

**SIX MONTH INDUSTRIAL TRAINING FINAL REPORT**  
**ON**  
**“AN INTELLIGENT EMOTION POLARITY**  
**CLASSIFIER USING WORD EMBEDDING AND**  
**BIDIRECTIONAL LSTM FOR ENHANCING AIR**  
**TRANSPORT SERVICE QUALITY”**

Submitted in partial fulfilment of the requirements for award of Degree  
of

**BACHELOR OF TECHNOLOGY**  
**(COMPUTER SCIENCE AND ENGINEERING)**  
**SESSION: 2015-2019**



**SUBMITTED TO**

Dr. PANKAJDEEP KAUR

**SUBMITTED BY**

KUNAL VERMA  
2015CSA1476

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**GURU NANAK DEV UNIVERSITY, AMRITSAR (PUNJAB)**  
**INDIA**



## Certificate of Completion

is hereby granted to  
**Kunal Verma**

to certify that he/she has successfully completed  
**Python with Data Science & Machine Learning**

Duration **Dec 15, 2018** to **May 20, 2019**

**Director**  
**Itronix Solution**



Certificate No. **IS/IT/19/018**

**Training Manager**

## **DECLARATION**

I the undersigned solemnly declare that the report of the project work entitled **“AN INTELLIGENT EMOTION POLARITY CLASSIFIER USING WORD EMBEDDINGS AND BIDIRECTIONAL LSTM FOR ENHANCING AIR TRANSPORT SERVICE QUALITY”**, is based my own work carried out during the course of my study under the supervision of Dr. PANKAJDEEP KAUR.

I assert that the statements made and conclusions drawn are an outcome of the project work. I further declare that to the best of my knowledge and belief that the project report does not contain any part of any work which has been submitted for the award of any other degree/diploma/certificate in this University or any other University.

**KUNAL VERMA**

**2015CSA1476**

## **ACKNOWLEDGEMENT**

It is my privilege to express my sincerest regards to our project coordinator, **Dr. PANKAJDEEP KAUR** , for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

It was purely on the basis of her experience and knowledge that I am able to gain all the required information and guidance to initiate this project.

It is of immense pleasure and profound privilege to express my gratitude and indebtedness along with sincere thanks to **Er. KARAN ARORA** , Itronix Solution for providing me the opportunity to work on project “**An Intelligent emotion polarity classifier using word embedding’s and Bidirectional LSTM for enhancing Air Transport service quality.**”.

I take this opportunity to thank all my lecturers who have directly or indirectly helped to complete this project. I pay my respects and love to my parents and all other family members and friends for their love and encouragement throughout my career. Last but not the least I express my thanks to friends for their cooperation and support.

**KUNAL VERMA**

**2015CSA1476**

## **ABSTRACT**

Sentiment analysis is the computational study of the people's opinions, sentiments, attitudes and emotions expressed in the form of feedback or comments in written language. It is one of the most engaging research areas in Natural Language Processing and text mining. Its popularity is mainly due to following reasons:

- 1.) It has wide range of applications, because opinions in the written form of language are the key influencers of the human behavioral and thought process. Opinions helps the person to choose optimum from number of available alternatives.
- 2.) It makes acquainted with many research problems and thus rapid growth of Natural Language Processing and Sentiment Analysis coincide with the social media on the Internet because, huge amount of opinions are shared over these websites in the digital form.

In this Project, We have discussed Sentiment Analysis and Review Classification as mainstream for determining the emotion polarity in the textual data.

Review Classification is the process to analyze the subjective information in the digital text and then classify the opinion into different categories such as positive opinion, negative opinion or neutral opinion. Where Sentiment Analysis is the procedure by which information extracted from the reviews, feedbacks and comments of the people on the particular service, entities and events. Sentiment Analysis is carried out three different levels i.e. phrase, sentence and documents. This Project aims at analyzing a solution for the review analysis on the reviews by different group of people on the Airline Service Quality and to determine how the extracted information can be used as an application for enhancing their services for customer satisfaction. Every day, over 100,000 flights carry passengers to and from destinations all around the world, and it's safe to say air travel brings out a fairly mixed bag of emotions in people. Through social media, customers now have a platform to say exactly what's on their mind while they are traveling, creating a real-time stream of customer opinion on social networks or directly on the website of the Airline.

For this, polarity of the sentence can be given in three categories as positive, negative and neutral.

## **ABOUT THE INSTITUTE**



**ITRONIX SOLUTIONS** that provides services in Education, Development and Consultancy under single umbrella. **ITRONIX SOLUTIONS** has completed successful **4 years** in industry and having own Research and Development centre along with state of art infrastructure. **ITRONIX SOLUTIONS** has been incorporated in order to focus on and take advantage of its expertise in the delivery of effective high end customized training from dot to dimensions.

Our company's extensive capabilities come as a result of being the most preferred training centre for our principles.

**ITRONIX SOLUTIONS** was established in 2013 by a team of professionals from global consulting firms having background in engineering and management from eminent institutes like Lovely Professional University, Phagwara, Punjab guided by senior people from Industry, Academia, & Government Sector; and specializes in customized IT Training on various software products and development tools addressing the growing needs of the IT market.

*Itronix Solutions is one of the leading Advance Embedded Systems, Information Security, IT solutions & Networking Training providers in India. It's a beginning of the new Revolution in the Electronics, Cyber Space Security, IT Solutions & Networking. The term ITRONIX is a union of Embedded Systems in Electronics and securing the Information Technology around the world. We feel like that there is something lagging behind our hometown Punjab, so we decided to run an industry of training to provide the solutions in various fields of the Electronics, Information Security, IT Solutions & Networking. We are assuring you to keep the company –*



*customer relationship at our best. So come and join hands with the beginning of new ITRONIX revolution.*

The revolt for knowledge, the mission for sharing, the promise for accomplishing the dream, and the drive for the passion had come together as ITRONIX. A dream for revolution which was started with *Er. Karan Arora & Er. Deepanshu Khanna* during the college days is finally beginning. This is a revolution and we are the revolutionaries. This is the revolution in the field of the Electronics and the Information Security. During our college days, we saw people memorizing the list of programming languages, configuring hardware's, and lots of other things. But they forgot the difference between the term learning, understanding and the memorizing. And at the end joined the same rush and got placed, but they didn't understand and learn the things, so they didn't survive much in the outer world. So, we all decided to run a firm that will not only help the students but also the corporate persons who work for the knowledge and use their potentials to build up their own path.

With many years of experience working with embedded systems, Linux, Information Security, Web Development and Networking we provide sought-after expertise in these fields. The Professional Development Courses help in honing the basic skills across platforms and methodologies while the Corporate Trainings drive enablement on specific tools and technologies. Itronix Solution has superior infrastructure and highly qualified expertise to training engineers to full fill their tasks & guide them to Embedded and IT related careers. Itronix Solution has world class curriculum listed below and state-of-art Infrastructure and tools required to train the engineers on the Embedded Skills, Information Security and IT Solutions Skills to enable them to independently handle complex projects.

We aim at raising standards of professionalism within the IT training industry and creating standards of excellence against which candidates are measured. **ITRONIX SOLUTIONS** training programs are continuously striving for excellence in education, training, research and consultancy in the fields of Management and Information Technology with a Mission of Offering Value Based Education. We produce IT professionals who can compete with the very best in the global arena and cater to the growing demands of the corporate world.

Our consulting and implementation group has worked in all settings, from small businesses with little or no computerization to large corporations with highly structured IT departments. Our experience in software development and implementation means that consulting recommendations are both practical and technically possible. Process redesign and improvement In the case that a company's supply chain, manufacturing, or distribution operations are structured around a system it is trying to replace, or are geared toward a way of doing business that no longer fits current needs, ITRONIX's consulting team can help the company define its needs, identify specific areas to be addressed, and make recommendations for change.



# **TABLE OF CONTENTS**

DECLARATION	(i)
ACKNOWLEDGEMENT	(ii)
ABSTRACT	(iii)
ABOUT THE INSTITUTE	(iv)
TABLE OF CONTENTS	(vii)
LIST OF ABBREVIATIONS	(x)
LIST OF FIGURES	(xi-xii)
CHAPTERS	PAGE NUMBER
Chapter 1 : Introduction	13-15
1.1) Application Overview and Scope	
1.2) Need for the Application	
1.3) Proposed Application	
Chapter 2 : Literature Survey and Analysis	16-30
2.1) Machine Learning Algorithms	
2.1.1) Logistic Regression	
2.1.2) Support Vector Machine	
2.1.3) Naïve Bayes Classification	
2.2) Deep Learning Algorithms	
2.2.1) Dense Neural Network	
2.2.2) Convolutional Neural Network	

2.2.3) Recurrent Neural Network 2.2.4) LSTM 2.2.5) Bidirectional LSTM  2.3) TF-IDF  2.4) Word Embedding 2.5) Methodologies used in Sentiment Analysis  2.5.1) Classifications 2.5.2) Term Frequency 2.5.3) n-grams 2.5.4) Part-of-Speech 2.5.6) Negations	
Chapter 3 : Application Environment 3.1) Hardware Requirements 3.2) Software Requirements	31
Chapter 4 : System Design  4.1) Design Methodologies  4.1.1) The Data Science Process 4.1.2) The NLP Process 4.1.3) The Model Training Process  4.2) Architecture of the Project  4.3) Data Flow Diagrams	35-42
Chapter 5 : Implementation & Application Design  5.1) Implementation  5.1.1) Exploring Data 5.1.2) Pre-Processing Reviews 5.1.3) Training Models 5.1.2) Deploying Models on Heroku  5.2) View Pages of Each Models	43-56

Chapter 6 : 6.1) Technologies Used  6.1.1) Presentation View 6.1.2) Business View 6.1.3) Database View	57-62
Chapter 7 : Conclusion & Future Scope 7.1 ) Conclusion and Results 7.2 ) Future Scope	63-65
Chapter 8 : References	66

## **LIST OF ABBREVIATIONS**

**ML** – Machine Learning

**DS** – Data Science

**RNN** - Recurrent Neural Network

**CNN** – Convolutional Neural Network

**LSTM** – Long-short term memory

**SVM** – Support Vector Machine

**DFD** – Data Flow Diagram

**RAM** – Random Access Memory

**SQL** – Structured Query Language

**CSV** – Coma Separated Values

**DNN** – Dense Neural Network

**NumPy** – Numerical Python

**PANDAS**- Python Data Analysis Library

**GUI** – Graphical User Interface

**NBC**- Naïve Bayes Classification

**NLTK**- Natural Language Toolkit

**ETC** – Etcetera

## **LIST OF FIGURES**

1. Figure 1 : Dense Neural Network : Page|21
2. Figure 2 : Convolutional Neural Network: Page|22
3. Figure 3 : Recurrent Neural Network : Page|23
4. Figure 4 : Long Short Term Memory Network : Page|24
5. Figure 5 : Bidirectional LSTM : Page|25
6. Figure 6 : Bag of Words : Page|26
7. Figure 7 : TF-IDF Representation : Page|27
8. Figure 8 : Word Embedding : Page|28
9. Figure 9 : Data Science Process : Page|32
10. Figure 10 : Lemmatization : Page|38
11. Figure 11 : Stop Words Removal : Page|38
12. Figure 12 : Machine Learning Process : Page|38
13. Figure 13 : Application Architecture: Page|40
14. Figure 14 : Zero Level DFD: Page|40
15. Figure 15 : 1st Level DFD : Page|41
16. Figure 16 : 2nd Level DFD : Page|41
17. Figure 17 : Most Reviewed Airlines : Page|43
18. Figure 18 : Polarity Chart : Page|44
19. Figure 19 : Classes of Sentiment Chart : Page|45
20. Figure 20 : Word Cloud : Page|45
21. Figure 21 : Original Review Source : Page|53
22. Figure 22 : Logistic Regression View: Page|54
23. Figure 23 : Logistic Regression Result: Page|54
24. Figure 24 : Linear SVM Result : Page|55

25. Figure 25 : Original Source Review : Page|55

26. Figure 26 : CNN Result : Page|56

27. Figure 27 : LSTM Result : Page|56

# **CHAPTER 1**

## **1.) Introduction**

Sentiment analysis is the computational study of the people's opinions, sentiments, attitudes and emotions expressed in the form of feedback or comments in written language. It is one of the most engaging research areas in Natural Language Processing and text mining.

