### 2.2.3 Neural Network

There exists neural networks with multi-layer perceptron based algorithms been extended for multi-label data. In BP-MLL which will be discussed in detail in Section 5.2, the error function for the very common neural network learning algorithm, back-propagation has been modified to account for multi-label data. Multi-layer perceptron is easy to extend for multi label data where one output node is maintained for each class label [7].



## Chapter 3

# Text Data Extraction and Preprocessing

The data analysis was done initially in R and later in `Python` software tools.

Dataset was in the form of JSON objects containing business, review, user, checkin and tip information. The JSON objects were large in size (hundreds of megabytes to gigabytes) and loading each time in R was difficult. These JSON objects were in the nested list format which needed to be flattened out. The nested JSON objects were flattened using `jsonlite` package in R. The flattened data was stored in the MySQL database. Data could then be easily retrieved in R using the `RMySQL` package in the required dataframe format.

There are a total of 77,445 unique businesses. To these businesses 22,25,213 reviews, 5,91,864 tips and 55,569 check-ins are provided by 5,52,339 different users.

Business Name	Business Category
Mr Hoagie	Fast Food, Restaurants
Clancy's Pub	Nightlife
Joe Cislo's Auto	Auto Repair, Automotive
Cool Springs Golf Center	Active Life, Mini Golf, Golf
Verizon	Shopping, Home Services, Internet Service Providers, Mobile Phones, Professional Services, Electronics
Greentree Animal Clinic	Veterinarians, Pets
Kings Family Restaurant	Burgers, Breakfast & Brunch, American (Traditional), Restaurants

Table 3.1: Example business dataset

Business category inference is based on supervised learning approach where we are trying to predict the “categories” a business can belong to. There were 260 businesses which did not have a category. All entries (business, review, tip, checkin and images) pertaining to these businesses were dropped from training set.

First task was to find all the possible categories in the training set that a business can belong to. There are in all 896 unique business categories. Each business belonged to atleast 1 and atmost 11 categories from these (Table 3.1 shows an illustrative example). On an average there are 3 categories per business. So this is a multi-label classification problem. Figure 3.1 shows the distribution of top 45 categories.

Restaurants	Shopping	Food
25071	11233	9250
Beauty & Spas	Health & Medical	Nightlife
6583	5121	5088
Home Services	Bars	Automotive
4785	4328	4208
Local Services	Active Life	Fashion
3468	3103	3078
Event Planning & Services	Fast Food	Pizza
2975	2851	2657
Mexican	Hotels & Travel	American (Traditional)
2515	2495	2416
Sandwiches	Arts & Entertainment	Coffee & Tea
2364	2271	2199
Hair Salons	Italian	Burgers
2091	1848	1774
Auto Repair	Doctors	Nail Salons
1716	1694	1667
Chinese	American (New)	Home & Garden
1629	1593	1586
Pets	Fitness & Instruction	Hotels
1497	1442	1431
Grocery	Real Estate	Breakfast & Brunch
1424	1424	1369
Dentists	Specialty Food	Women's Clothing
1195	1150	1138
Bakeries	Professional Services	Ice Cream & Frozen Yogurt
1115	1025	1018
Cafes	Financial Services	Pubs
1002	875	874

Figure 3.1: Top 45 categories

These categories were a large number to get started with. So decided to identify a smaller subset of categories. On doing a little research identified 22 categories to which a business can primarily belong to (Figure 3.2). Again it can belong to multiple categories from these 18 categories. It is still a multi-label classification problem.

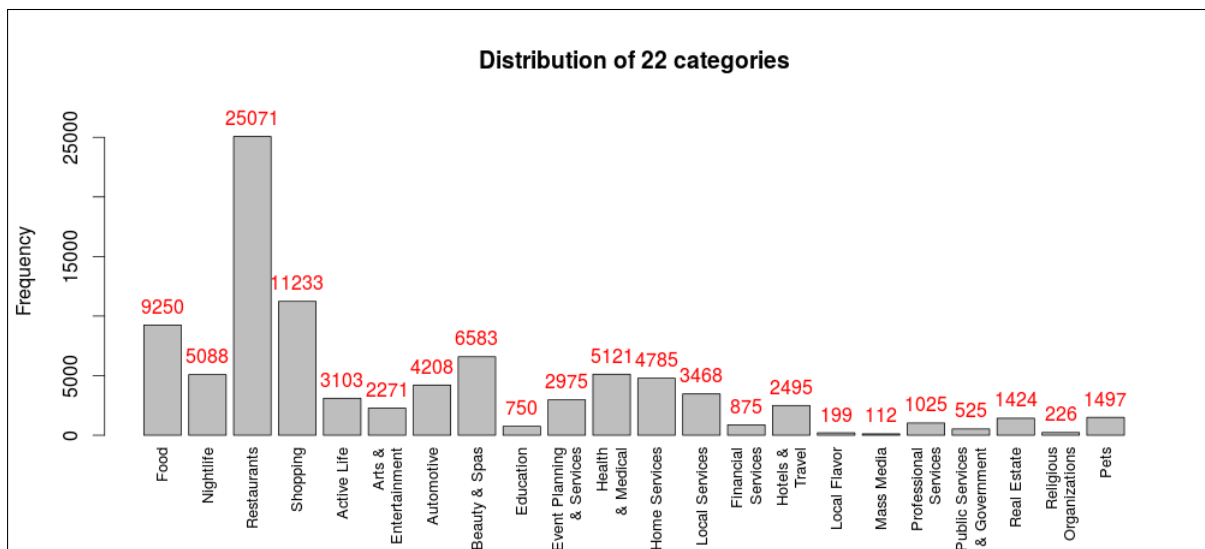


Figure 3.2: Primary 22 categories

As we can see from Figure 3.2, Food has 9,250 businesses and Shopping has 11,233 businesses which have nearly equal business distribution. There are examples belonging to either one or both the categories. So for building multi-label classification model these two categories were

chosen as candidates.

The distribution of businesses in these 2 categories is as below:

There are total 19,744 businesses

- 10,524 - Shopping
- 8,541 - Food
- 709 - Both

This dataset consists of businesses which are “open” and which have been “closed” now. A business can be shutdown due to a number of reasons. So in order to remove the influence of shutdown businesses, businesses which had “open” status as false were removed. Therefore on removal of closed businesses, there are total 16,864 businesses and the distribution now becomes:

- 9,238 - Shopping
- 6,978 - Food
- 648 - Both

The review, tip and check-in count for these businesses was 3,40,062 , 1,08,910 and 13,571 respectively. Review and tip counts were large for R to handle. So reduction in number of samples was done. For every business having reviews and tips greater than 5, any 5 reviews and tips out of them were selected. So per business there were maximum 5 reviews and 5 tips. Revised review, tip and check-in counts are:

- Review count: 73,576
- Tip count: 36,517
- Check-in count: 13,571

### 3.1 Exploratory Analysis

The distribution of businesses, reviews, tips and check-ins in the 2 categories is given in Table 3.2. For this initial Exploratory analysis, businesses which had both “Food” and “Shopping” as categories were added in both the categories (eg. “Shopping” = 9,238 + 648 = 9,886).

	Shopping	Food
Business	9,886	7,626
Review	42,274	34,225
Tip	16,561	21,811
Check-in	7,403	6,774

Table 3.2: Data distribution across “Shopping” and “Food” categories

Each business has a “star rating” associated with itself given by the reviewers suggesting its quality. Star rating is given on a scale of 1 to 5, 1 - low quality and 5 - high quality. The distribution of businesses on the basis of star rating is shown in Figure 3.3. The average star rating for “Shopping” is 3.70 and for “Food” is 3.75.

Most businesses have entries related to their open and close timings. For example consider the businesses “Starbucks” and “Minerva Bakery”. Their time information is shown in Table 3.3

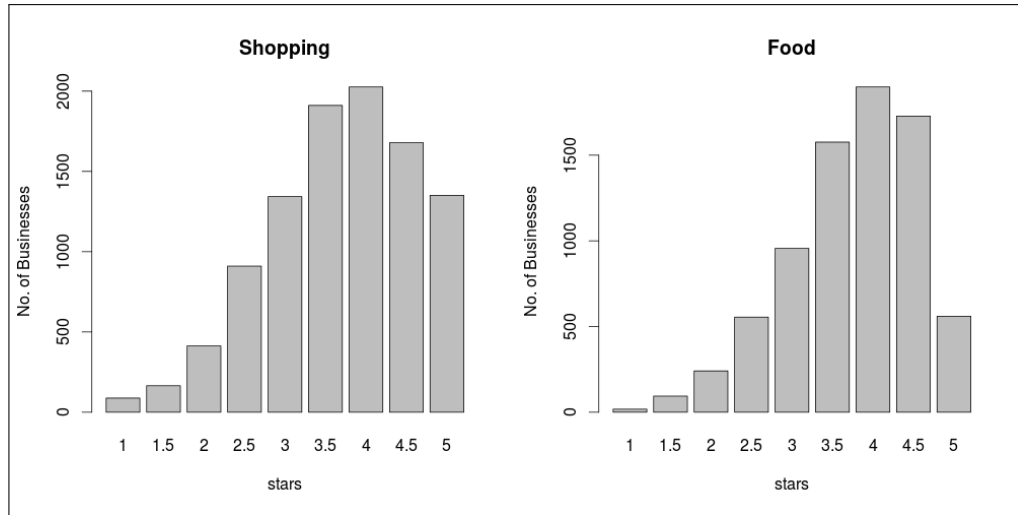


Figure 3.3: Business distribution as per Star rating

Day	Open	Close
Monday	06:00	21:30
Tuesday	06:00	21:30
Wednesday	06:00	21:30
Thursday	06:00	22:30
Friday	07:00	22:30
Saturday	07:00	21:00
Sunday	06:00	21:30

Table 3.3: Time data for “Starbucks”

and Table 3.4 respectively.

“Starbucks” is open throughout the week whereas “Minerva Bakery” is open only 5 days a week. I introduced a feature **nOpenDays** to capture the number of days a business is open throughout the week. Businesses which had all time entries as NA (Not Available) were considered as missing totally, whereas those which had partial NA (like “Minerva Bakery”) were considered to be closed on those particular days. Figure 3.4 shows the distribution of businesses as per their **nOpenDays**.

Almost all the businesses in “Shopping” and “Food” categories are open throughout the week

Day	Open	Close
Monday	NA	NA
Tuesday	07:00	16:30
Wednesday	07:00	16:30
Thursday	07:00	16:30
Friday	07:00	16:30
Saturday	07:00	15:30
Sunday	NA	NA

Table 3.4: Time data for “Minerva Bakery”

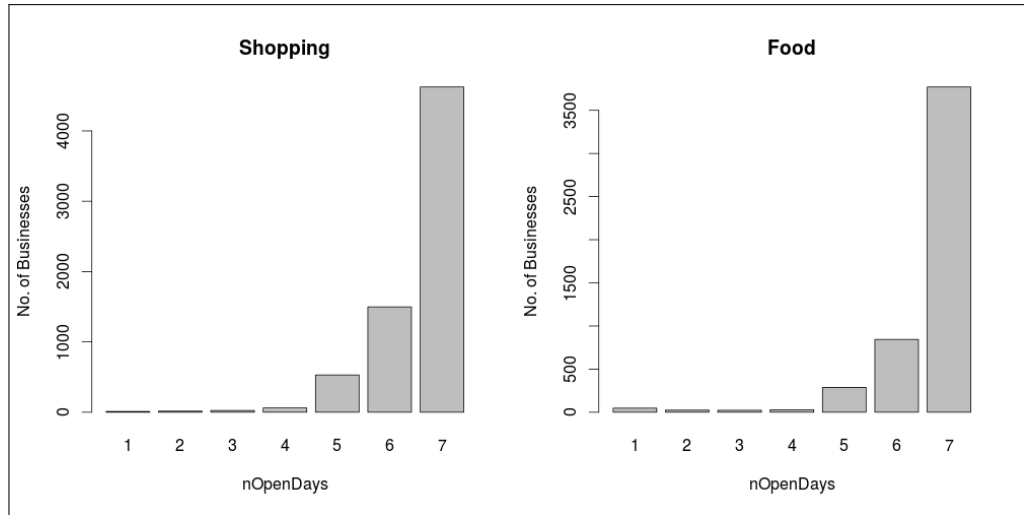


Figure 3.4: Business distribution as per `nOpenDays`

(`nOpenDays` =7). There are some businesses which do not have the time information, they have been excluded for the analysis of this feature.

## 3.2 Feature Extraction

### 3.2.1 Textual features

#### Data cleaning

Data cleaning was initially done in R [27] and later in Python [28] because of the powerful features offered by Natural Language Toolkit (NLTK)[29] package in Python.

#### Using R

1. Data cleaning was accomplished using the `tm` package. The text was cleaned by converting to lowercase, removing numbers, removing punctuations, removing common English “stopwords” (the, is, was etc.).
2. Next “stemming” was performed which truncates words to their specific stems(e.g., “compute”, “computes” and “computing” all become “comput”). However, these stems need to be identified first.
3. `SnowballC` package allows to identify such stem elements which uses the Porter stemmer algorithm. This package has to be loaded with `tm` package to perform stemming effectively using the `tm_map()` function in `tm` package.

#### Using Python

1. For text cleaning, functions in NLTK package were used. There were foreign accented words like café which were converted to nearest possible English alphabets. This was done by decoding the text in ISO-8859-1 format to UTF-8 (Unicode 8 bit) and then to simple ASCII character set (e.g: café was converted to cafe).

2. Text was converted to lowercase, numbers, punctuation and English stopwords were removed as before.
3. Additional stopwords ('http', 'www', 'net', 'com', 'jpg', 'etc', 'like', 'likes', 'still', 'definitely', 'always', 'feel', 'feels', 'difficult', 'different', 'usual', 'usually') were added to the existing list.
4. Misspelled words were checked and corrected using **pyEnchant** package and specially written function for word correction.
5. Lemmatisation was done to get the correct stem of the word. It gives a real word rather than just stemming. Consider the following examples  
 eg 1: The word "better" has "good" as its lemma. This link is missed by stemming, as it requires a dictionary look-up.  
 eg 2: The word "walk" is the base form for word "walking", and hence this is matched in both stemming and lemmatisation.
6. Parts-of-Speech (POS) Tagging was done to further remove noisy words. This was done using `nltk.pos_tag()` using the default `maxent_treebank_pos_tagger` tagger. It is a Maximum Entropy tagger which has been trained on **Penn Treebank tagset**. Only the noun, adjective and verb tags were retained.
7. Stemming was performed as above.

eg 1: **Raw review:** Pretty darn good little café at the Spa at Camelback. Most of the fare is quite healthy (except their extensive selection of cocktails) and very fresh and tasty. I've eaten here a couple of times and on my most recent visit had a very good grilled chicken Caesar spinach wrap with vegetable chips. Good flavor, especially in the chicken where you could really taste fresh herbs, and with the chips. They're still basically a salt delivery vehicle, but marginally better for you than the potato variety. There are a few tables and then a bar with comfy stools which looks directly out over the length of the spa pool and the mountains beyond. The service here isn't quiiiiiiite as polished as at other outlets in the resort, but it's still fine.

**Cleaned review:** darn good littl cafe spa camel fare healthi extens select cocktail fresh tasti eat coupl time recent visit good grill chicken caesar spinach wrap veget chip good flavor chicken tast fresh herb chip salt deliveri vehicl potato varieti tabl bar comfi stool look length spa pool mountain servic quietud polish outlet resort fine

The cleaned review has all accents removed, and only quality words are retained.

eg 2: **Raw review:** Awesome samosas and kachoris!! We ordered roti and sabji as well. Chapati and matar paneer was ok not so good. In all great for snacks :)

**Clean review:** awesom samovar chorist order roi saber apathet maker paper ok good great snack

Here, it has missed the Indian dishes and corrected them unnecessarily.

### N-gram features

An N-gram is a continuous sequence of N items that occur together from a given text sequence. Unigrams means a set of one word, bigrams means a set of two, trigrams means a set of three words that occur together and so on. The N-grams generated are used in building the Vector



Space Model which will be discussed in Section 5.1.

Consider a sentence: “a man is walking”

Unigrams from this sentence:

- a
- man
- is
- walking

Bigrams from this sentence:

- a man
- man is
- is walking

If  $X$  = Num of words in a given sentence  $K$ , the number of  $N$ -grams for sentence  $K$  would be:

$$\text{N-gram}_K = X - (N - 1)$$

### Word2vec features

Word vectors were obtained from Word2vec algorithm in Python’s `gensim` [30] package. The description and working of the model is explained in Section 5.3.

#### 3.2.2 Time features

As we saw in Section 3.1, we have the “open” and “close” timings of the businesses. There were 14 features giving “open” and “close” timings of the businesses throughout the week. 11 features were created from this data, based on time and based on days.

Features by time: I binned the time data in 4 bins indicating the time-slots in which the business is open

- Morning : 05:00 - 12:00
- Afternoon : 12:00 - 17:00
- Evening: 17:00 - 21:00
- Night: 21:00 - 05:00

Features by days are the 7 days of the week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday which indicate if the business is open on that particular day.

Consider the same example of “Starbucks” and “Minerva Bakery” in Section 3.1. The time features generated for these 2 businesses by above description are shown in Table 3.5 and Table 3.6 respectively.

Business	Morning	Afternoon	Evening	Night
Starbucks	1	1	1	1
Minerva	1	1	0	0
Bakery				

Table 3.5: Features by Time

Business	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Starbucks	1	1	1	1	1	1	1
Minerva	0	1	1	1	1	1	0
Bakery							

Table 3.6: Features by Days

### 3.2.3 Check-in features

As seen in the Check-in file JSON object, there are check-in given for every hour for every day of the week. These account for a total of 168 features ( $24 \times 7 = 168$ ). I generated 24 features (1 feature for every hour of the day) from these 168 by taking a sum of the check-in over that particular hour for the entire week. Consider the same example of “Starbucks” as in Section 3.1. For illustration purposes, consider check-in data for 3 hours out of 24 for all 7 days of this business. Table shows the check-in data for the 3 hours.

The 3 hours considered are from “9-10”, “10-11” and “11-12” in a 24 hour clock cycle.

Day	“9-10”	“10-11”	“11-12”
Monday	1	2	3
Tuesday	1	1	1
Wednesday	4	1	3
Thursday	4	2	1
Friday	4	4	4
Saturday	NA	2	3
Sunday	2	2	7

Table 3.7: Check-in dataset example

Businesses which had all check-in entries as NA were considered as missing totally, whereas those which had partial NA (like “Starbucks”) were considered to have “no check-in” on those particular timings and assumed as zero for calculation purposes (eg: In Table 3.7 check-in between “9-10” on Saturday is assumed as zero). The 3 new features generated will have names “9”, “10” and “11” respectively as shown in Table 3.8.

	“9”	“10”	“11”
“Starbucks”	16	14	22

Table 3.8: Check-in dataset example

### 3.2.4 User features

In the User dataset, Yelp gives the number of years a user had “Elite” status. As discussed in Section 1.3.3, an “Elite” user serves as a local expert. So more the number of years he is Elite

he can be regarded as trusted Yelper. A feature **nElite** was created to measure the eliteness of the user as in Table 3.9.

User	Elite years	nElite
1	2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015	11
2	-	0
3	2005, 2006, 2007, 2008, 2010, 2011, 2012	7

Table 3.9: User dataset example

### 3.2.5 Missing value imputation

There was a considerable amount of missing information in the Business data which needed to be imputed so that their information loss could be avoided. These included the special attribute features such as (credit card acceptance, wi-fi, parking availability etc.), time features and check-in features.

#### Special attribute imputation

There are 71 special attribute features which had missing information. But they could give clues to the category inference problem. So imputing these missing values was carried out.

A simple analysis was carried out to find the percentage of missing information in these features. 63 features had missing information greater than 70%. Table 3.9 shows the attributes which had missing information less than 70% in the increasing order of missing content.

Attribute	Missing Information (%)
Accepts Credit Cards	06.469
Price Range	06.908
Parking Garage	12.885
Parking Valet	12.945
Parking Street	12.951
Parking Lot	12.951
Parking Validated	13.722
Wheelchair Accessible	62.951

Table 3.10: Special attribute missing information

Features having missing values greater than 30% were completely ignored and removed from the dataset. Missing value were imputed for other features. Referring to Table 3.10 the top 7 features were imputed.

Missing value imputation was carried out by machine learning procedure:

1. One attributed is imputed each time.
2. Dataset is divided in two sections training and testing. Training set includes the examples without the missing value attribute. Testing set includes examples with the missing value attribute.
3. Classifier is trained on the training set with output as the attribute that needs imputation.

4. Missing value is predicted for the testing set.
5. Predicted value is imputed to the missing field in the original data.
6. Process is repeated for all the attributes having missing values.

The order of predicting missing value attributes was starting with the attribute having least missing information (“Accepts Credit Cards”) to the attribute having the most missing information (“Parking Validated”). The algorithm used for predicting missing values was C50 which will be discussed in the later section.

### **Time feature imputation**

Examples having all time entries as NA were treated as missing data. There were 32.93% such entries which were imputed by the same process as described in the Section 3.2.5.

### **Check-in feature imputation**

Examples having all check-in entries as NA were treated as missing data. All 168 check-in features had missing values in different proportions. 166 features had missing data greater than 50%. All check-in features were imputed. Imputation was carried out by taking mean of the check-ins of the businesses with same “star” rating with no missing values for that attribute.

## Chapter 4

# Image Data Extraction and Preprocessing

As discussed in Section 1.3.6, there is an image archive and a JSON object linking every image in the archive to its respective business. The image archive had 2,00,000 total images. Using a `shell` script 37,921 images were filtered belonging to “Food” and “Shopping” business categories. This data was huge to process on a 8GB CPU system, so only 1 image per unique business was decided. This brought down the total images belonging to these categories to 9,026. The distribution of these images in the two categories is as follows:

- 4,642 - Shopping
- 4,707 - Food
- 323 - Both

### 4.1 Introduction to Images

A digital image is a 2-D array made up pixel(picture element) values. The specific color that a pixel shows is a mixture of three channels of the color spectrum - Red Green Blue (RGB). Each pixel is an intensity value of the light RGB channel space. In color images there are 3 such 2-D arrays, one for each channel.