### **1.1) Application Overview and Scope**

Sentiment Analysis is the task of grouping textual data into categories based upon their sentiment/emotions with which the text was written. Sentiment Analysis is a significant learning problem that is at the core of many information management and retrieval tasks. Sentiment Analysis performs an essential role in various applications that deals with recommendation, classifying, summarization and concisely representing a significant amount of information. Automatic sentiment analysis can be broadly classified into two categories. These are supervised and Semi-supervised. In Supervised sentiment analysis, some mechanism external to the analysis model (generally human) provides information related to the correct sentiment for the text. Thus, in case of supervised sentiment analysis, it becomes easy to test the accuracy of the model. In case of Semi-supervised sentiment analysis, parts of the documents are labelled by an external mechanism.

Taking about challenges, there are two main factors which contribute to making sentiment analysis a challenging task: (a) feature extraction; (b) topic ambiguity. First, Feature extraction deals with taking out the right set of features that accurately describes the sentiment and helps in building a good analysis model. Second, many broad topic are themselves so complicated that it becomes difficult to put it into any specific category

### **1.2) Need of the Application**

Some of the applications that make use of the above techniques for document classification are listed below:



- Email routing: Routing an email to a general address, to a specific address or mailbox depending on the topic of the email.
- Language identification: Automatically determining the language of a text. It can be useful in many use cases one of them being the direction in which the language should be processed.
- Readability Assessment: Automatically determining how readable any document is for an audience of a certain age.
- Sentiment Analysis: Determining the sentiment of a speaker based on the content of the document.

### **1.3) Proposed Application**

Some of the techniques that are employed for document sentiment analysis include Expedition maximization, Naïve Bayes classifier, Support Vector Machine, Neural Network, etc. In our project, we aim to use a probabilistic model of machine learning to make predictions of what class of sentiment does a review belong to. The probabilistic model called “Naïve Bayes” makes use of Bayesian Formula for probability to predict the probability of an event A occurring on condition that event B has already occurred. The Reason to use Naïve Bayes instead of other algorithms are as follow:

- Our dataset will be small and hence Naïve Bayes will work good enough on small set of documents.
- Naïve Bayes is a multiclass classifier as opposed to SVM which can classify between only two objects. Though one can point out that neural networks can practically classify in any number of classes , using neural network would be an overkill for the project since the dataset is small and the model would quickly over fit to the problem. Also SVM could have been used with the “One vs Rest” or “One vs One” voting scheme but the accuracy takes a hit with the voting scheme.
- The hardware requirements for Naïve Bayes Classifier is less as compared Neural Networks.
- Naïve Bayes is fairly accurate and commonly used algorithm for text classification since it does not require the sequence analysis of words to make predictions, rather depends upon the conditional probability of words found in each document.

The idea to is design a User Interface using either Web technologies such as “HTML, CSS” or use the built in Tkinter library/flask to provide user an interface to input textual review to be classified according to the types. These types are divided into three categories i.e +ve, -ve, and neutral. When the textual reviews are fed to the trained model they will classify the reviews according to these three categories. Further Emotion polarity of the Reviewer is calculated on the bases of the probability factor of each model and the final emotion in the text is determined.

## **CHAPTER 2**

### **Literature Survey and Analysis**

In this project work, we use two class of supervised learning algorithms the machine learning algorithms such as logistic regression, SVM(Linear), Naïve Bayes, Decision tree and then Deep Learning models such as Dense Neural Nets, Convolutional Neural Networks, GRU, LSTM, Bidirectional LSTM and Recurrent Neural Networks.

#### **2.1) Machine Learning Algorithms**

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

**2.1.1) Logistic Regression** In statistics, the logistic model (or logit model) is a widely used statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an indicator variable, where the two values are labelled "0" and "1".

In the logistic model, the log-odds (the logarithm of the odds) for the value labelled "1" is a linear combination of one or more independent variables ("predictors"); the

independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labelled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labelling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from logistic unit, hence the alternative names.

Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each dependent variable having its own parameter; for a binary independent variable this generalizes the odds ratio. The probability that the output is 1 given its input can be represented as:

$$P(y = 1 | x)$$

Linear regression model can generate the predicted probability as any number ranging from negative to positive infinity, whereas probability of an outcome can only lie between  $0 < P(x) < 1$ . This is solved by logistic regression which is expressed as :

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X.$$

where, the left hand side is called the logit or log-odds function, and  $p(x)/(1-p(x))$  is called odds.

### 2.1.2) Support Vector Machine

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data is unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering algorithm, created by applying the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabelled data, and is one of the most widely used clustering algorithms in industrial applications. For linear kernel the equation for prediction for a new input using the dot product between the input ( $x$ ) and each support vector ( $x_i$ ) is calculated as follows:

$$f(x) = B(0) + \sum(a_i * (x, x_i))$$

### **2.1.3) Naive Bayes Classification**

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

## **2.2) Deep Learning Algorithms**

Deep learning is a sub-field of machine learning dealing with algorithms inspired by the structure and function of the brain called artificial neural networks. In other words, It mirrors the functioning of our brains. Deep learning algorithms are similar to how nervous system structured where each neuron connected each other and passing information.

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts.

Neural networks were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog.

### **2.2.1) Dense Neural Network**

Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks and astrocytes that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labelled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge ‘

about cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

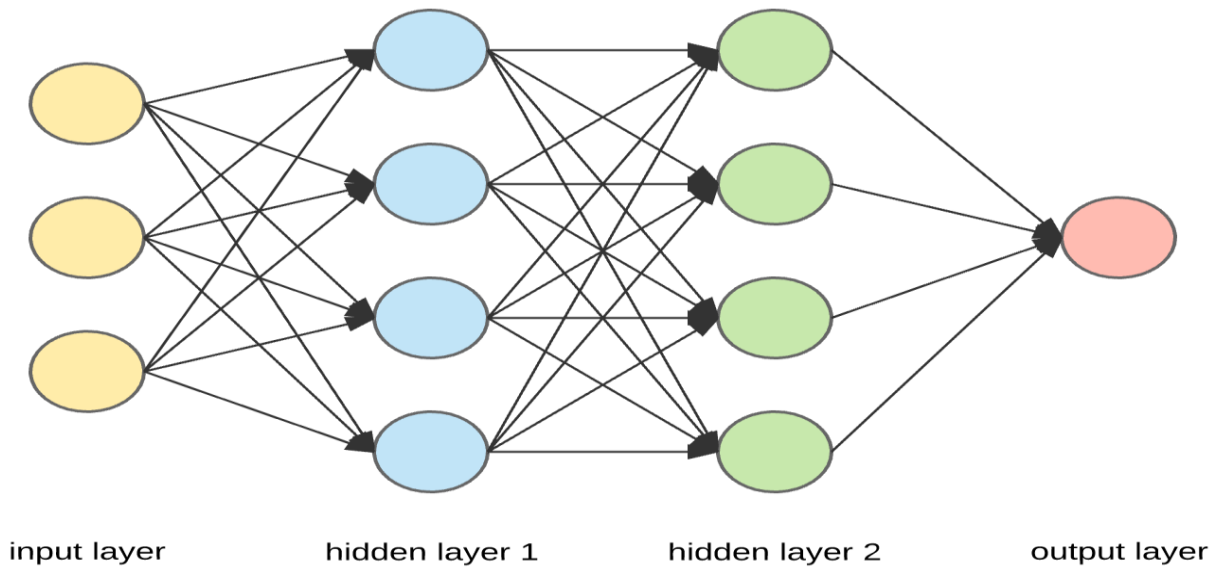
An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical



diagnosis.



### 2.2.2) Convolutional Neural Network

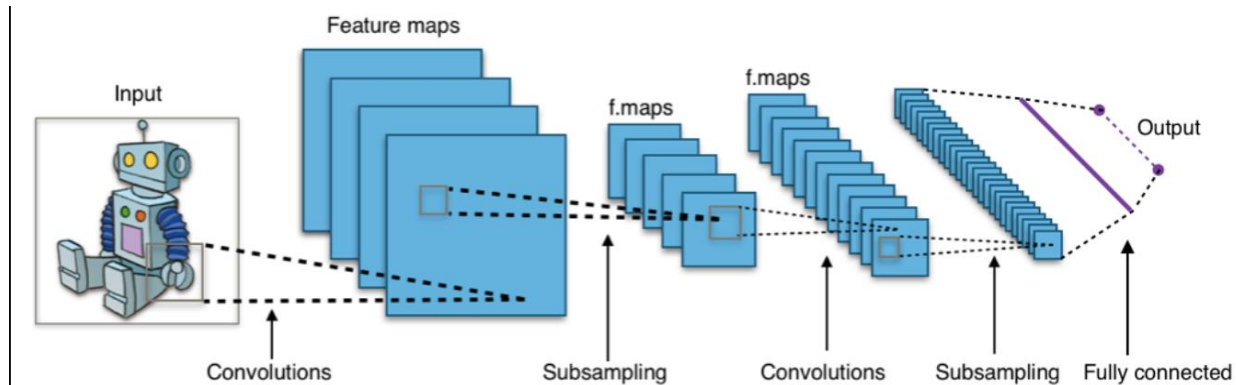
In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analysing visual imagery.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization includes adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional

algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

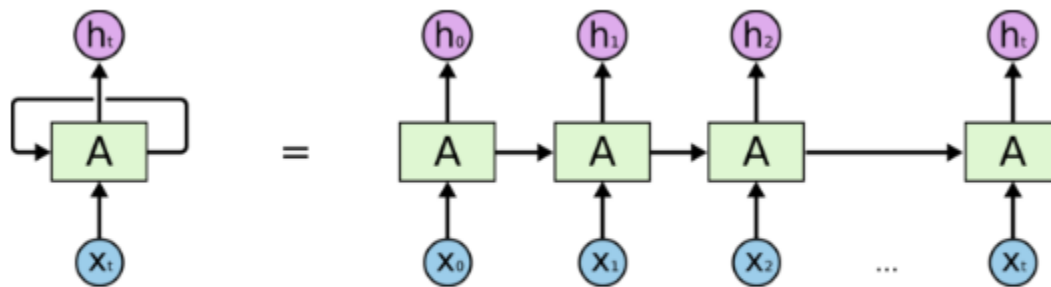


### 2.2.3) Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behaviour for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

A chunk of neural network, 17A, looks at some input  $x(t)$  and outputs a value  $h(t)$ . A loop allows information to be passed from one step of the network to the next. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop.

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modelling, translation, image captioning... The list goes on.



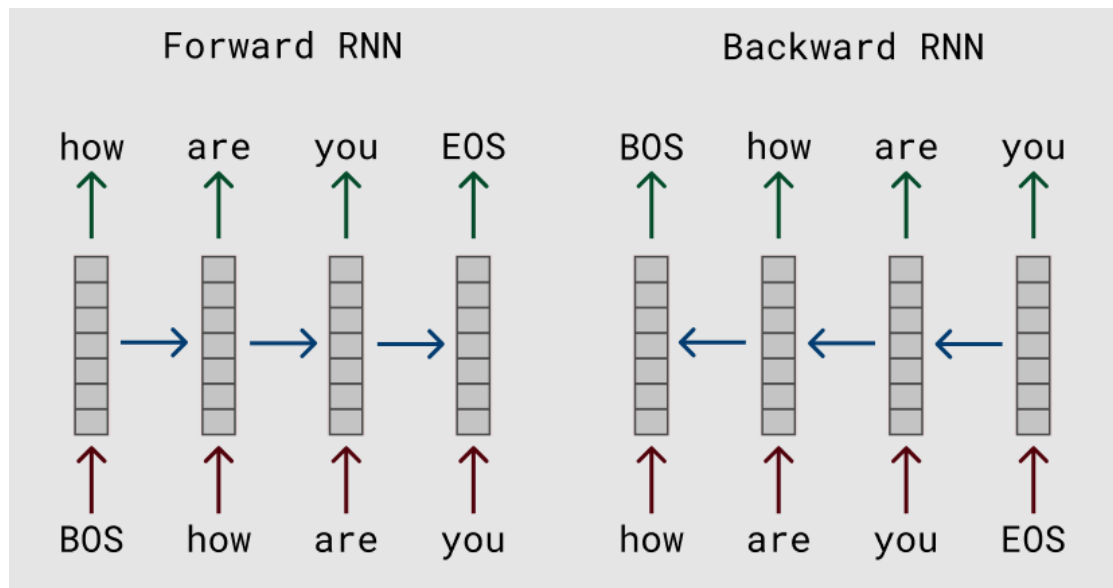
An unrolled recurrent neural network.