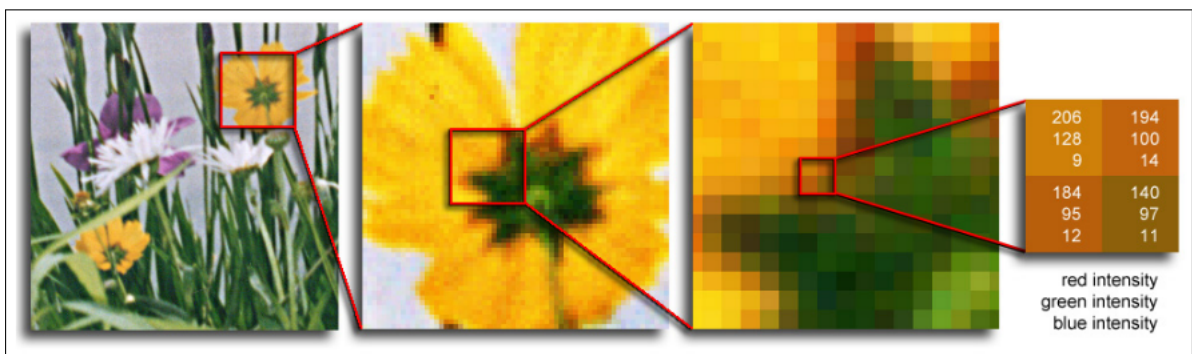


Figure 4.1: Image basics

In order to extract features from the image data, every image was resized to  $224 \times 224$  pixels. This was done using “OpenCV” package `cv2` for Python.



Figure 4.2: Image resizing

All image analyses was done in `Python`. The separate RGB channels were extracted from the images and stored in a “numpy” array. The resized images were passed through the pretrained VGG16 model which will be discussed in Section 5.6 to extract 4096 features which were used for the classification.

To check the effect of normalization, all images were normalized by the mean RGB values (123.68, 116.779, 103.939). Models were built on both plain and normalized images to check for performance.

12	123	55	201	111	27	15
14	12	18	79	15	21	125
213	1	88	200	18	245	21
26	111	54	10	87	244	202
174	14	34	100	139	99	56
127	189	123	8	17	111	49
199	11	20	11	12	145	85

1	1	1
1	0	3
0	1	1

Figure 4.3: Example of a digital image and  $3 \times 3$  convolution matrix (kernel) [10]

## 4.2 Convolutional layer

Convolution is done for emphasizing some of the image features. For doing so, a digital image and a convolution matrix is needed (Figure 4.3). The convolution matrix is also called filter or kernel. A kernel is like a sliding window that moves across the original image. It moves from left to right and top to bottom direction. At every shift, it multiplies the overlapping entries

of the input image and kernel, and does a summation. The result is the convolved value of the image (Figure 4.4). The values obtained from all such movements of the kernels above the image are inserted in a new image [10].

There is another parameter called stride  $S$  which is used to stride the kernel on the input image. If  $S = 1$ , the kernels in Convolutional Neural Networks (CNN) will move by one pixel from left to right and from top to bottom. We can decide the number of kernels  $K$  in convolutional layer. Zero padding is also one algorithm parameter in CNN which pads the zeros around the corner of the images.

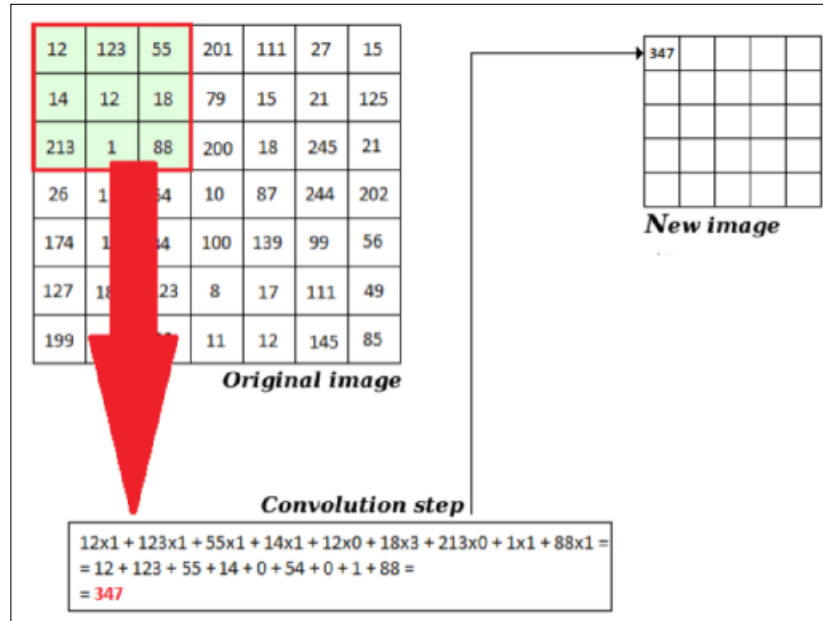


Figure 4.4: Convolution process with  $3 \times 3$  convolution matrix (kernel) [10]

### 4.3 Pooling layer

Pooling is done to reduce the dimensionality of images. Pooling converts a section covered by the pooling kernel into one value like mean or max. Max pooling is normally done where we select the maximum value from the part covered by kernel (Figure 4.5).

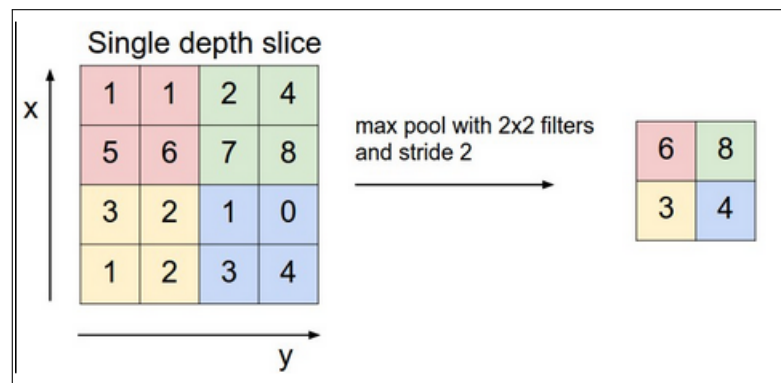


Figure 4.5: Pooling process [11]





## Chapter 5

# Models and Methods

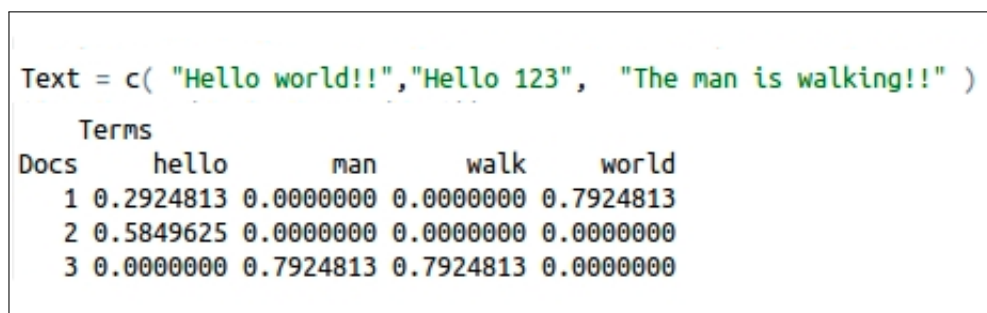
**Supervised Learning Technique:** In supervised learning, inputs (features) as well as the desired outputs are supplied to the learning algorithm.

In this work, all supervised learning models have been used.

### 5.1 Vector Space Model

This is the most “popular” model used in text analysis. It is used for representing text documents as vectors. In this model, a corpus is created from the training data. This corpus is nothing but a collection of words extracted from all the documents. A Document Term Matrix (DTM) is created from this corpus. It is a matrix that describes the frequency of terms that occur in a collection of documents. In a DTM, rows correspond to documents in the collection and columns correspond to terms (Figure 5.1).

The Term Frequencies (TF) and Inverse Document Frequencies (IDF) of the unigrams and bigrams is computed to build a Document Term Matrix (DTM) with TF-IDF weighting which is used in the Vector Space Model.



```
Text = c( "Hello world!!", "Hello 123", "The man is walking!!" )  
Terms  
Docs    hello    man    walk    world  
1 0.2924813 0.0000000 0.0000000 0.7924813  
2 0.5849625 0.0000000 0.0000000 0.0000000  
3 0.0000000 0.7924813 0.7924813 0.0000000
```

Figure 5.1: DTM example with TF-IDF weighting

#### 5.1.1 TF-IDF weighting

It is a way to score the importance of words (or “terms”) in a document based on how frequently they appear across multiple documents.

- If a word appears frequently in a document, it’s important. Give the word a high score.

- But if a word appears in many documents, it's not a unique identifier. Give the word a low score.

Therefore, common words like “the” and “for”, which appear in many documents, will be scaled down. Words that appear frequently in a single document will be scaled up [12].

Mathematically,

**Term Frequency:**  $\text{TF}(t, d)$  : No. of times term  $t$  occurs in document  $d$

$$\text{TF}(t, d) = 1 \text{ if } t \text{ occurs in } d \text{ and } 0 \text{ otherwise}$$

**Inverse Document Frequency:**  $\text{IDF}(t, d)$  : Whether the term  $t$  is common or rare across all documents

$$\text{IDF}(t, d) = \log_2 \frac{N}{|\{d \in D : t \in d\}|}$$

where

$N = |D|$  total no. of documents  $|\{d \in D : t \in d\}|$  is no. of documents where term  $t$  appears

A high IDF score implies it is a rare term.

TF-IDF score is simply the product of TF and IDF. Ideally, a term is important to a document if it has high TF and high IDF scores [12]. Figure 5.1 shows an example DTM with TFIDF weighting.

## 5.2 Artificial Neural Networks

Human Brain is composed of interconnected electro-chemical transmitting neurons which forms a big intense network. Biological neural networks are just one of many possible solutions to the problem of processing information. The main difference between neural networks and conventional computer systems is the massive parallelism and redundancy which they exploit in order to deal with the unreliability of the individual computing units. Moreover, biological neural networks are self-organizing systems and each individual neuron is also a delicate self-organizing structure capable of processing information in many different ways [17].

ANN is a family of statistical learning models inspired from biological neural network. It is a massively parallel distributed processor made up of simple processing units that has a natural tendency for storing experimental knowledge and making it available for use [17].