## 2.2.4) LSTM

LSTM is a recurrent neural network (RNN) architecture that REMEMBERS values over arbitrary intervals. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods.

The structure of RNN is very similar to hidden Markov model. However, the main difference is with how parameters are calculated and constructed. One of the advantage with LSTM is **insensitivity to gap length**. RNN and HMM rely on the hidden state before emission / sequence. If we want to predict the sequence after 1,000 intervals instead of 10, the model forgot the starting point by then. LSTM REMEMBERS.

The **long-term memory** is usually called the **cell state**. The looping arrows indicate recursive nature of the cell. This allows information from previous intervals to be stored with in the LSTM cell. Cell state is modified by the forget gate placed below the cell state and also adjust by the input modulation gate. From equation, the previous cell state forgets by multiply with the forget gate and adds new information through the output of the input gates.

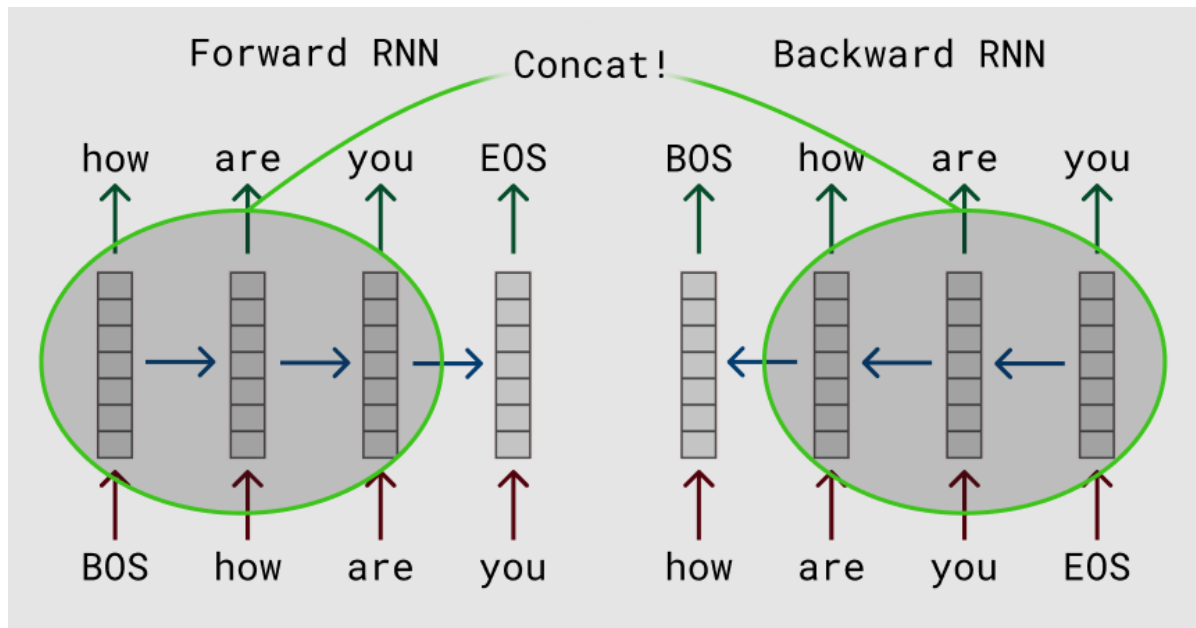


### 2.2.5) Bidirectional LSTM

The usual approach in building a language model is to predict a word given the previous words. We can use either use an n-gram language model or a variant of a recurrent neural network (RNN). An RNN (theoretically) gives us infinite left context (words to the left of the target word). But what we would really like is to use both left and right contexts see how well the word fits in the sentence. A bidirectional language model can enable this.

Most deep learning frameworks will have support for bidirectional RNNs. They will usually return two sets of RNN hidden vectors where one is the output of the forward RNN and the other is the output of the backward RNN. These hidden vectors will be used to predict the next word in the sentence, where next word is the previous word for the backward RNN.

The simple trick is to stagger the hidden vectors so after concatenating them, they are predicting on the same token. Remember to add some padding at both ends of the sentence so we have enough context to predict the words. Here, we add a BOS (beginning of sentence) and EOS (end of sentence) padding tokens.



## Review Representation

Since Different Machine Learning models take as input different word representation of the textual data. We opted for three types of representation mainly, **TF-IDF**, **CountVector** and **Word2Vec**. TF-IDF and CountVector are mainly used classic machine learning algorithms such as SVM etcetera and Word2Vec is very commonly used in deep learning to feed textual data. All these techniques are as follow:

### CountVector / Bag of Words:

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision.

The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier

Bag-of-word model is an order less document representation—only the counts of words mattered. For instance, in the above example "John likes to watch movies. Mary likes movies too", the bag-of-words representation will not

reveal that the verb "likes" always follows a person's name in this text. As an alternative, the n-gram model can store this spatial information. Applying to the same example above, a bigram model will parse the text into the following units and store the term frequency of each unit as before.

Based on these two text documents, a list constructed as follows for each document:

```
"John", "likes", "to", "watch", "movies", "Mary", "likes", "movies", "too"  
"John", "also", "likes", "to", "watch", "football", "games"
```

Representing each bag-of-words as a [JSON object](#), and attributing to the respective [Javascript](#) variable:

```
Bow1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};  
Bow2 = {"John":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};
```

## TF-IDF

In information retrieval, tf-idf or TFIDF, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modelling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. Tf-idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf-idf.

Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf-idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Document space	$t_1$	$t_2$	$t_3$	...	$t_n$	Term vector space
$D_1$	$a_{11}$	$a_{12}$	$a_{13}$	...	$a_{1n}$	
$D_2$	$a_{21}$	$a_{22}$	$a_{23}$	...	$a_{2n}$	
$D_3$	$a_{31}$	$a_{32}$	$a_{33}$	...	$a_{3n}$	
...						
$D_m$	$a_{m1}$	$a_{m2}$	$a_{m3}$	...	$a_{mn}$	
$Q$	$b_1$	$b_2$	$b_3$	...	$b_n$	

## Word Embedding

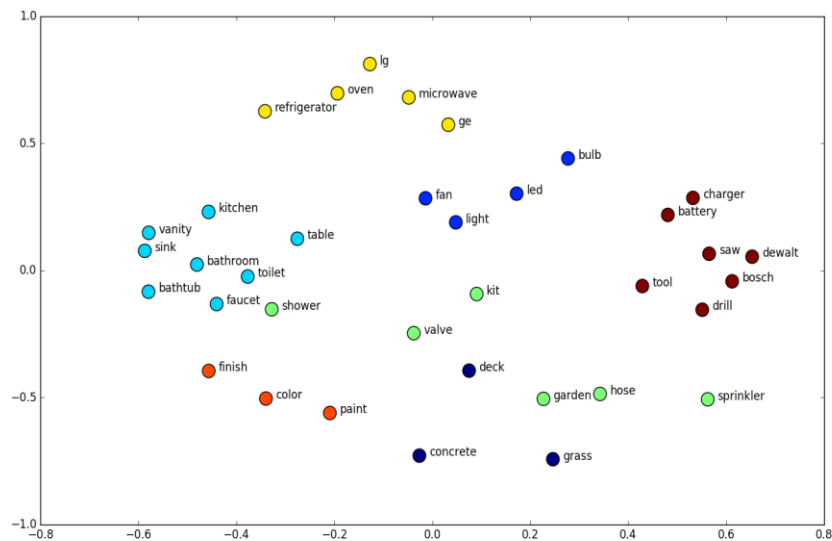
Word embedding is the collective name for a set of language modelling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Methods to generate this mapping include neural networks, dimensionality reduction on the word co-occurrence matrix, probabilistic models, explainable knowledge base method, and explicit representation in terms of the context in which words appear.

Word and phrase Embedding, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing and sentiment analysis.

One of the main limitations of word Embedding (word vector space models in general) is that possible meanings of a word are conflated into a single representation (a single vector in the semantic space). Sense Embedding are proposed as a solution to this problem: individual meanings of words are represented as distinct vectors in the space.





## METHODOLOGIES USED IN SENTIMENT ANALYSIS

A wide range of tools and techniques can be employed to tackle the goals described in the previous section. This section therefore describes some of the most common and widely used ones.

### Classification

: Many of the tasks in Sentiment Analysis can be thought of as classification. Machine Learning offers many algorithms designed to under-take that, but this task of classifying text according to its sentiment presents many

unique challenges. These can be formulated in one question: “What kinds of features do we use?”

### Term Frequency or Presence

: Traditional Information Retrieval systems have long emphasized the importance of term frequency. The widely used TF-IDF (Term Frequency - Inverse Document Frequency) measure is well-used in modelling documents according to Jones. TF-IDF is a measure of how is the co-currencies of a given word. The intuition is that terms that often appear in the document but seldom in the whole collection are more informative as to what the document is about as compared to the terms mentioned just once. TF-IDF have been shown to be quiet effective in sentiment classification In the field of Sentiment Analysis we find that instead of paying

attention to most frequent terms, it is more beneficial to seek out the most unique ones. Pang et al improved the performance of this system using term presence instead of frequency. Wiebe and Hoffman states in their paper that, “apparently people are creative when they are being opinionated”, implying the importance of low

-frequency terms in opinionated texts.

### **n-grams**

: Term positions are also important in document representation for Sentiment Analysis. The position of terms determines, and sometimes reverses, the polarity of the phrase. So, position information is sometimes encoded into the feature vector. Wiebe and Hoffman selects n-grams( $n=1,2,3,4$ ) based on precision calculated using annotated documents. The n-grams are a word-stem, part-of-speech pair, for instance (in-prep the-det can- noun) is a 3-gram.

### **Part-of-Speech:**

Adjectives are a good indicator of sentiment in text, and in the past decade they have been commonly exploited in Sentiment Analysis. This is true for other fields in textual analysis, since part-of-speech tags can be considered to be a crude form of word sense disambiguation.. In his work, Turney used part-of-speech patterns, to including an adjective and even went further to used adverb as well, for sentiment detection at the document level. Syntax information has also been used in feature sets, though there is still discussion about the advantages of this information in Sentiment classification (Pang & Lee, 2008). This information however may include important text features such as negation, intensifiers, and diminishes used sub tree-based boosting algorithm with dependency tree-based features for polarity classification, and show that it outperforms the bag-of-words baseline.

### **Negations**

: Negations have been long known to be integral in Sentiment Analysis. The usual bag-of-words representation of text disconnects all of the words, and considers sentences like “I like this car” and “I don’t like this car” very similar, since only

one word distinguishes one from the other. But when talking about sentiment, a negation changes the polarity of a whole phrase. Negations are often considered in post-processing of results, while the original representation of text ignores them (Hu

& Liu,2005), one could explicitly include the negation in the document representation by

appending them to the terms that are close to negations; for example term “like

-NOT” would be extracted from “I don’t like this book” (Pang & Lee, 2008). Though using co

-location may be too crude a technique. It would be incorrect to negate the sentiment in a sentence such as “No wonder everyone loves this car”. To handle such cases. use specific part-of-speech tags patterns to identify the negations relevant to the sentiment polarity of a phrase

## **CHAPTER 3**

### **Application Environment**

#### **(i) HARDWARE REQUIREMENT**

• <b>Operating System</b>	Window 7, Window 8,8.1 etc
• <b>PROCESSOR</b>	Dual Core
• <b>RAM</b>	Minimum 4GB
• <b>STORAGE</b>	10-20 GB recommended

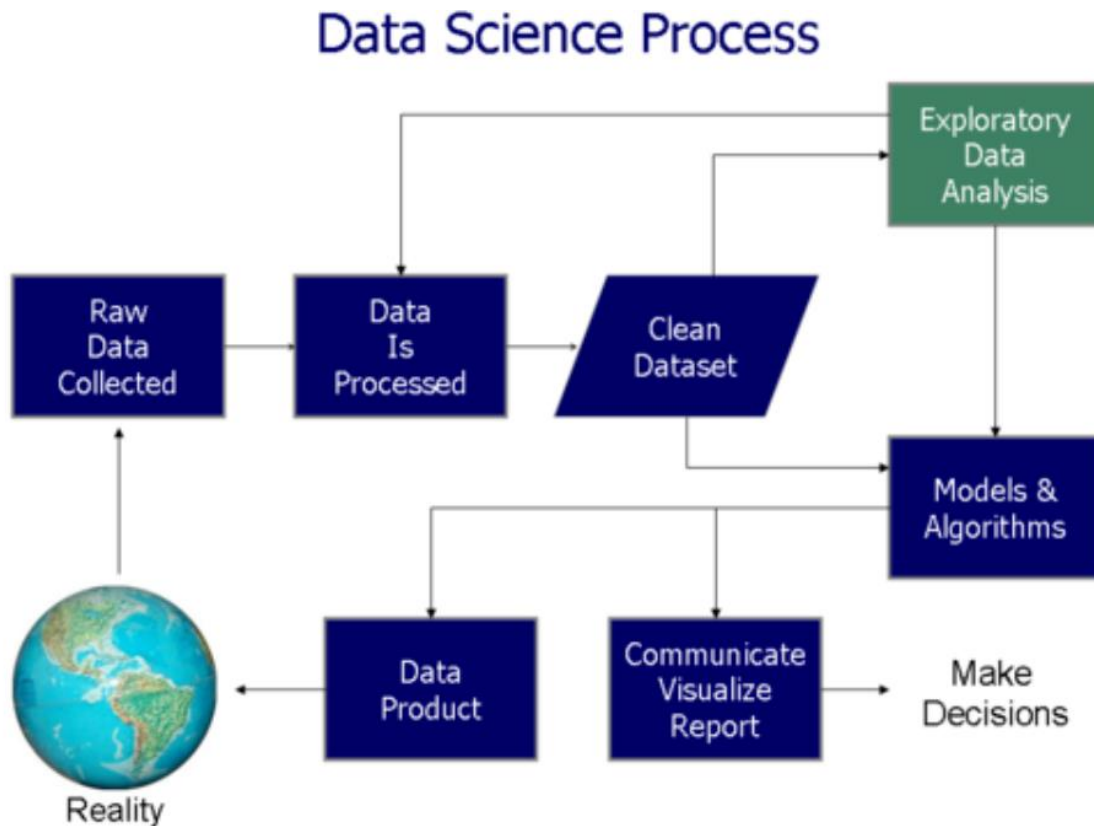
#### **(ii) SOFTWARE REQUIREMENT**

- **Jupyter Notebook** The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning, transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.
- **Visual Studio Code** It is an open Source Code Editor developed and managed by Microsoft. It provides the support of various languages and also implemented with internal Intelligence code completion feature of all the supported languages.
- **My-SQL** My SQL Server is a relational database management system developed by MySQL AB. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet).
- **Virtual Environment** If the application is to be run on the localhost then it is recommended to use virtual environment so that, the required dependencies do not mess with the other programs.

## CHAPTER 4

### 4.1) DESIGN METHODOLOGIES

#### 4.1.1 The Data Science Process



#### **Step 1: Frame the problem**

The first thing you have to do before you solve a problem is to define exactly what it is. You need to be able to translate data questions into something actionable.

You'll often get ambiguous inputs from the people who have problems. You'll have to develop the intuition to turn scarce inputs into actionable outputs—and to ask the questions that nobody else is asking.

Say you're solving a problem for the VP Sales of your company. You should start by understanding their goals and the underlying why behind their data questions. Before you can start thinking of solutions, you'll want to work with them to clearly define the problem.

A great way to do this is to ask the right questions.

You should then figure out what the sales process looks like, and who the customers are. You need as much context as possible for your numbers to become insights.

You should ask questions like the following:

Who are the customers?

Why are they buying our product?

How do we predict if a customer is going to buy our product?

What is different from segments who are performing well and those that are performing below expectations?

How much money will we lose if we don't actively sell the product to these groups?

In response to your questions, the VP Sales might reveal that they want to understand why certain segments of customers have bought less than expected. Their end goal might be to determine whether to continue to invest in these segments, or de-prioritize them. You'll want to tailor your analysis to that problem, and unearth insights that can support either conclusion.

It's important that at the end of this stage, you have all of the information and context you need to solve this problem.

## **Step 2: Collect the raw data needed for your problem**

Once you've defined the problem, you'll need data to give you the insights needed to turn the problem around with a solution. This part of the process involves thinking through what data you'll need and finding ways to get that data, whether it's querying internal databases, or purchasing external datasets.

You might find out that your company stores all of their sales data in a CRM or a customer relationship management software platform. You can export the CRM data in a CSV file for further analysis.

### **Step 3: Process the data for analysis**

Now that you have all of the raw data, you'll need to process it before you can do any analysis. Oftentimes, data can be quite messy, especially if it hasn't been well-maintained. You'll see errors that will corrupt your analysis: values set to null though they really are zero, duplicate values, and missing values. It's up to you to go through and check your data to make sure you'll get accurate insights.

You'll want to check for the following common errors:

Missing values, perhaps customers without an initial contact date

Corrupted values, such as invalid entries

Timezone differences, perhaps your database doesn't take into account the different timezones of your users

Date range errors, perhaps you'll have dates that makes no sense, such as data registered from before sales started

You'll need to look through aggregates of your file rows and columns and sample some test values to see if your values make sense. If you detect something that doesn't make sense, you'll need to remove that data or replace it with a default value. You'll need to use your intuition here: if a customer doesn't have an initial contact date, does it make sense to say that there was NO initial contact date? Or do you have to hunt down the VP Sales and ask if anybody has data on the customer's missing initial contact dates?

Once you're done working with those questions and cleaning your data, you'll be ready for exploratory data analysis (EDA).

### **Step 4: Explore the data**

When your data is clean, you'll should start playing with it!

The difficulty here isn't coming up with ideas to test, it's coming up with ideas that are likely to turn into insights. You'll have a fixed deadline for your data science project (your VP Sales is probably waiting on your analysis eagerly!), so you'll have to prioritize your questions. ‘

You'll have to look at some of the most interesting patterns that can help explain why sales are reduced for this group. You might notice that they don't tend to be very active on social media, with few of them having Twitter or Facebook accounts. You might also notice that most of them are older than your general audience. From that you can begin to trace patterns you can analyse more deeply.

### **Step 5: Perform in-depth analysis**

This step of the process is where you're going to have to apply your statistical, mathematical and technological knowledge and leverage all of the data science tools at your disposal to crunch the data and find every insight you can.

In this case, you might have to create a predictive model that compares your underperforming group with your average customer. You might find out that the age and social media activity are significant factors in predicting who will buy the product.

If you'd asked a lot of the right questions while framing your problem, you might realize that the company has been concentrating heavily on social media marketing efforts, with messaging that is aimed at younger audiences. You would know that certain demographics prefer being reached by telephone rather than by social media. You begin to see how the way the product has been marketed is significantly affecting sales: maybe this problem group isn't a lost cause! A change in tactics from social media marketing to more in-person interactions could change everything for the better. This is something you'll have to flag to your VP Sales.

You can now combine all of those qualitative insights with data from your quantitative analysis to craft a story that moves people to action.

### **Step 6: Communicate results of the analysis**

It's important that the VP Sales understand why the insights you've uncovered are important. Ultimately, you've been called upon to create a solution throughout the data science process. Proper communication will mean the difference between action and inaction on your proposals.

You need to craft a compelling story here that ties your data with their knowledge. You start by explaining the reasons behind the underperformance



of the older demographic. You tie that in with the answers your VP Sales gave you and the insights you've uncovered from the data. Then you move to concrete solutions that address the problem: we could shift some resources from social media to personal calls. You tie it all together into a narrative that solves the pain of your VP Sales: she now has clarity on how she can reclaim sales and hit her objectives.

### **4.1.2 The NLP Process**

Let's look at a piece of text from Wikipedia:

*London is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia. It was founded by the Romans, who named it Londinium.*

#### **Step 1: Sentence Segmentation**

The first step in the pipeline is to break the text apart into separate sentences. That gives us this:

"London is the capital and most populous city of England and the United Kingdom."

"Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia."

"It was founded by the Romans, who named it Londinium."

We can assume that each sentence in English is a separate thought or idea. It will be a lot easier to write a program to understand a single sentence than to understand a whole paragraph.

Coding a Sentence Segmentation model can be as simple as splitting apart sentences whenever you see a punctuation mark. But modern NLP pipelines often use more complex techniques that work even when a document isn't formatted cleanly.

#### **Step 2: Word Tokenization**

Now that we've split our document into sentences, we can process them one at a time. Let's start with the first sentence from our document:

“London is the capital and most populous city of England and the United Kingdom.”

The next step in our pipeline is to break this sentence into separate words or tokens. This is called tokenization. This is the result:

“London”, “is”, “ the”, “capital”, “and”, “most”, “populous”, “city”, “of”, “England”, “and”, “the”, “United”, “Kingdom”, “.”

Tokenization is easy to do in English. We’ll just split apart words whenever there’s a space between them. And we’ll also treat punctuation marks as separate tokens since punctuation also has meaning.

### **Step 3: Text Lemmatization**

In English (and most languages), words appear in different forms. Look at these two sentences:

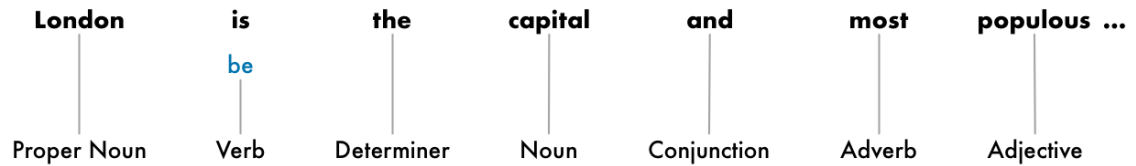
I had a pony.

I had two ponies.

Both sentences talk about the noun pony, but they are using different inflections. When working with text in a computer, it is helpful to know the base form of each word so that you know that both sentences are talking about the same concept. Otherwise the strings “pony” and “ponies” look like two totally different words to a computer.

In NLP, we call finding this process lemmatization—figuring out the most basic form or lemma of each word in the sentence.

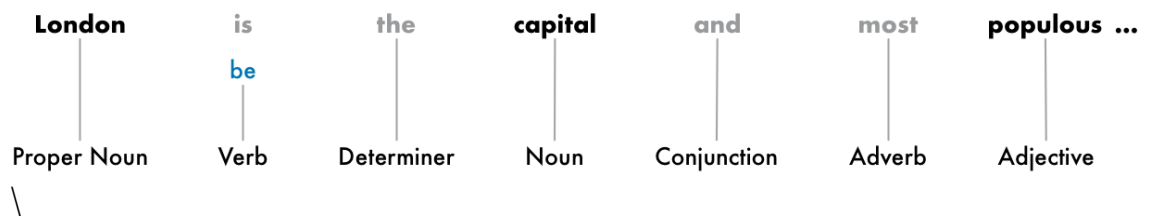
The same thing applies to verbs. We can also lemmatize verbs by finding their root, unconjugated form. So “I had two ponies” becomes “I [have] two [pony].”