It resembles the brain in two respects [19]:

1. Knowledge is acquired by the network through a learning process
2. Interconnection strengths known as synaptic weights are used to store the knowledge

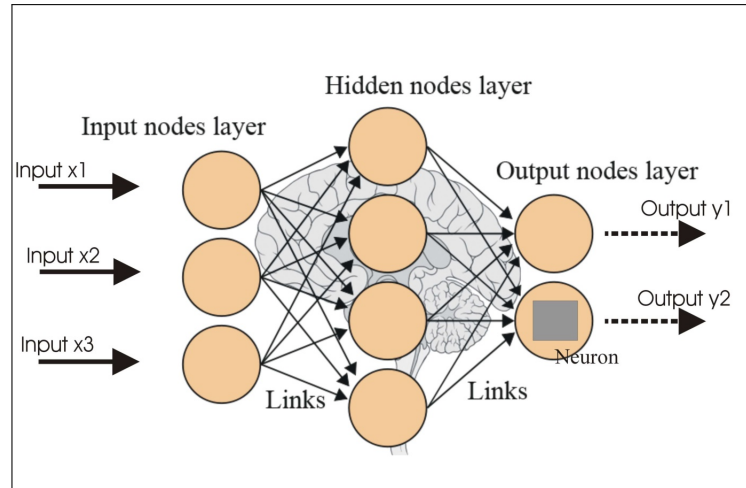


Figure 5.2: ANN representation [20]

An artificial neuron is a computational model inspired from the natural neurons. Natural neurons receive signals through synapses located on the dendrites or membrane of the neuron. When the signals received are strong enough (exceeds a certain threshold), the neuron is activated and emits a signal through the axon. This signal might be sent to another synapse, and might activate other neurons. The complexity of real neurons is highly abstracted when modelling artificial neurons [18].

These basically consist of inputs (like synapses), which are multiplied by weights (strength of the respective signals), and then computed by a mathematical function which determines the activation of the neuron. Another function computes the output of the artificial neuron. ANNs combine artificial neurons in order to process information [18].

The output of a neuron is a function of the weighted sum of the inputs plus a bias. The function of the entire neural network is simply the computation of the outputs of all the neurons. Activation function is applied to the weighted sum of the inputs of a neuron to produce the output. Majority of NNs use sigmoid functions due to their smooth, continuous, and monotonically increasing behaviour (derivative is always positive). Sigmoid functions have fixed range of working but they never reach maximum or minimum. The most common sigmoid function used is the logistic function (Figure 5.3). The calculation of derivatives are important for neural networks and the logistic function has a very nice derivative [18].

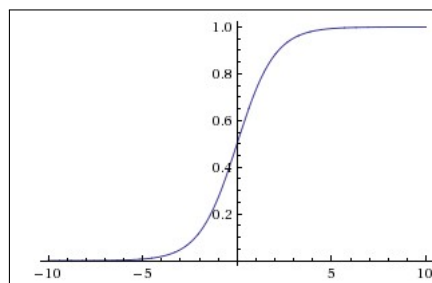


Figure 5.3: Sigmoid function [21]

### 5.2.1 Backpropagation (BPP) Algorithm

It involves 2 phases, Forward phase and Backward phase. Initially, the parameters are given very small random values. The input signal is forward propagated through the network layer by layer till it reaches the output. The forward phase finishes with the computation of an error function. During the backward phase the error signal is propagated through the network in the backward direction to the previous layer. It is during this phase that adjustments are applied to the parameters so as to minimize the error. Error minimization takes place through the Gradient Descent optimization method.

The 4 main steps of the BPP algorithm are:

1. Forward propagate error signals to output
2. Calculate output error  $E$ , and backpropagate error signal
3. Use forward signal and backward signals to calculate parameter gradients
4. Update network parameters

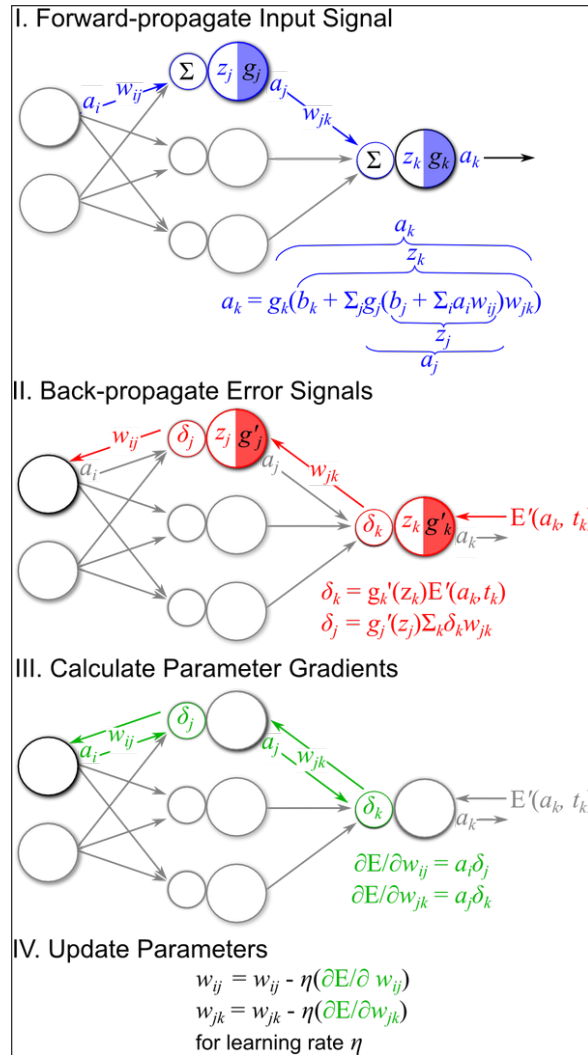


Figure 5.4: BPP algorithm [13]

Figure 5.4 shows the working of the forward and backward phase. The following notations are used in the figure.

$z_j$  : input to node j for layer l

$g_j$  : activation function for node j in layer l (applied to  $z_j$ )

$a_j = g_j(z_j)$ : output/activation of node j in layer l

$w_{ij}$  : weights connecting node i in layer (l-1) to node j in layer l

$b_j$  : bias for unit j in layer l

$t_k$  : target value for node k in the output layer [13]

Some algorithm parameters used in training ANN:

1. **Epoch:** An epoch is a measure of the number of times all of the training examples are used once to update the weights.
2. **Visible and Hidden dropout:** The term dropout refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we temporarily remove it from the network, along with all its incoming and outgoing connections.

### 5.3 Word2vec Algorithm

Word2vec uses a single hidden layer, fully connected neural network. The hidden layer neurons are linear neurons. Input and output layer size is the same as that of vocabulary for training. A dimension has to be given to the algorithm as to get the number of features per word. Hidden layer size is that of the size of the dimension [14].

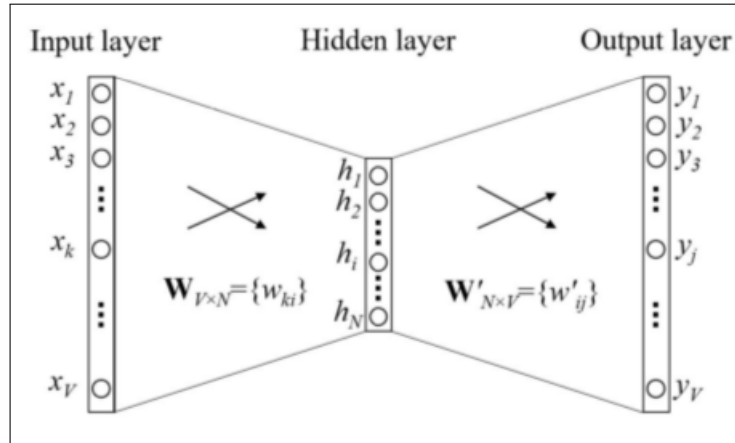


Figure 5.5: Word2vec model architecture [15]

Let  $V$  be the vocabulary size and  $N$  be the dimension of the word vectors. The word2vec model stores two matrices  $W$  of dimension  $V \times N$  for storing weights of input to hidden layer connections. Each row represents a word in the vocabulary  $W'$  is of dimension  $N \times V$  for hidden layer to output layer connections. Each column represents a word in the vocabulary Figure(5.5) [14].

The input is one hot encoded vector which means only one input is set to 1 and the rest are zero. One hot encoding just copies the wordvector from input to the hidden layer.

Output at the Hidden layer  $H^t = X^t W$

Output at the output layer  $= H^t W'$

In order to produce probabilities for words in the output layer, the activation values of output layer neurons is converted using the softmax function as given below:

$$P(word_k | word_{context}) = \frac{\exp(activation(k))}{\sum_{n=1}^V \exp(activation(k))}$$

Using the target vector, the error for the output layer can be computed by subtracting the probability vector from the target vector. After the errors are calculated, the weights in the matrices  $W$  and  $W'$  are updated using the BPP algorithm. In this way, word2vec learns relationships between words and creates vector representation for words in the corpus [14].

Input and output words are selected by 1) Continuous Bag of Words Model and 2) Skip Gram Model

### 5.3.1 Continuous Bag of Words (CBOW) model

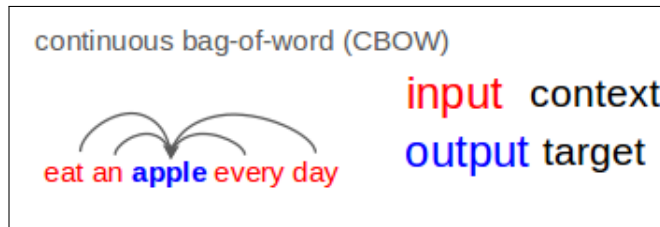


Figure 5.6: Input-Output for CBOW [16]

For CBOW model context is input whereas target is the output. In this model, context may be given by many words for a given target word. For a given number of context words  $C$ ,  $C$  times input to hidden layer connections are cloned. Their outputs are added and normalized by division by  $C$  operation at the hidden layer. Output layer is the same as used for 1 context model above [14].

### 5.3.2 Skip Gram (SG) Model

SG model reverses the usage of target and context words. Target is given as input and context is predicted. Hidden layer is of the number of dimensions (word vectors) similar to that of CBOW. The output layer is cloned the context word times. Multiple output vectors are generated at output layer with multi context predictions. Their results are summed up to adjust weights in BPP algorithm [14].

## 5.4 Decision Tree based Algorithms

C5.0 and Random Forest algorithms are based on Decision Tree learning. They classify instances by sorting them down the tree. Every node in the tree specifies a test of some attribute of the instance.