## Step 4: Identifying Stop Words

Next, we want to consider the importance of each word in the sentence. English has a lot of filler words that appear very frequently like “and”, “the”, and “a”. When doing statistics on text, these words introduce a lot of noise since they appear way more frequently than other words. Some NLP pipelines will flag them as stop words—that is, words that you might want to filter out before doing any statistical analysis.

Here’s how our sentence looks with the stop words grayed out:



### 4.1.3 The Model Training Process



Machine learning is a field of computer science that gives computers the ability to learn without being programmed explicitly. The power of machine learning is that you can determine how to differentiate using models, rather than using human judgment. The basic steps that lead to machine learning and will teach you how it works are described below in a big picture:

#### 1. Gathering Data

Once you know exactly what you want and the equipments are in hand, it takes you to the first real step of machine learning- Gathering Data. This step is very crucial as the quality and quantity of data gathered will directly determine how good the predictive model will turn out to be. The data collected is then tabulated and called as Training Data.

## **2. Data Preparation**

After the training data is gathered, you move on to the next step of machine learning: Data preparation, where the data is loaded into a suitable place and then prepared for use in machine learning training. Here, the data is first put all together and then the order is randomized as the order of data should not affect what is learned.

This is also a good enough time to do any visualizations of the data, as that will help you see if there are any relevant relationships between the different variables, how you can take their advantage and as well as show you if there are any data imbalances present. Also, the data now has to be split into two parts. The first part that is used in training our model, will be the majority of the dataset and the second will be used for the evaluation of the trained model's performance. The other forms of adjusting and manipulation like normalization, error correction, and more take place at this step.

## **3. Choosing a model**

The next step that follows in the workflow is choosing a model among the many that researchers and data scientists have created over the years. Make the choice of the right one that should get the job done.

## **4. Training**

After the before steps are completed, you then move onto what is often considered the bulk of machine learning called training where the data is used to incrementally improve the model's ability to predict.

The training process involves initializing some random values for say A and B of our model, predict the output with those values, then compare it with the model's prediction and then adjust the values so that they match the predictions that were made previously.

This process then repeats and each cycle of updating is called one training step,

## **5. Evaluation**

Once training is complete, you now check if it is good enough using this step. This is where that dataset you set aside earlier comes into play. Evaluation allows the testing of the model against data that has never been seen and used for training and is meant to be representative of how the model might perform when in the real world.

## **6. Parameter Tuning**

Once the evaluation is over, any further improvement in your training can be possible by tuning the parameters. There were a few parameters that were implicitly assumed when the training was done. Another parameter included is the learning rate that defines how far the line is shifted during each step, based on the information from the previous training step. These values all play a role in the accuracy of the training model, and how long the training will take.

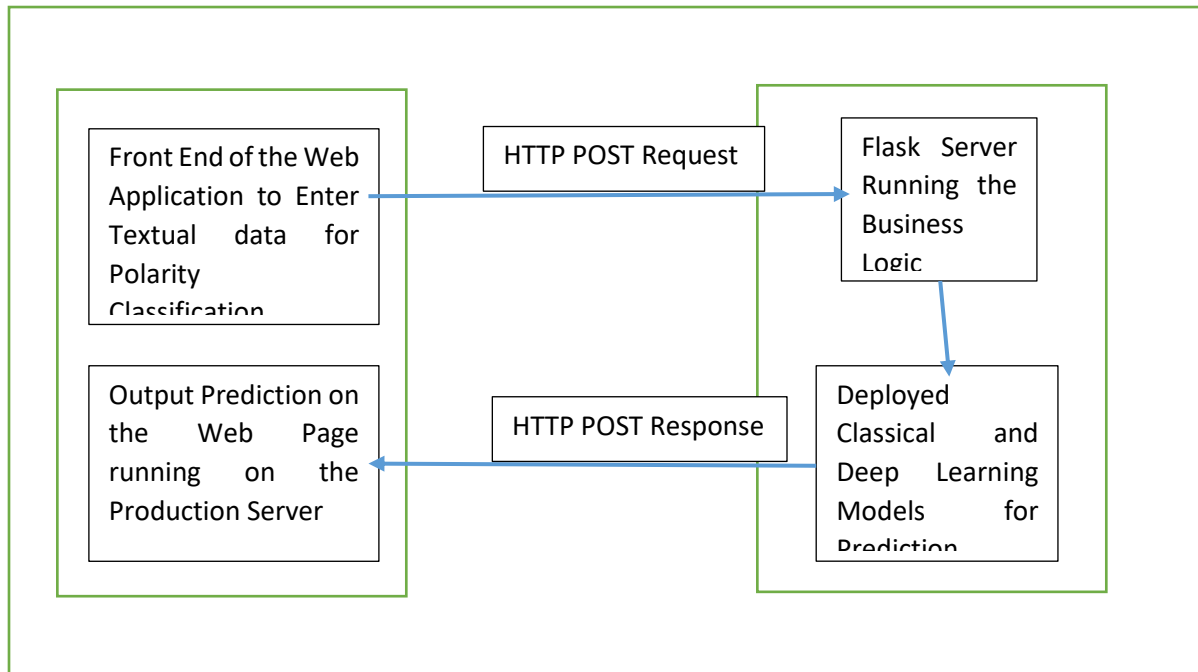
For models that are more complex, initial conditions play a significant role in the determination of the outcome of training. Differences can be seen depending on whether a model starts off training with values initialized to zeroes versus some distribution of values, which then leads to the question of which distribution is to be used. Since there are many considerations at this phase of training, it's important that you define what makes a model good. These parameters are referred to as Hyper parameters. The adjustment or tuning of these parameters depends on the dataset, model, and the training process. Once you are done with these parameters and are satisfied you can move on to the last step.

## **7. Prediction**

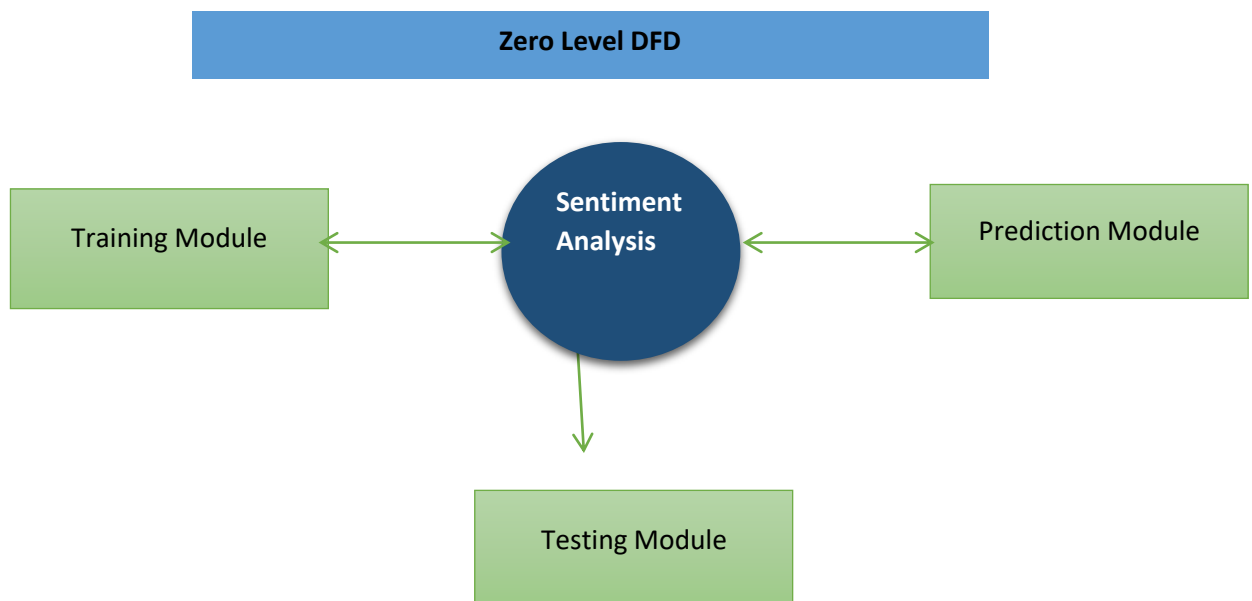
Machine learning is basically using data to answer questions. So this is the final step where you get to answer few questions. This is the point where the value of machine learning is realized. Here you can finally use your model to predict the outcome of what you want.

The above-mentioned steps take you from where you create a model to where you Predict its output and thus acts as a learning path.

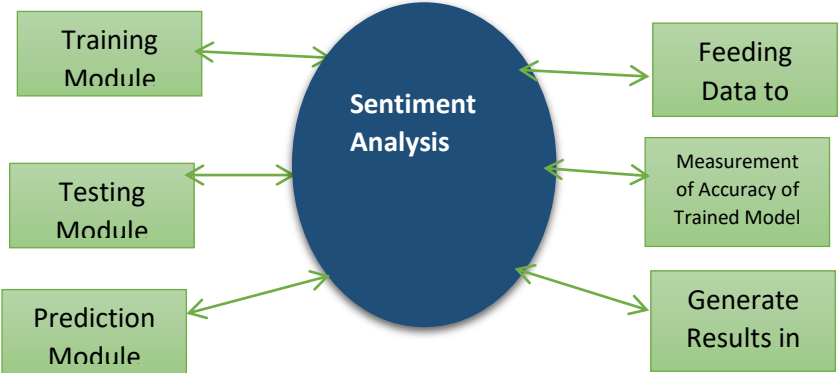
## 4.2 Architecture of the Project



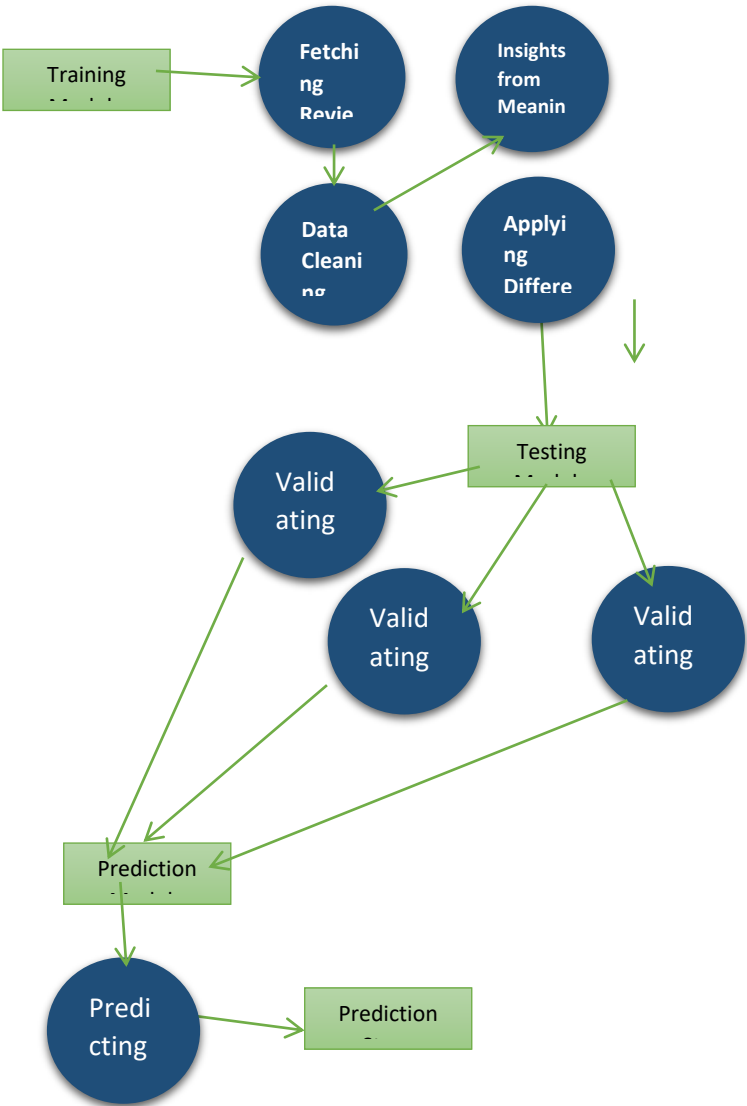
## 4.3 DFD for the Project



1<sup>st</sup> Level DFD



2<sup>nd</sup> Level DFD



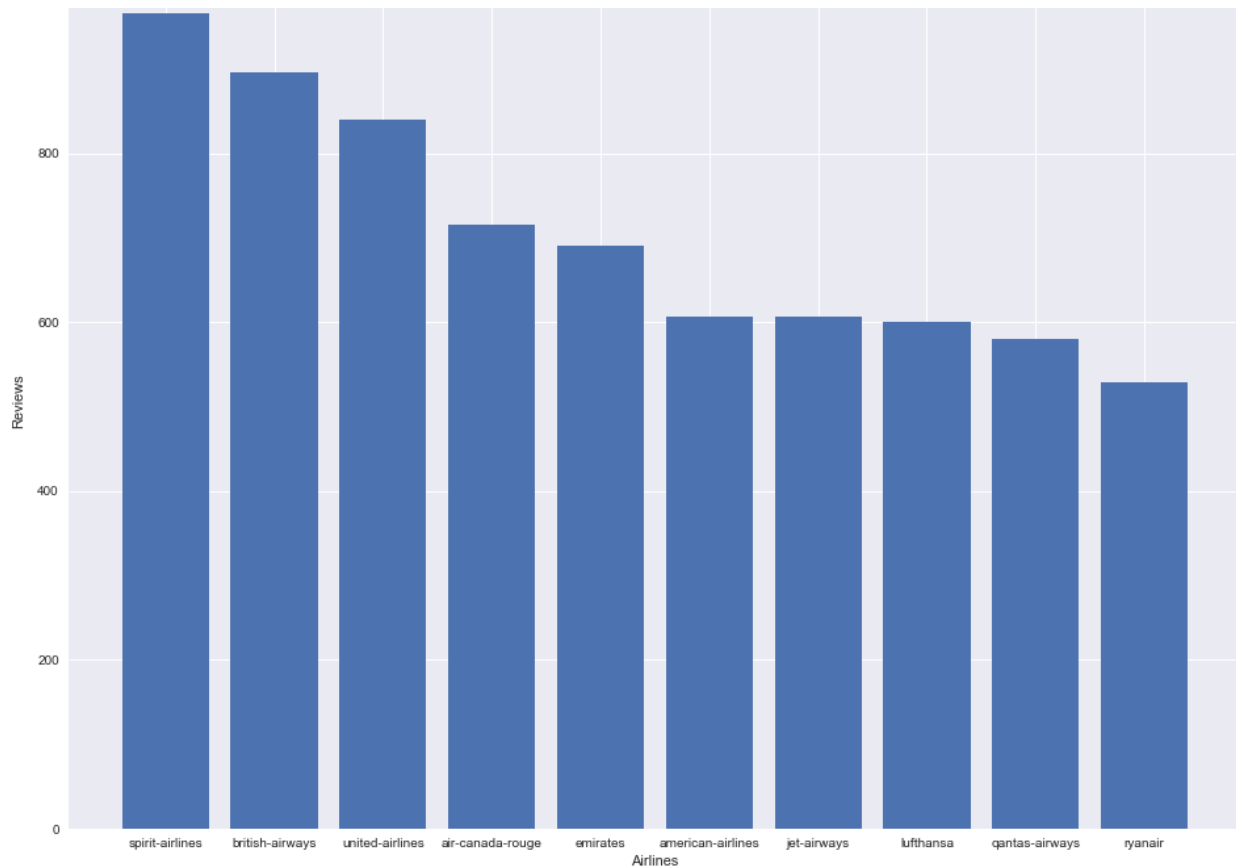
# CHAPTER 5

## 5.1 Implementation

### 5.1.1) Exploring Data

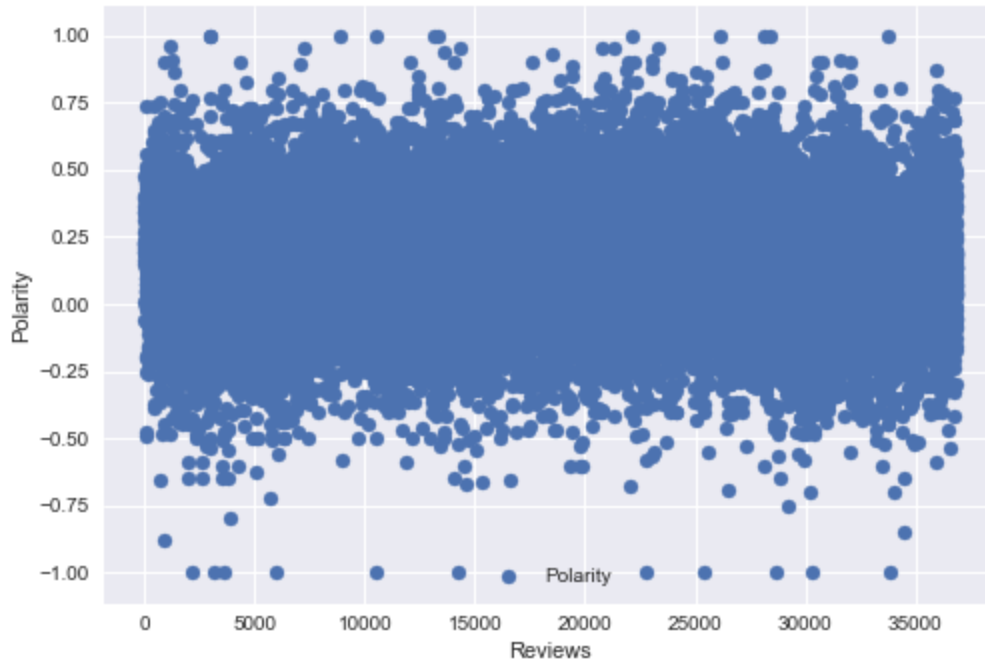
The review data is collected from the skytraxratings.com online freely available repository. This data is then processed and explored before diving into the project.

The observed top 10 most reviewed airlines according to the received data:



Since the data received is not labelled with polarities so by using the textblob library from python we first determined the polarities of each review on the scale of -1 to 1 and then assigned the review classification in three categories i.e -1 -> Negative, 0 -> Neutral and 1 -> Positive Review. On the basis of these categories the further investigation is carried out.





After this the average positive, average neutral and average negative reviews of each airline is calculated and visualize using the matplotlib library of python in the form of bar chart for the further data science process. This will provide the clear picture about the reputation of each airline that is being used for the data analysis.

Scatterplot of the polarities of the reviews give the picture about the reviews that they are uniformly divided on the scale of -1 to 1. The total of 36861 reviews are analysed for 357 different airlines.

I consider the polarity calculation as the best method to assign the sentiment value to the reviews instead of provided average rating because in the data repository above 90% of average rating data was missing and the provided average rating can't be trusted to train the classical as well as deep neural networks.



### **5.1.2) Pre-processing Reviews**

Before training the model pre-processing is main step which involves the removal of punctuation symbols, unnecessary white spaces and stop words from the reviews to make the training of the model effective and efficient. For this purpose nltk (Natural Language Toolkit) library for natural language processing is used.

```
# Remove Numbers
reviews1=[]
for i in reviews:
    reviews1.append(re.sub(r'\d+', '', i))

# Remove Punctuation
reviews2=[]
for i in reviews1:
    table = str.maketrans({key: None for key in string.punctuation})
    reviews2.append(i.translate(table))

# Remove Whitespaces
reviews3=[]
for i in reviews2:
    reviews3.append(i.strip())

# Removing Stop Words
# stop_words = set(stopwords.words('english'))
tokenize_reviews_no_stop_words=[]
for i in reviews3:
    tokens=tokenize(i)
    result=[j for j in tokens if not j in stop_words]
    temp=' '.join(result)
    tokenize_reviews_no_stop_words.append(temp)
```

### **5.1.3) Training Models.**

For this project, I have worked on two classical models i.e. Logistic Regression and Linear Support Vector Machine and two deep neural network, Convolutional Neural Network and Bidirectional Long short term memory network. For training purpose, In classical models words are represented as Vectors using CountVector() and Word Embeddings in case of deep neural network.

### **5.1.4) Deploying Models on Heroku**

These trained models are then deployed on the Heroku Cloud platform for making its access to public for testing purpose. Flask Framework is used for the development of the web application with 4 view pages where textual data is entered for prediction. The flask server uses the gunicorn framework for controlling the routing of the application to the different pages.

```
from flask import Flask,render_template,request,redirect,url_for
import os
import flask
import pickle
import re
import string
# from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from keras.preprocessing.sequence import pad_sequences
from keras import backend as K
import numpy as np
app=Flask(__name__)
app.static_folder='static'
stop_words={'yourselves', "aren't", "it's", 'again', 'whom', 'of', "she's", 'up',
'will', 'wouldn', 'which', 'before', 'ma', 'didn', 'no', 'just', 'this', 'my',
'same', 'can', "doesn't", 'hasn', 'her', 'who', 'those', 'had', "shan't",
'weren', 'has', 'o', 'if', "mightn't", "that'll", 'couldn', 'have', 'each',
'been', 'll', 'too', 'out', "you'll", 'but', 'there', 'such', 'doing', 'with',
'is', 'y', 'am', 'ourselves', 'having', 'here', 'haven', 'himself', 'where',
'we', 'itself', 'now', 'myself', 'until', "should've", 'above', "don't", 'were',
'aren', 'me', 'ours', 'was', 'once', 'more', "haven't", 'while', 'shan', 'won',
'then', 'd', 'into', 'from', 'at', 'isn', 'both', 'own', 'your', 'does',
'mightn', 'other', "wouldn't", 'why', "won't", 'during', 'through', 'few', 'he',
's', 'doesn', 'by', 'when', 'or', 'than', 'she', 'so', 'over', 'do', 'in',
'some', "you're", 'further', 'shouldn', "wasn't", 'to', 'how', 'nor', 'down',
'it', 're', 'did', 'because', "weren't", "you'd", 'm', 'their', 'for', 'most',
'wasn', "didn't", 'and', 'after', "mustn't", 'needn', 'on', "hadn't", 'any',
'only', 'are', "hasn't", 'an', 'yourself', 'herself', 'that', 'a', 'ain', 'be',
'them', 'under', 'you', "you've", 'his', 'against', 'very', 'theirs',
'themselves', 'being', 've', "shouldn't", 'as', 'don', 'him', 'mustn',
"couldn't", 'hers', "isn't", 'its', 'these', 'between', 'off', 'they', 't',
'below', 'all', 'not', 'hadn', 'the', 'our', 'about', "needn't", 'i', 'what',
'should', 'yours'}
```

```
def preprocess_review(sentence):
    result=re.sub(r'\d+', '',sentence)
    table = str.maketrans({key: None for key in string.punctuation})
```

```

result_new=result.translate(table)
result_new=result_new.strip()

#     stop_words=set(stopwords.words('english'))
tokens=word_tokenize(result_new)
final=[j for j in tokens if not j in stop_words]
temp=' '.join(final)
return temp

def prediction_logistic_regression(sentence):
    print(sentence)
    loaded_model=pickle.load(open('logisticModel0.1.pkl','rb'))
    vectorizer=pickle.load(open('logisticVectorizer0.1.pkl','rb'))
    sen=preprocess_review(sentence[0])
    t=vectorizer.transform([sen])
    sentimentresult=loaded_model.predict(t)
    prob=loaded_model.predict_proba(t)
    final=[]
    final.append(sentimentresult[0])
    final.append(prob[0][0]*100)
    final.append(prob[0][1]*100)
    final.append(prob[0][2]*100)

    return final

def prediction_linear_svm(sentence):
    loaded_model=pickle.load(open('svmModel0.1.pkl','rb'))
    vectorizer=pickle.load(open('logisticVectorizer0.1.pkl','rb'))
    sen=preprocess_review(sentence[0])
    t=vectorizer.transform([sen])
    sentimentresult=loaded_model.predict(t)
    prob=loaded_model.predict_proba(t)
    final=[]
    final.append(sentimentresult[0])
    final.append(prob[0][0]*100)
    final.append(prob[0][1]*100)
    final.append(prob[0][2]*100)

    return final

def prediction_lstm(sentence):
    final=[]