Attribute selection: Based on Information Gain and Entropy

$$Entropy = \sum_j -p_j \log_2 p_j$$

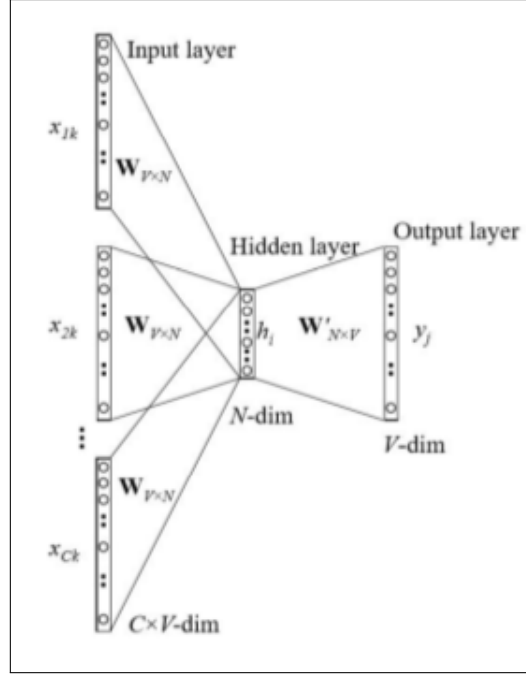


Figure 5.7: Multi context CBOW model [15]

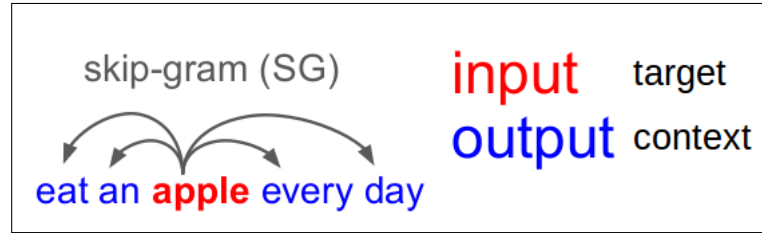


Figure 5.8: Input-Output for SG [16]

$p_j$  is the probability of class  $j$

$$InfoGain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where  $Values(A)$  is the set of all possible values for attribute  $A$ , and  $S_v$  is the subset of  $S$  for which attribute  $A$  has value  $v$ .

Figure 5.10 shows a decision tree created for Credit worthy problem.

In order to avoid overfitting a decision tree, pruning is done which removes the nodes whose removal most increases the decision tree accuracy on the graph. Pruning is done in two ways:

1. Reduced Error Pruning
2. Cost Complexity Pruning

### C5.0

ID3 is the basic algorithm whereas C4.5 and C5.0 its extended versions. ID3 works with only nominal attributes for classification with no missing values. C4.5 works with continuous attributes, handles missing values, supports pruning. C5.0 supports boosting, pre-filter attributes

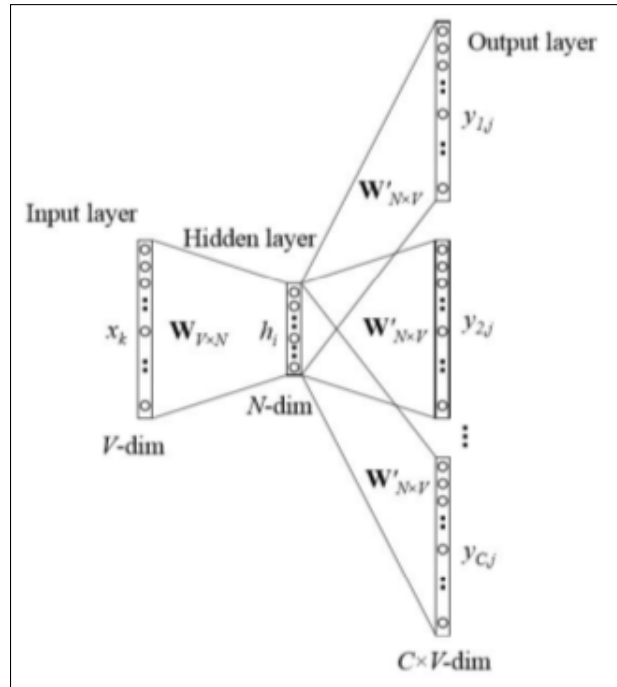


Figure 5.9: Multi context SG model [15]

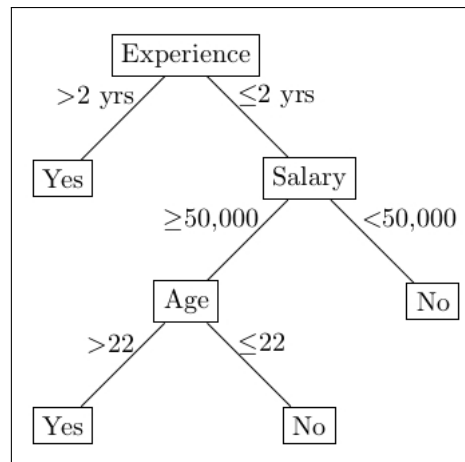


Figure 5.10: Decision Tree algorithm representation

(winnowing) and improved scalability.

### Random Forests

Random forests create an ensemble of decision trees. Random subset of features are selected and a forest having multiple decision trees is generated. Every example passes through all trees and gets the output that has the highest votes given by all the trees.

## 5.5 Dimensionality Reduction: Singular Value Decomposition (SVD)

SVD is a matrix decomposition technique. It is used for dimensionality reduction in text and other data analysis tasks. The mathematical formulation of SVD is explained below. The SVD of matrix  $A$  having dimensions  $m \times n$  can be written as:



$$A = U W V^T$$

where,

$U$  = The columns of  $U$  are the eigenvectors of  $AA^T$ . Dimension:  $m \times \min(m, n)$

$W$  = Vector having singular values of  $A$ . They are the square root of the eigenvalues of both  $AA^T$  and  $A^T A$ . Dimension:  $\min(m, n)$

$V$  = The columns of  $V$  are the eigenvectors of  $A^T A$ . Dimension:  $n \times \min(m, n)$

In relation to text mining, SVD is applied to the huge DTM to reduce the dimensionality. Let DTM be our  $A$  matrix of dimension  $m(\text{documents}) \times n(\text{terms})$ . After analysing the variance explained by  $W$ , we detect that it is sufficient to keep first  $k$  ( $k < n$ ) number of terms instead of  $n$ . Now, we drop all the columns in  $V$  beyond  $k$ . The new  $V$  matrix is truncated to  $n \times k$ . Lets call it  $V'$ .  $V'$  is also called the Mixing Matrix.

The mixing matrix allows to project the original data in the  $k$  dimensional space. The original  $A$  matrix is multiplied by  $V'$  to get the low dimensional space ( $A'$ ) in the following way:

$$A' = A_{[m \times n]} \times V'_{[n \times k]}$$

$$A' = A_{[m \times k]}$$

The mixing matrix should be retained after the training to predict on new data [22].

## 5.6 VGG16

VGG16 model is developed by the The Visual Geometry Group at the Oxford University, UK. It is based on using Convolutional Neural Networks for classifying images. It has been trained for about 1000 classes using the ImageNet data. It contains 16 convolutional layers for extracting different features in an image. 5 max-pooling layers are added in between to reduce the dimensionality. The last layer of the network is removed for my purpose. Figure 5.11 explains the convolutional layers used in this model.

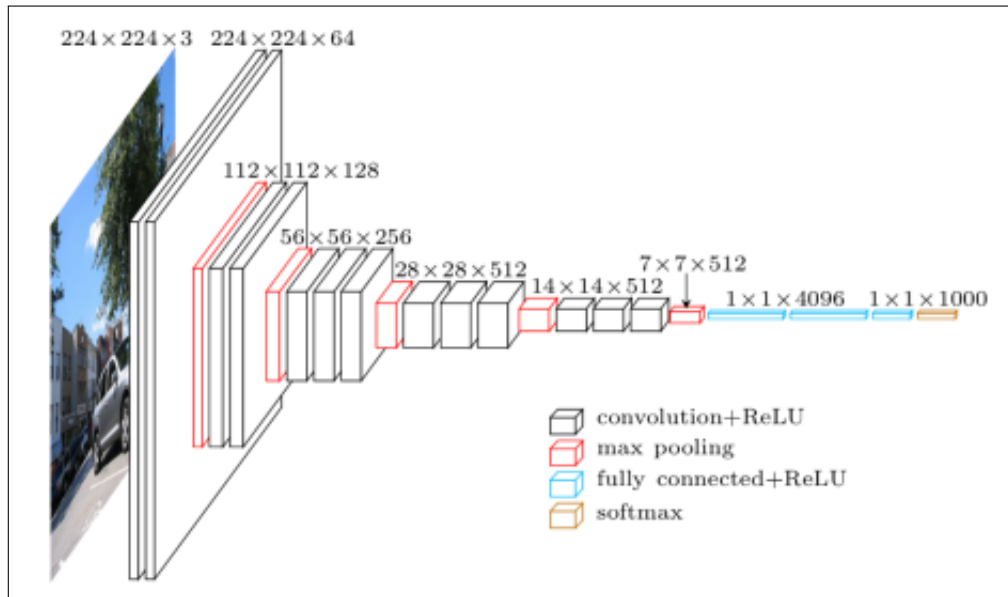


Figure 5.11: VGG16 model representation [24]

### Architecture of the model

The image is passed through a stack of convolutional layers, where filters with a very small receptive field of  $3 \times 3$  and  $1 \times 1$ .  $1 \times 1$  convolution filters can be seen as a linear transformation of the input channels. The convolutional stride is fixed to 1 pixel. The spatial padding of convolutional layer input is such that the spatial resolution is preserved even after convolution, the padding is 1 pixel for  $3 \times 3$  convolutional layers. Spatial pooling is carried out by 5 max-pooling layers which are followed by some convolutional layers. Max-pooling is performed over a  $2 \times 2$  pixel window, with stride 2.

All hidden layers are equipped with the rectification non-linearity (ReLU) function [23]. Mathematical expression for ReLU:  $f(x) = \max(0, x)$ . Figure 5.12 shows the graph of ReLU function.

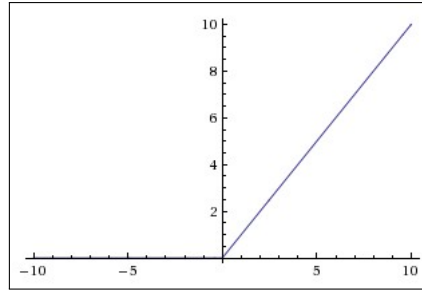


Figure 5.12: ReLU function [21]

The input to this model is a  $224 \times 224$  fixed size colour image (RGB components). It passes through the model to give 4096 convolved features which can be passed to any classifier. These features were then passed to the 2-hidden layer ANN with 4096 and 1000 neurons in each layer respectively. Keras [31] package running on top of Theano was used to build the ANN in Python.

## 5.7 K-Fold Cross Validation (CV) Method

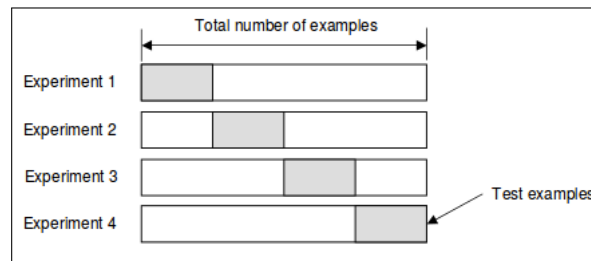


Figure 5.13: k-fold cross validation [25]

In  $k$ -fold CV method, the dataset is divided into  $k$  subsets. Each time the  $k$  fold is kept as test set and the other  $k - 1$  folds are used for training (Figure 5.13). The evaluation measures are then averaged out for all the folds. The advantage of  $k$ -fold CV method is that all examples in the dataset are used for training and testing.

## 5.8 Evaluation Measure Definitions

*Notations:*  $D$  being the Multi Label Dataset (MLD),  $L$  the full set of labels used in  $D$ ,  $Y_i$  the subset of predicted labels for the  $i$ th instance. and  $Z_i$  the true subset of labels.