```

```

loaded_model=pickle.load(open('lstmModel0.1.pkl','rb'))
token=pickle.load(open('lstmToken.pkl','rb'))
s=preprocess_review(sentence[0])
sen=token.texts_to_sequences([s])
padded_docs = pad_sequences(sen, maxlen=38, padding='post')
proba=loaded_model.predict(padded_docs)
result=np.argmax(proba)
if result==0:
    final.append(-1)
if result==1:
    final.append(0)
if result==2:
    final.append(1)
# K.clear_session()
# loaded_model=pickle.load(open('lstmModel0.1.pkl','rb'))
# proba=loaded_model.predict(padded_docs)
final.append(proba[0][0]*100)
final.append(proba[0][1]*100)
final.append(proba[0][2]*100)
K.clear_session()
return final

```

```

def prediction_cnn(sentence):
    final=[]
    loaded_model=pickle.load(open('cnnModel0.1.pkl','rb'))
    token=pickle.load(open('lstmToken.pkl','rb'))
    sen=preprocess_review(sentence[0])
    sen=token.texts_to_sequences([s])
    padded_docs = pad_sequences(sen, maxlen=38, padding='post')
    proba=loaded_model.predict(padded_docs)
    result=np.argmax(proba)
    if result==0:
        final.append(-1)
    if result==1:
        final.append(0)
    if result==2:
        final.append(1)
    # K.clear_session()
    # loaded_model=pickle.load(open('lstmModel0.1.pkl','rb'))
    # proba=loaded_model.predict(padded_docs)
    final.append(proba[0][0]*100)
    final.append(proba[0][1]*100)
    final.append(proba[0][2]*100)

```

```

        K.clear_session()
        return final

@app.route('/')
def index():
    return flask.render_template('index.html')

@app.route('/logistic')
def logistic():
    return flask.render_template('logistic.html')

@app.route('/logisiticrosult',methods=['POST'])

def logisitc_result():
    if(request.method=='POST'):
        sentence=request.form.to_dict()
        sentence=list(sentence.values())
        result=prediction_logistic_regression(sentence)
        sentiment=result[0]
        sentimentresult=''
        if(sentiment==-1):
            sentimentresult='Negative'
        elif(sentiment==0):
            sentimentresult='Neutral'
        elif(sentiment==1):
            sentimentresult='Positive'

        pos_proba=result[3]
        neut_proba=result[2]
        neg_proba=result[1]
        pos_proba=round(pos_proba,2)
        neut_proba=round(neut_proba,2)
        neg_proba=round(neg_proba,2)
        pos_proba=str(pos_proba)+' %'
        neut_proba=str(neut_proba)+ ' %'
        neg_proba=str(neg_proba)+ ' %'

        return
    flask.render_template('logistic_result.html',sentence=sentence[0],sentimentresult=
    =sentimentresult,pos_proba=pos_proba,neg_proba=neg_proba,neut_proba=neut_proba)

@app.route('/svm')
def svm():
    return flask.render_template('svm.html')

```

```

@app.route('/svmresult',methods=['POST'])
def svm_result():
    if(request.method=='POST'):
        sentence=request.form.to_dict()
        sentence=list(sentence.values())
        result=prediction_linear_svm(sentence)
        sentiment=result[0]
        sentimentresult=''
        if(sentiment==-1):
            sentimentresult='Negative'
        elif(sentiment==0):
            sentimentresult='Neutral'
        elif(sentiment==1):
            sentimentresult='Positive'

        pos_proba=result[3]
        neut_proba=result[2]
        neg_proba=result[1]
        pos_proba=round(pos_proba,2)
        neut_proba=round(neut_proba,2)
        neg_proba=round(neg_proba,2)
        pos_proba=str(pos_proba)+' %'
        neut_proba=str(neut_proba)+ ' %'
        neg_proba=str(neg_proba)+ ' %'

        return

    flask.render_template('svm_result.html',sentence=sentence[0],sentimentresult=sentimentresult,pos_proba=pos_proba,neg_proba=neg_proba,neut_proba=neut_proba)

@app.route('/lstm')
def lstm():
    return flask.render_template('lstm.html')

@app.route('/lstmresult',methods=['POST'])

def lstm_result():
    if(request.method=='POST'):
        sentence=request.form.to_dict()
        sentence=list(sentence.values())
        result=prediction_lstm(sentence)
        sentiment=result[0]
        sentimentresult=''
        if(sentiment==-1):
            sentimentresult='Negative'

```



```

        elif(sentiment==0):
            sentimentresult='Neutral'
        elif(sentiment==1):
            sentimentresult='Positive'

        pos_proba=result[3]
        neut_proba=result[2]
        neg_proba=result[1]
        pos_proba=round(pos_proba,2)
        neut_proba=round(neut_proba,2)
        neg_proba=round(neg_proba,2)
        pos_proba=str(pos_proba)+' %'
        neut_proba=str(neut_proba)+ ' %'
        neg_proba=str(neg_proba)+ ' %'

    return

flask.render_template('lstm_result.html',sentence=sentence[0],sentimentresult=sentimentresult,pos_proba=pos_proba,neg_proba=neg_proba,neut_proba=neut_proba)

@app.route('/cnn')
def cnn():
    return flask.render_template('cnn.html')

@app.route('/cnnresult',methods=['POST'])
def cnn_result():
    if(request.method=='POST'):
        sentence=request.form.to_dict()
        sentence=list(sentence.values())
        result=prediction_cnn(sentence)
        sentiment=result[0]
        sentimentresult=''
        if(sentiment==-1):
            sentimentresult='Negative'
        elif(sentiment==0):
            sentimentresult='Neutral'
        elif(sentiment==1):
            sentimentresult='Positive'

        pos_proba=result[3]
        neut_proba=result[2]
        neg_proba=result[1]
        pos_proba=round(pos_proba,2)
        neut_proba=round(neut_proba,2)

```

```

neg_proba=round(neg_proba,2)
pos_proba=str(pos_proba)+' %'
neut_proba=str(neut_proba)+' %'
neg_proba=str(neg_proba)+' %'

return

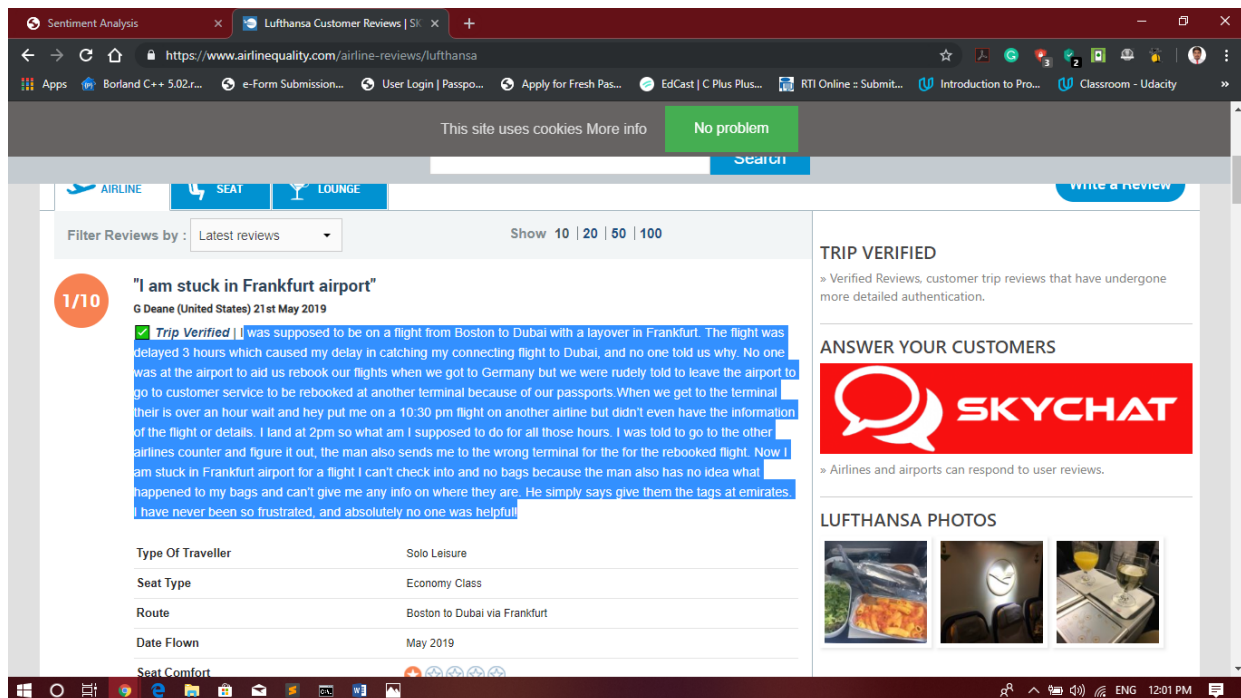
flask.render_template('cnn_result.html',sentence=sentence[0],sentimentresult=sentimentresult,pos_proba=pos_proba,neg_proba=neg_proba,neut_proba=neut_proba)

@app.route('/aboutus')
def aboutus():
    return flask.render_template('aboutus.html')

if (__name__=='__main__'):
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, port=port)

```

### 5.1.4) View Pages for Different Models






As it can be seen from the original source of the Review that it is a negative review and rating given is 1/10.

This review is fed to the trained model

The screenshot shows a web browser window with the URL <https://knowyoursentiment.herokuapp.com/logistic>. The page has a navigation bar with links: Sentiment Analyzer, Home, Logistic Regressor, Linear SVM Model, CNN, LSTM, and About Us. The main heading is "Logistic Regression" with the subtitle "A Project to Determine the Emotion Polarity from the Service Quality Reviews of Air Transport". Below this, there is a section titled "Logistic Emotion Polarity Analyzer" with the instruction "Enter the Airline Review.". A text area contains the following review: "I was supposed to be on a flight from Boston to Dubai with a layover in Frankfurt. The flight was delayed 3 hours which caused my delay in catching my connecting flight to Dubai, and no one told us why. No one was at the airport to aid us rebook our flights when we got to Germany but we were rudely told to leave the airport to go to customer service to be rebooked at 6". A red "6" is visible at the end of the text. Below the text area is a blue button labeled "Find Sentiment". To the right of the input area is a large empty box labeled "Result".

The screenshot shows the same web browser window as above, but now the "Result" section displays the output. The word "Negative" is centered at the top of the result area. Below it is a table with two columns: "Sentiment" and "Confidence".

Sentiment	Confidence
	0.96 %
	0.24 %
	98.8 %

Output Webpage showing the classification of the review and Emotion polarity emoticons.

## Linear Support Vector Machine

The screenshot shows a web browser window with the URL <https://knowyoursentiment.herokuapp.com/svmresult>. The page has a navigation bar with links: Review Sentiment Analyzer, Home, Logistic Regressor, Linear SVM Model, CNN, LSTM, and About Us. The main heading is "Linear Support Vector Machine" with the subtitle "A Project to Determine the Emotion Polarity from the Service Quality Reviews of Air Transport".

On the left, a box titled "Logistic Emotion Polarity Analyzer" contains the text: "Enter the Airline Review." and a text area with a sample review: "I was supposed to be on a flight from Boston to Dubai with a layover in Frankfurt. The flight was delayed 3 hours which caused my delay in catching my connecting flight to Dubai, and no one told us why. No one was at the airport to aid us rebook our flights when we got to Germany but we were rudely told to leave the airport to go to customer service to be rebooked at". Below the text area is a "Find Sentiment" button.

On the right, a "Result" box displays the classification: "Negative". Below this, a table shows the sentiment and confidence levels:

Sentiment	Confidence
	5.06 %
	4.21 %
	90.73 %

## Original Source for Positive Review

The screenshot shows a web browser window with the URL <https://www.airlinequality.com/airline-reviews/lufthansa>. The page has a navigation bar with links: Apps, Borland C++ 5.02.r..., e-Form Submission..., User Login | Passpo..., Apply for Fresh Pas..., EdCast | C Plus Plus..., RTI Online :: Submit..., Introduction to Pro..., and Classroom - Udacity. The main heading is "Overall highly recommended" with a rating of 9/10 and 27 reviews. The review is by Michael Schade (Germany) on 21st May 2019. The review text is: "Trip Verified | Hamburg to Istanbul via Frankfurt. Both flights on time, typical German efficiency. Flight attendants very friendly, decent seat pitch and even a proper meal on the longer flights. Only the transfer in Frankfurt through that dreadful tunnel wasn't something I want to experience more often. Only if you are fit you will be able to make your connecting flight with just 45 minutes transfer time. Overall, highly recommended".

Below the review, a table shows flight details:

Field	Value
Aircraft	A319/320
Type Of Traveller	Solo Leisure
Seat Type	Economy Class
Route	Hamburg to Istanbul via Frankfurt
Date Flown	May 2019
Seat Comfort	★★★★☆
Cabin Staff Service	★★★★★
Food & Beverages	★★★★☆
Ground Service	★★★★★
Value For Money	★★★★☆

# Convolutional Neural Network

The screenshot shows a web browser window with the URL <https://knowyoursentiment.herokuapp.com/cnnresult>. The page has a navigation bar with links: Review Sentiment Analyzer, Home, Logistic Regressor, Linear SVM Model, CNN, LSTM, and About Us. The main heading is "Convolutional Neural Network" with the subtitle "A Project to Determine the Emotion Polarity from the Service Quality Reviews of Air Transport".

The interface is divided into two main sections: "CNN Emotion Polarity Analyzer" and "Result".

**CNN Emotion Polarity Analyzer:** It contains a text input area with the sample review: "Hamburg to Istanbul via Frankfurt. Both flights on time, typical German efficiency. Flight attendants very friendly, decent seat pitch and even a proper meal on the longer flights. Only the transfer in Frankfurt through that dreadful tunnel wasn't something I want to experience more often. Only if you are fit you will be able to make your connecting flight with just 45 minutes transfer". Below the input is a "Find Sentiment" button.

**Result:** It displays the sentiment analysis results for the input text.

Positive	
Sentiment	Confidence
	99.07 %
	0.0 %
	0.93 %

# Long Short Term Memory Network

The screenshot shows a web browser window with the URL <https://knowyoursentiment.herokuapp.com/lstmresult>. The page has a navigation bar with links: Review Sentiment Analyzer, Home, Logistic Regressor, Linear SVM Model, CNN, LSTM, and About Us. The main heading is "LSTM" with the subtitle "A Project to Determine the Emotion Polarity from the Service Quality Reviews of Air Transport".

The interface is divided into two main sections: "LSTM Emotion Polarity Analyzer" and "Result".

**LSTM Emotion Polarity Analyzer:** It contains a text input area with the same sample review as the CNN model: "Hamburg to Istanbul via Frankfurt. Both flights on time, typical German efficiency. Flight attendants very friendly, decent seat pitch and even a proper meal on the longer flights. Only the transfer in Frankfurt through that dreadful tunnel wasn't something I want to experience more often. Only if you are fit you will be able to make your connecting flight with just 45 minutes transfer". Below the input is a "Find Sentiment" button.

**Result:** It displays the sentiment analysis results for the input text.

Positive	
Sentiment	Confidence
	97.05 %
	0.05 %
	2.89 %

# **CHAPTER 6**

## **6.1 TECHNOLOGIES USED**

### **6.1.1 Presentation View/Front End Development Technologies**

- **Python-Tkinter (GUI)**

Tkinter is not the only Gui Programming toolkit for Python. It is however the most commonly used one. Cameron Laird calls the yearly decision to keep TkInter "one of the minor traditions of the Python world.". Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

- **HTML5**

HTML5 is the latest and most enhanced version of HTML. Technically, HTML is not a programming language, but rather a mark up language. The term represents two different concepts. It is a new version of the language HTML, with new elements, attributes, and behaviours, **and** a larger set of technologies that allows the building of more diverse and powerful Web sites and applications. This set is sometimes called HTML5 & friends and often shortened to just HTML5.

- **CSS**

CSS is used to control the style of a web document in a simple and easy way. CSS is the acronym for "Cascading Style Sheet". It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based

markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments.

- **JavaScript**

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

- **jQuery**

jQuery is a JavaScript library that allows web developers to add extra functionality to their websites. It is open source and provided for free under the MIT license. In recent years, jQuery has become the most popular JavaScript library used in web development.

- **Bootstrap**

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

- **Adobe Photoshop**

Photoshop is unlike other common software interfaces which emulate virtual typewriters or graphing paper. Photoshop creates an artist's virtual studio/darkroom. When you open the program you see a toolbox on the left with tools you will use to manipulate your images, and on the right, a white square which is your "canvas" or work area. The grey area surrounding the canvas is not part of your image, but only defines its edges.

- **Corel Draw**

CorelDraw is a powerful graphic design tool that can be used for a wide range of creative projects. Whether you are a professional designer or a hobbyist looking to explore your creative side, the course will lay the foundations you need to get the most out of CorelDraw.

### **6.1.2 Business View/Logical View End Development Technologies**

- **Python 3.6**

**Python** is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.<sup>[26]</sup> Van Rossum led the language community until stepping down as leader in July 2018. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- **NumPy**

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.



- **Pandas**

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals

- **Flask**

**Flask** is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.<sup>[3]</sup> It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

- **SciKit**

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

Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

- **Matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

- **NLTK**

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

- **TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

### **6.1.3 Database View/Back End Development Technologies**

- **MySQL**

MySQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet).

- **CSV**

**CSV** is a simple **file** format used to store tabular data, such as a spreadsheet or database. **Files** in the **CSV** format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or Open Office Calc. **CSV** stands for "comma-separated values"

#### **Heroku**

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages. Heroku was acquired by Salesforce.com in 2010 for \$212 million.

## **CHAPTER 7**

### **7.1) Result**

<b>Model</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-Score</b>	<b>Validation Accuracy</b>	<b>Word Representation</b>
Dense NN	0: .62 1: .89	0: .56 1: .91	0: .59 1: .90	83.8%	Word Embeddings
Convolution 1-D NN	0: .58 1: .89	0: .59 1: .89	0: .59 1: .89	82.7%	Word Embeddings
GRU	0: .58 1: .88	0: .52 1: .90	0: .55 1: .89	82.3%	Word Embeddings
LSTM	0: .63 1: .91	0: .66 1: .90	0: .64 1: .90	84.8%	Word Embeddings
Bi-Directional LSTM	0: .64 1: .91	0: .67 1: .90	0: .65 1: .91	85.2 %	Word Embeddings
Logistic Regression	0: .71 1: .87	0: .46 1: .95	0: .56 1: .91	84.8%	TF-IDF
Linear SVM	0: .69 1: .91	0: .66 1: .92	0: .67 1: .92	86.7%	TF-IDF
Naïve Bayes	0: 1.0 1: .8	0: .05 1: 1.0	0: .09 1: .89	80.2%	TF-IDF
Logistic Regression	0: .69 1: .90	0: .61 1: .93	0: .64 1: .91	86.1%	CountVector
Linear SVM	0: .70 1: .91	0: .64 1: .93	0: .67 1: .92	86.8%	CountVector
Naïve Bayes	0: .75 1: .88	0: .48 1: .96	0: .59 1: .92	85.9%	CountVector

In our models, we used three types of word/text representation techniques to feed data as input. Word Embedding are the best methods for word representation when the models being used are neural networks. They help us capture the context and help visualize the words in the multidimensional space. For the classical machine learning models such as logistic regression and naïve Bayes, we make use to two commonly used word representations called “TF-IDF Vector” and “Count Vector”.

In order to catch the context of up to two words, we make use of N-Gram approach and hence we take into consideration 1-2 gram. Increasing the words to be considered increased the vector representation length to exceedingly high which was costly on memory and that is the reason both these methods are only suitable for small datasets because as the vocabulary size increases the footprint of the vector used to represent data increases a lot and it becomes impractical to train classical models on such huge data.

From the results, it can be observed that Linear SVM still manages to outperform all the models in terms of validation accuracy. The precision and recall scores are also among the best for LinearSVM making it the most accurate model among the models being considered. Surprisingly, it manages to outperform even Bidirectional LSTM which is well known for its efficiency in handling textual data. This can be attributed to the fact that neural networks typically tend to perform well if the dataset is huge and the dataset under consideration has about 7000 reviews which is very small. On the other hand, SVM is considered the best performer on small datasets and hence clearly shows its prowess in the results.

If we consider neural networks, clearly, bidirectional LSTM and LSTM perform the best in the classification task both having a validation accuracy of about 85% which is very good considering the dataset is very small. Both LSTM and Bidirectional LSTM show very comparable precision and recall scores to the LinearSVM model which further shows the strength of LSTM on textual data. Other neural networks such as Convolution 1-D, GRU and Dense Neural Nets lag behind LSTM in terms of both validation accuracy score and the metrics (precision and recall) score.

It is conspicuous that CountVector representation of textual data consistently gave better results on all the classical machine learning models. Naïve Bayes particularly had a big jump on validation accuracy increasing from 80.2% to 85.9%. Logistic Regression had a jump of about 2% in validation accuracy when using CountVector model for word representation. Consequently, All the classical models displayed better precision and recall scores when TF-IDF was used to feed data.

Overall, the neural networks lag behind classical models in terms of validation accuracy, precision and recall score. This clearly shows that neural networks though being powerful need huge amount of data to show exemplary accuracies. The classical model, hence perform better on small datasets and they usually do not generalize well to huge datasets. The reason for classical models not generalizing well to huge datasets and also performing well on small datasets lie in

the fact that the representation of textual data used in these classical model allows them to perform well but as the data increases, it becomes difficult to use the word representation techniques used in these models as the memory footprint would be too large to be handled by such models. On the other hand, neural networks can easily handle large data by increasing the depth of network and hence they perform better on large datasets. In our case, we are targeting the domain of airline reviews and it makes sense that classical models perform well on a small dataset targeted at this particular domain.

## **7.2 Future Scope**

Transfer Learning is something that might help neural networks further improve accuracy. Pre trained word Embedding such as glove , Word2Vec by google are trained on huge datasets and hence can help improve the representation of the words.

This Web application project can be extended to the further aspect-term detection in the text which will be used to carry out opinion mining of the customers in regards to a specific service.

### **Business Point of View**

By knowing the emotions of the customers towards the service of the company, the organization will know where they are excelling and where they are lacking as compared to the other competitions in the market. This will help them to improve their service quality in order to provide ultimate customer satisfaction. The Company will immediately act upon the services or area of services if the reviews about that service is negative so that they can avoid the possible loss of the Customer Trust.

### **Customer Point Of View**

From the customer point of view before buying any service if they get the whole visualization of the service in the form of emotions of the different customers that had use the service prior helps them making a smart decision whether to buy the service or not avoiding the possible waste of their assets.

## **CHAPTER 8**

### **References**

1. <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714> [ LSTM ]
2. <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148> [CNN]
3. <https://spacy.io/usage/processing-pipelines> [ NLP Pipeline ]
4. <https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e> [ Steps of Machine Learning ]
5. <https://tutorialpoint.com/courses/flask>