1. Hamming Loss (**HLoss**): The symmetric difference between predicted and true labels and divided by the total number of labels in the MLD.

$$HammingLoss = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|}$$

2. Accuracy (**Acc**): The proportion of correctly predicted labels with respect to the total number of labels for each instance.

$$Accuracy = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

3. Subset Accuracy (**subAcc**): This metric is also known as 0/1 *Subset Accuracy* and *Classification Accuracy*. It is the most strict evaluation metric. The  $\llbracket expr \rrbracket$  operator returns 1 when  $expr$  is true and 0 otherwise. It is 1 only if the predicted set of labels equals the true one.

$$SubsetAccuracy = \frac{1}{|D|} \sum_{i=1}^{|D|} \llbracket Y_i = Z_i \rrbracket$$

For Precision, Recall, F1 measure and Area Under the Curve, considered the micro-averaging approach, which first aggregates the counters for all the labels and then computes the metric only once.

4. Precision (**Prec**): The ratio of relevant labels predicted by the classifier.

$$Precision = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|}$$

5. Recall (**Rec**): The ratio of predicted labels which are relevant.

$$Precision = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i|}$$

6. F1-measure (**F1**): This metric is the harmonic mean between Precision and Recall, providing a balanced assessment between precision and sensitivity.

$$F1 - measure = 2 * \frac{Precision \cdot Recall}{Precision + Recall}$$

7. Area Under the Curve (**AUC**): This metric calculates the area under the curve ROC (Receiver Operating Characteristic) by trapezoidal rule of integration.

$$MicroAUC = \frac{1}{|L|} \frac{|x', x'' : y', y'' : rank(x', y') \geq rank(x'', y''), (x', y') \in S^+, (x'', y'') \in S^-|}{|S^+| + |S^-|}$$

where,  $rank(x_i, y)$  returns the position of  $y$ , a certain label, in the  $x_i$  instance.

$$S^+ = (x_i, y) : y \in Y_i, \quad S^- = (x_i, y) : y \notin Y_i$$

ROC curve is a graphical representation that measures performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate ( $TPR$ ) against the false positive rate ( $FPR$ ) at various threshold settings (Figure 5.15), where,

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

and

$$FPR = \frac{FP}{FP + TN}$$

Refer Figure 5.14 for the  $TP, TN, FP, FN$  terminologies.

8. Binary cross-entropy: The cross entropy error for a single example with  $n$  independent targets.

$$Entropy = - \sum_{i=1}^n \{t_i \log(x_i) + (1 - t_i) \log(1 - x_i)\}$$

where,  $t$  is the target,  $x$  is the output, indexed by  $i$ . The activation function is the logistic function applied to the weighted sum of the neurons inputs,

$$x_i = \frac{1}{1 + e^{-s_i}}$$

$$s_i = \sum_{j=1} x_j w_{ji}$$

where  $x_j$  is the activation of the  $j$  node in the hidden layer

		Predicted Condition	
		Predicted Condition positive	Predicted Condition negative
True condition	condition positive	True Positive (TP)	False Negative (FN)
	condition negative	False Positive (FP)	True Negative (TN)

Figure 5.14: Confusion matrix

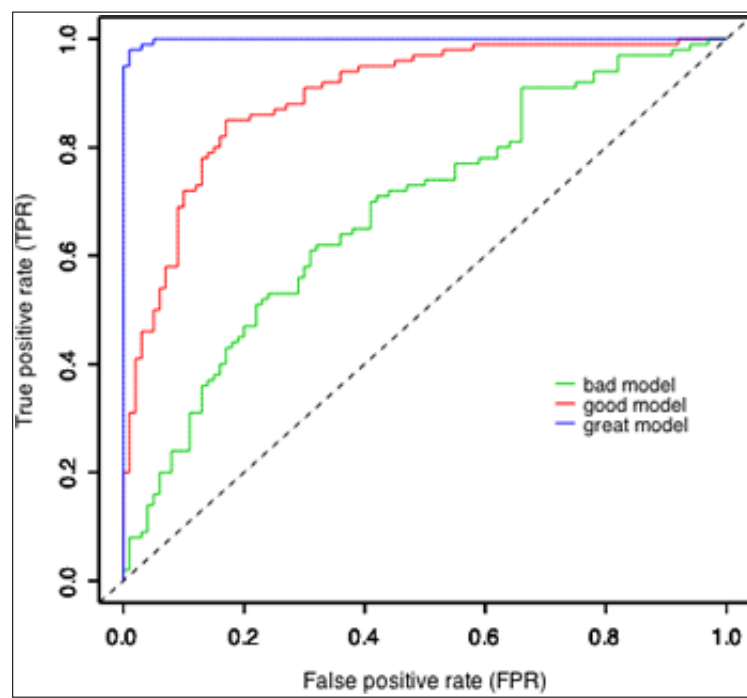


Figure 5.15: ROC curve [26]



## Chapter 6

# Results and Discussion

The models for business, reviews, tips and images were built and optimized separately. The result tables show the mean values obtained from 5-fold cross validation procedure described in Section 5.7. Study was done to check the performance of a Problem Transformation based algorithm (Decision Tree C5.0) and Algorithm Adaptation based algorithm (Deep Neural Networks). These algorithms were implemented using packages C50 and deepnet in R software.

Note: If all the classifiers yield negative output then the label with highest frequency in training is directly assigned to that example.

### 6.1 Business dataset

There were total 16,864 businesses in the Business dataset

#### 6.1.1 Data cleaned by R

Business data had a lot of missing values. Therefore, decision tree C5.0 algorithm was chosen to start-with as it handles missing values in model building process. Pruning of the tree was done to prevent overfitting. Table 6.1 summarizes these results. Basically, a study was made to check the importance of missing values in the data.

#### Decision tree C5.0

DTM Parameters: Sparsity: 0.999 (Unigrams and Bigrams)

No. of Training examples (per fold): 13,491

Feature Set: Non-missing data: 517, Special attributes: 7, Time features: 11, Check-in features: 24

Models Explanation in brief:

1. Vector space model as discussed in Section ?? was built with unigram and bigram features.
2. Other non-missing features like `star rating`, `review count`, `nOpendays` were binded to the Model-1 and sent to the classifier.
3. Special attributes having missing information less than 30% were added to Model-2.
4. Time features generated as per Section 3.2.2 were added to Model-3. These also had missing values.

5. Check-in features generated as per Section 3.2.3 were added to Model-3. These also had missing values.
6. Time features and Check-in features were added together to Model-3.
7. Feature `review count` was removed from Model-6 as it did not add much information.

Model	HLoss	Acc	AUC	F1	Prec	Rec
1	0.200	0.800	0.800	0.807	0.812	0.801
2	0.185	0.815	0.815	0.820	0.825	0.816
3	0.150	0.850	0.850	0.855	0.856	0.855
4	0.143	0.857	0.857	0.861	0.865	0.858
5	0.143	0.857	0.857	0.862	0.864	0.859
6	0.140	0.860	0.859	0.864	0.868	0.861
7	0.141	0.859	0.859	0.864	0.865	0.862

Table 6.1: C5.0 results on **business** dataset (I)

Missing values did carry valuable information to category inference indicated by decrease in the HLoss value. By missing value imputation, Recall also increased by 6%.

### Deep neural networks

Deep neural networks (DNN) require complete data and no missing values. 2-3 layer neural network was used in the training process. Optimization was done by tuning the parameters like number of neurons per layer (**Neurons**), number of epochs (**Epoch**), hidden dropout (**hd**) and visible dropout (**vd**). Activation and Output function used was “sigmoid”. Non-missing data included N-gram features, `star rating`, `nOpenDays`. Table 6.2 summarizes these results.

Default parameters for DNN in `deepnet` package are: learning rate (**lr**):0.8, momentum: 0.5, batch size: 100

#### Without missing values

DTM Parameters: Sparsity: 0.999 (Unigrams and Bigrams)

No. of Training examples (per fold): 13,491

Feature Set: Non-missing data: 517

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	200,250	100	-	-	0.145	0.855	0.854	0.863	0.847	0.880
2	200,80	100	-	-	0.142	0.858	0.858	0.864	0.862	0.866
3	100,50	100	-	-	0.146	0.854	0.853	0.860	0.853	0.867
4	100,50	100	0.1	-	0.145	0.855	0.855	0.860	0.861	0.867
5	100,50	100	0.2	-	0.153	0.847	0.847	0.852	0.855	0.850
6	100,50	100	0.3	-	0.148	0.852	0.852	0.859	0.853	0.865
7	100,50	100	0.1	0.1	0.150	0.850	0.849	0.858	0.844	0.873
8	100,50	100	0.2	0.1	0.149	0.851	0.850	0.858	0.848	0.871

Table 6.2: DNN results on **business** dataset (I)

Model-3 was selected for further development because it had optimum results and a good Recall value.



**With missing value imputation**

Special attributes, time features and check-ins were imputed by the process described in Sections 3.2.5 respectively. Table 6.3 summarizes these results.

DTM Parameters: Sparsity: 0.999 (Unigrams and Bigrams)

No. of Training examples (per fold): 13,491

Feature Set: Non-missing data: 517, Special attributes: 7, Time features: 11, Check-in features: 24

Models Explanation in brief:

1. Non-missing data and imputed special attributes (Model-9)
2. Non-missing data and imputed special attributes, imputed time features (Model-10,11,12)
3. Non-missing data and imputed special attributes, imputed time features, imputed check-in features (Model-13)
4. SVD was performed on DTM, rest all other features were kept same as Model-10. By taking features upto 90% variance dimensionality reduction was achieved. 515 DTM features were replaced by 365 SVD components (Model-14).

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
9	100,50	100	-	-	0.116	0.884	0.884	0.889	0.885	0.894
10	100,50	100	-	-	0.100	0.900	0.900	0.903	0.905	0.902
11	180,150,100	100	-	-	0.106	0.894	0.894	0.898	0.897	0.900
12	150,70,150	100	-	-	0.103	0.897	0.897	0.901	0.904	0.898
13	100,50	100	-	-	0.188	0.812	0.812	0.820	0.817	0.822
14	100,50	100	-	-	0.107	0.893	0.893	0.896	0.899	0.894

Table 6.3: DNN results on **business** dataset (I)

Special attribute imputation (Model-9) lowered the HLoss which showed the imputation worked.

Adding time imputed features (Model 10-12) lowered the HLoss even further and gave a good F1 score.

Check-in imputation (Model-13) did not help in improving the results. Also almost all the check-in features had missing content greater than 50%. Check-ins were not considered further in model development.

SVD did not improve the accuracy.

Model-10 was selected as the best model so far.

**6.1.2 Data cleaned by Python**

The best models from were recreated with the new cleaned data to check for improvement. Table 6.4 summarizes these results.

DTM Parameters: Sparsity: 0.999 (Unigrams and Bigrams)

No. of Training examples (per fold): 13,491

Feature Set: Non-missing data: 535, Special attributes: 7, Time features: 11, Check-in features: 24

Models Explanation in brief:

1. Model-7 from Section 6.1.1 was reproduced with the new cleaned data.
2. Feature selection can be done in algorithm C5.0 by setting the parameter `winnow` which removes non-informative features. Model-8 from Section 6.1.1 reproduced with feature selection.
3. Model-4 from Section 6.1.1 was recreated with increased number of epochs.
4. SVD was performed on DTM, rest all other features were kept same as Model-3. By taking features upto 90% variance dimensionality reduction was achieved. 535 DTM features were replaced by 412 SVD components and DNN was used as classifier (Model-4).

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.145	0.855	0.855	0.861	0.858	0.864
2	-	-	-	-	0.158	0.842	0.841	0.847	0.847	0.848
3	100,50	300	-	-	0.082	0.918	0.918	0.921	0.923	0.919
4	100,50	300	-	-	0.081	0.919	0.919	0.922	0.924	0.920

Table 6.4: C5.0 and DNN results on **business** dataset (II)

Winnowing did increase the HLoss because it removed some features but it is less overfitted model. DNN did show improvement in the Accuracy and hence `Python` cleaned data was used for training in final model building process.

## 6.2 Review dataset

There were total 73,576 reviews in Review dataset. In addition to text there are 4 special attributes namely: `stars`, `votes.useful`, `votes.funny`, `votes.cool` which were used in the model building.

As discussed in Section 3 I had considered maximum 5 reviews per business. In the non-Elite dataset any 5 random reviews are selected if a business has more than 5 reviews.

In the Elite dataset reviews are sorted in the decreasing order of eliteness (Section ) and top 5 were selected. This was done to reduce garbage reviews from entering the dataset. However, if a business had less than or equal to 5 reviews no such measure was used.

### 6.2.1 Non-Elite dataset

#### Data cleaned by R

R Model: Default parameters with Pruning

DNN Model: Optimization was done by tuning the parameters like number of neurons per layer (`Neurons`), number of epochs (`Epoch`), hidden dropout (`hd`) and visible dropout (`vd`). Activation and Output function used was “sigmoid”.

DTM Parameters: Sparsity: 0.99 (Unigrams) and 0.995 (Bigrams)

No. of Training examples (per fold): 58,861

Feature Set: N-gram features: 1046, Special attributes: 4

Models Explanation in brief:

1. C5.0 algorithm (Model-1)
2. DNN algorithm (Models 2-6)
3. SVD was performed on DTM, rest all other features were kept same as Model-2. By taking features upto 90% variance dimensionality reduction was achieved. 1050 DTM features were replaced by 850 SVD components and DNN was used as classifier (Model-7).

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.105	0.895	0.896	0.899	0.904	0.894
2	100,50	100	-	-	0.101	0.899	0.899	0.902	0.909	0.896
3	100,50	100	0.2	-	0.100	0.900	0.901	0.903	0.914	0.892
4	100,50	300	-	-	0.084	0.916	0.916	0.918	0.924	0.913
5	100,50	300	0.2	-	0.098	0.902	0.902	0.905	0.909	0.902
6	100,50	300	-	0.2	0.100	0.900	0.900	0.903	0.911	0.896
7	100,50	300	-	-	0.089	0.911	0.911	0.914	0.919	0.909

Table 6.5: C5.0 and DNN results on **review** dataset (I)

Model-4 gave the best results and was used later.

#### Data cleaned by Python

R Model: Default parameters with Pruning

DNN Model: Optimization was done by tuning the parameters like number of neurons per layer (**Neurons**), number of epochs (**Epoch**), hidden dropout (**hd**) and visible dropout (**vd**). Activation and Output function used was “sigmoid”.

DTM Parameters: Sparsity: 0.99 (Unigrams) and 0.995 (Bigrams)

No. of Training examples (per fold): 58,861

Feature Set: N-gram features: 833, Special attributes: 4

Models Explanation in brief:

1. C5.0 algorithm (Model-1)
2. DNN algorithm (Model-2) with the parameters of (Model-4) in Section 6.2.1.
3. SVD was performed on DTM, rest all other features were kept same as Model-2. By taking features upto 90% variance dimensionality reduction was achieved. 833 DTM features were replaced by 687 SVD components and DNN was used as classifier (Model-7).

Python cleaned data gave a good recall value. SVD (Model-3) did not improve the results and hence was not used further.

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.105	0.895	0.895	0.898	0.905	0.891
2	100,50	300	-	-	0.083	0.917	0.918	0.919	0.930	0.909
3	100,50	300	-	-	0.101	0.899	0.900	0.902	0.911	0.894

Table 6.6: C5.0 and DNN results on **review** dataset (II)

### 6.2.2 Elite dataset

R Model: Default parameters with Pruning and Winnowing

DNN Model: Optimization was done by tuning the parameters like number of neurons per layer (**Neurons**), number of epochs (**Epoch**), hidden dropout (**hd**) and visible dropout (**vd**). Activation and Output function used was “sigmoid”.

DTM Parameters: Sparsity: 0.99 (Unigrams) and 0.995 (Bigrams)

No. of Training examples (per fold): 58,861

Feature Set: N-gram features: 951, Special attributes: 4

Models Explanation in brief:

1. C5.0 algorithm Pruning (Model-1)
2. C5.0 algorithm with Pruning and Winnowing (Model-2)
3. DNN algorithm (Models 3-4) with the parameters of Model-2 in Section 6.2.1.

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.098	0.902	0.902	0.905	0.910	0.901
2	-	-	-	-	0.105	0.895	0.895	0.899	0.903	0.896
3	100,50	300	-	-	0.075	0.925	0.925	0.927	0.931	0.924
4	100,50	300	0.2	-	0.081	0.919	0.919	0.922	0.928	0.916

Table 6.7: C5.0 and DNN results on **review** dataset (III)

Model-3 gave the best result so far and was used in the final model.

### Word2vec results

Word vectors obtained from Section 3.2.1 were converted to Average Feature vectors for the entire text of an example. Skip-gram model with hierarchical softmax type was used. DNN and Random forest (RF) were used as classifiers. Number of trees (**N<sub>Est</sub>**) parameter of RF was kept as 500. Parameters of **word2vec** algorithm were:

No. of Training examples (per fold): 58,861

Number of processors: 4

Minimum word count: 1

Learning rate: 0.025

Epochs: 5

Some others were varied to check performance which are discussed below:

Models Explanation in brief:

1. Model 1-3 parameters: Number of features: 100, Context: 10, Batch of words: 1000

2. Model-4 parameters: Number of features: 300, Context: 5, Batch of words: 500

Model	NEst	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	500	-	-	0.051	0.949	0.948	0.949	0.948	0.951
2	-	100,50	100	0.053	0.947	0.947	0.948	0.951	0.945
3	-	100,50	300	0.053	0.948	0.948	0.948	0.949	0.947
4	500	-	-	0.052	0.948	0.948	0.949	0.945	0.955

Table 6.8: Word2vec results on **review** dataset (III)

Model-1 gave the best performance and was used later.

## 6.3 Tip dataset

There were total 36,517 tips in Tip dataset. In addition to text there is one special attribute **likes** which was used in model building.

As discussed in Section 3, I had considered maximum 5 tips per business. Elite and non-Elite datasets were prepared as in Section 6.2.

### 6.3.1 Non-Elite dataset

#### Data cleaned by R

R Model: Default parameters with Pruning

DNN Model: Optimization was done by tuning the parameters like number of neurons per layer (**Neurons**), number of epochs (**Epoch**), hidden dropout (**hd**) and visible dropout (**vd**). Activation and Output function used was “sigmoid”.

DTM Parameters: Sparsity: 0.995 (Unigrams and Bigrams)

No. of Training examples (per fold): 29,214

Feature Set: N-gram features: 235 , Special attribute: 1

Models Explanation in brief:

1. C5.0 algorithm (Model-1)
2. DNN algorithm (Models 2-9)
3. SVD was performed on DTM, rest all other features were kept same as Model-2. By taking features upto 90% variance dimensionality reduction was achieved. 235 DTM features were replaced by 194 SVD components and DNN was used as classifier (Model-10).

None of the models gave an Accuracy more than 80%.

#### Data cleaned by Python

R Model: Default parameters with Pruning

DNN Model: Optimization was done by tuning the parameters like number of neurons per layer (**Neurons**), number of epochs (**Epoch**), hidden dropout (**hd**) and visible dropout (**vd**). Activation and Output function used was “sigmoid”.

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.246	0.754	0.730	0.738	0.751	0.726
2	100,50	100	-	-	0.265	0.735	0.734	0.748	0.747	0.750
3	100,50	300	-	-	0.262	0.738	0.736	0.755	0.741	0.771
4	100,50	300	0.15	-	0.268	0.732	0.731	0.749	0.738	0.762
5	100,50	300	-	0.1	0.268	0.732	0.731	0.749	0.738	0.762
6	100,100,50	100	-	-	0.312	0.688	0.688	0.700	0.708	0.691
7	200,100	100	-	-	0.287	0.713	0.681	0.728	0.724	0.731
8	220,150	100	-	-	0.283	0.717	0.716	0.729	0.732	0.726
9	100,50,20	100	-	-	0.314	0.686	0.686	0.702	0.702	0.701
10	100,50	300	-	-	0.255	0.745	0.743	0.764	0.743	0.788

Table 6.9: C5.0 and DNN results on **tip** dataset (I)

DTM Parameters: Sparsity: 0.995 (Unigrams and Bigrams)

No. of Training examples (per fold):29,214

Feature Set: N-gram features: 210, Special attributes: 1

Models Explanation in brief:

1. C5.0 algorithm (Model-1)
2. DNN algorithm (Model-2) with the parameters of (Model-3) in Section 6.3.1.
3. SVD was performed on DTM, rest all other features were kept same as Model-2. By taking features upto 90% variance dimensionality reduction was achieved. 210 DTM features were replaced by 176 SVD components and DNN was used as classifier (Model-3).

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.271	0.729	0.730	0.737	0.753	0.721
2	100,50	300	-	-	0.248	0.752	0.751	0.763	0.764	0.762
3	100,50	300	-	-	0.251	0.749	0.748	0.765	0.756	0.774

Table 6.10: C5.0 and DNN results on **tip** dataset (II)

Model-2 gave an accuracy greater than 75% but still it was less to be used in the final model. SVD (Model-3) did not improve the results and hence was not used further.

### 6.3.2 Elite dataset

R Model: Default parameters with Pruning and Winnowing

DNN Model: Optimization was done by tuning the parameters like number of neurons per layer (**Neurons**), number of epochs (**Epoch**), hidden dropout (**hd**) and visible dropout (**vd**). Activation and Output function used was “sigmoid”.

DTM Parameters: Sparsity: 0.995 (Unigrams and Bigrams)

No. of Training examples (per fold):29,214

Feature Set: N-gram features: 215, Special attributes: 1

Models Explanation in brief:

1. C5.0 algorithm Pruning (Model-1)

2. C5.0 algorithm with Pruning and Winnowing (Model-2)
3. DNN algorithm (Models 3-4) with the parameters of Model-2 in Section 6.3.1.

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.274	0.726	0.727	0.732	0.753	0.712
2	-	-	-	-	0.275	0.725	0.725	0.734	0.747	0.721
3	100,50	300	-	-	0.261	0.739	0.738	0.756	0.744	0.769
4	100,50	300	0.2	-	0.271	0.729	0.728	0.746	0.736	0.756

Table 6.11: C5.0 and DNN results on **tip** dataset (III)

Model-3 was selected as the model to be used further.

## 6.4 Image dataset

2-layer DNN was implemented in **Python** on image features. Parameters set were: lr: 0.01, momentum: 0.9, loss: 'binary\_crossentropy', batch size: 128, activation: reLu, output: sigmoid. Training examples: 6946, Testing examples: 2080

Models Explanation in brief:

1. No normalization of RGB components
2. Normalization of RGB components

Model	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	4096,1000	100	0.247	0.753	0.753	0.763	0.745	0.781
2	4096,1000	100	0.219	0.781	0.781	0.790	0.770	0.811

Table 6.12: DNN results on **image** dataset

Both models were nearly equal. Normalization did not give any significant results.

## 6.5 Ensemble of classifiers

The best models from business, review, tip and image models were used to build an Ensemble. Separate training set (approx. same quantity as that used by the individual models above) was constructed of 13,500 businesses with their respective 59,174 reviews, 30,499 tips and 6,946 images. A test set of 2,080 examples was constructed containing atleast business information and an image, review and tip were optional. Therefore, some examples did not have review or tips. But this made the dataset look more real.

After data cleaning the text in the examples was completely removed (in case of small tokens). Such examples were removed from the training set and if present in the testing test then the label with highest frequency in training was directly assigned to that example. The model is based on the assumption that the data belonged to either one or both the categories. In case, if a classifier gave negative results for both the categories for the testing set then, the label with highest frequency in training was directly assigned to that example.

Ensemble model contained mandatorily business (“Bus”), review (“Rev”) and image models. Tip (“Tip”) model and word2vec review model (“w2v”) was checked if it improved the overall performance. Image model was checked to see if Plain (“Img”) or Normalized (“ImgN”) image was giving better performance. In the end, the models that gained higher measures were kept and others discarded from the Ensemble model. A new measure **subAcc** as described in Section ?? is also measured in these results. Training and Testing accuracies as well as Ensembled test results are summarized in Table 6.13.



Data	Model	NEst	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec	SubAcc
<b>Training</b>											
Bus	DNN	-	100,50	300	0.085	0.915	0.915	0.918	0.922	0.914	0.902
Rev	DNN	-	100,50	300	0.074	0.926	0.927	0.929	0.937	0.927	0.905
w2v	RF	500	-	-	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Tip	DNN	-	100,50	300	0.271	0.729	0.728	0.747	0.743	0.745	0.675
Img	DNN	-	4096,1000	100	0.000	1.000	1.000	1.000	1.000	1.000	1.000
ImgN	DNN	-	4096,1000	100	0.000	1.000	1.000	1.000	1.000	1.000	1.000
<b>Testing</b>											
Bus	DNN	-	100,50	300	0.155	0.845	0.845	0.847	0.849	0.843	0.830
Rev	DNN	-	100,50	300	0.068	0.932	0.932	0.933	0.935	0.930	0.919
w2v	RF	500	-	-	0.050	0.950	0.950	0.951	0.949	0.954	0.936
Tip	DNN	-	100,50	300	0.328	0.672	0.672	0.681	0.673	0.690	0.642
Img	DNN	-	4096,1000	100	0.247	0.753	0.753	0.763	0.745	0.781	0.710
ImgN	DNN	-	4096,1000	100	0.219	0.781	0.781	0.790	0.770	0.811	0.735
<b>Ensemble</b>											
1. Tips, No word2vec, Plain Image					0.095	0.906	0.907	0.901	0.960	0.850	0.848
2. No tips, No word2vec, Plain Image					0.072	0.928	0.928	0.929	0.925	0.934	0.908
3. No tips, No word2vec, Normalized Image					0.111	0.889	0.890	0.880	0.979	0.799	0.801
4. No tips, Word2vec, Plain Image					0.054	0.946	0.947	0.945	0.976	0.916	0.916
5. No tips, Word2vec, Normalized Image					0.044	0.958	0.958	0.959	0.961	0.956	0.948

Table 6.13: Ensemble results

Tips added more loss as they had a high loss in their individual model. So, they were removed from the Ensemble model.

Image Normalization in Ensemble-3 gave a very low Recall value. However, it performed well as an individual model.

Ensemble-2 gave the best Recall and Subset Accuracy while Ensemble-4 had the lowest Hamming Loss and a high Precision.

Word2vec model on reviews performed wonderfully as an individual model. It has also uplifted the Ensemble model.

Ensemble-5 is the **Best** model of this entire work using the business, review, word2vec on review and normalized image models. It has given the best results in all the measures.



## Chapter 7

# Conclusions and Future Work

### 7.1 Conclusions

I will summarize the work in this section with some of the key findings in this entire project.

Reviews are the most indicative in predicting the business category. Good quality reviews given by “Elite” users makes the model stronger since garbage and fake reviews are not included. Word2vec model gave the best results.

Data preprocessing is an extremely important task in text analysis. The model works on the funda: “Garbage in Garbage out”. Missing data imputation is also an important task so as to reduce the loss of information due to missing value data. 30% missing value imputation worked wonders and helped achieve an Accuracy above 90%. Special attributes and business timings conveyed quality information in categorizing businesses.

Images also convey information helpful to business category inference. Without much processing on the images, the model gave accuracy greater than 75%. Given more infrastructure, this model can be developed even further.

### 7.2 Future work

As it is rightly said, “There’s always scope for improvement”, this model also has a lot of scope for improvement.

Image data model has the most scope for improvement. Only the basic version was used in this work due to time constraints. They are ~ 38K images belonging to the two businesses considered out of which only ~ 9K were used which is even less than 25%. Given more infrastructure this model will achieve great heights.

Tip model was the only model which was discarded even after putting a lot of efforts in lifting its performance. The information in them was lost in building the ensemble. To utilise their information, a small modification can be made by aggregating all the tips received by a business to make it look like one big review, and then adding it in the review dataset as a new example.

Many reviews are written in languages other than English (eg: Spanish) but which have English script. For e.g. consider this review written in Spanish language.

**Raw review:** De las tiendas más bonitas que puedes encontrar en Edimburgo. Te quieres comprar todo lo que ves. Si te gusta el diseo, la ilustracines y las cosas bonitas este es tu sitio.... Tienen ilustraciones para decorar la casa, accesorios, bolsos, chorradas varias, etc. Es una tienda en la que curiosear y dejarse los ahorros del mes. Sin duda alguna si tienes que hacer un regalo a alguien, quieres decorar tu casa o simplemente eres una loca de estas cosas como yo pues este es tu sitio

**English translation:** Of the most beautiful shops you can find in Edinburgh. You want to buy everything you see. If you like the design, artwork and beautiful things this is your place .... Have pictures to decorate the house, accessories, handbags, chorradas groups, etc. It is a shop where you look around and let the savings of the month. No doubt if you have to make a gift to someone, you want to decorate your home or you're just crazy about these things as I do because this is your place

The information in these reviews is mislead because the model could not relate with the English words with the same meaning. Addition of a language translation model in data cleaning part could solve this problem.

Doing a hierarchical classification to find the subcategories a business can have.

# Bibliography

- [1] Reviews that mention posh in London. n.d. photograph, viewed 4 January 2016, <https://www.yelp.co.uk/wordmap/london/posh>
- [2] Crain, K., Heh, K., and Winston, J. (2014). An Analysis of the Elite Users on Yelp.com, Stanford university project report, USA
- [3] Felix, W. (2014). On the efficiency of social recommender networks. *Yelp Academic Challenge Dataset*.
- [4] Hood, B., Hwang, V., and King, J. (2013). Inferring future business attention. *Yelp Challenge, Carnegie Mellon University*.
- [5] Huang, J., Rogers, S., and Joo, E. (2014). Improving restaurants by extracting subtopics from yelp reviews. *iConference 2014 (Social Media Expo)*.
- [6] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.
- [7] Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. Technical report.
- [8] Zhang, M. and Zhou, Z. (2013). A review on multi-label learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1.
- [9] Clare, A. and King, R. D. (2001). *Knowledge Discovery in Multi-label Phenotype Data*, pages 42–53. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [10] Lomonaco, V., (2015). photograph, *Deep Learning for Computer Vision: A comparison between Convolutional Neural Networks and Hierarchical Temporal Memories on object recognition tasks*, Unpublished master’s thesis, University of Bologna, Emilia-Romagna, Italy.
- [11] Understanding Convolutional Neural Networks for NLP. (2015, November 7). photograph, viewed 10 June 2016, <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>
- [12] Loria, S., Tutorial: Finding Important Words in Text Using TF-IDF. (2013, September 1). Retrieved from, <http://stevenloria.com/finding-important-words-in-a-document-using-tf-idf/>
- [13] A Gentle Introduction to Artificial Neural Networks. (2011). photograph, viewed 10 February 2016, <https://theclevermachine.wordpress.com/2014/09/11/a-gentle-introduction-to-artificial-neural-networks/>

- [14] Krishan, Words as Vectors. (2015, April 13). Retrieved from <https://iksinc.wordpress.com/tag/continuous-bag-of-words-cbow/>
- [15] Rong, X. (2014). Word2vec Parameter earning explained. *CoRR*, abs/1411.2738.
- [16] Rong, X., Word Embedding Explained and Visualized. (2015, November 7). photograph, viewed on 26 April 2016, <https://docs.google.com/presentation/d/1yQWN1CDWLzxGeIAvnGgDsIJr5xmy4dB0VmHFKkLiibo/>
- [17] Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.
- [18] Anderson, J. A. (1995). *An introduction to neural networks*. MIT press.
- [19] Haykin, S., Feedforward Neural Networks: An Introduction. n.d. Retrieved from <http://catalogimages.wiley.com/images/db/pdf/0471349119.01.pdf>
- [20] Tadiou, K., Artificial Neural Networks. n.d. photograph, viewed on 4 June 2016, <http://futurehumanevolution.com/artificial-intelligence-future-human-evolution/artificial-neural-networks>
- [21] CS231n Convolutional Neural Networks for Visual Recognition. n.d. photograph, viewed 5 June 2016, <http://cs231n.github.io/neural-networks-1/>
- [22] Albright, R. (2004). Taming text with the svd.
- [23] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.
- [24] A brief report of the Heuritech Deep Learning Meetup. (2016, February 29). photograph, viewed 8 June 2016, <https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/>
- [25] Intelligent Sensor Systems. n.d. photograph, viewed on 8 June 2016, [http://research.cs.tamu.edu/prism/lectures/iss/iss\\_l13.pdf](http://research.cs.tamu.edu/prism/lectures/iss/iss_l13.pdf)
- [26] ROC Curves. (2010, November 10). photograph, viewed on 8 June 2016, <https://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture22.htm>
- [27] R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- [28] Python Software Foundation. Python Language Reference, version 2.7. Available at <http://www.python.org>
- [29] Loper, E. and Bird, S. (2002). NLTK: The Natural Language Toolkit. Technical Report cs.CL/0205028.
- [30] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. <http://is.muni.cz/publication/884893/en>.
- [31] Chollet, F. (2015). Keras. <https://github.com/fchollet/keras>.