NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, BIHTA, PATNA

REPORT

ON

QUORA DUPLICATE QUESTION PAIR DETECTION

SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT

OF

BACHELOR OF TECHNOLOGY (COMPUTER SCIENCE & ENGINEERING)

FROM

ARYABHATTA KNOWLEDGE UNIVERSITY, PATNA, BIHAR

SUBMITTED BY:

Roushan Raj (171045)

Shivangi Srivastava (171027)

Misha Kiran(171051)

VII SEM

SESSION 2020-21

An ISO 9001:2008 Certified Institution

Approved by AICTE, New Delhi

Affiliated to Aryabhatta Knowledge University, Patna, Bihar

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, BIHTA, PATNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION 2020-21

ACKNOWLEDGEMENT

We would like to thank everyone who helped us to make this project. We would like to thank Dr. Sunil Saumya for his guidance and support throughout this project. It was really nice working under him as we got a chance to enhance our knowledge. We are very grateful to Assistent Professor of NSIT Patna Mr. Gopal Krishna and Mr. Subhash Chandra Pandit for being instrumental in the completion of our project. We would like to thank IIIT Dharwad for giving us this opportunity. Finally, we take this opportunity to extend our deep appreciation to our family and friends, for all that they meant to us during the crucial times of the completion of our project. Last but not the least we would like to thank God for his blessings.

Roushan Raj (171045)

Date: 31st August 2020 Shivangi Srivastava (171027)

Misha Kiran(171051)

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, BIHTA, PATNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION 2020-21

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We declare that we have properly and accurately acknowledged all sources used in the production of this report. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Roushan Raj (171045)

Date: 31st August 2020

Shivangi Srivastava (171027)

Misha Kiran(171051)

Abstract

Quora is the most popular question-answer platform where users can create and edit answers and also comment on questions and its answers. The negative side of a question-answer platform is that Quora has to deal with repeated questions and answers which is a difficult task. To determine duplicate or similar questions, natural language processing is the most effective method. The cost of miss classification of duplicate questions is very high. Identifying semantically identical questions on, Question and Answering(Q&A) social media platforms like Quora is exceptionally significant to ensure that the quality and the quantity of content are presented to users, based on the intent of the question and thus enriching overall user experience. Detecting Duplicate questions is a challenging problem because natural language is very expressive, and a unique intent can be conveyed using different words, phrases, and sentence structuring. Machine learning and deep learning methods are known to have accomplished superior results over traditional natural language processing techniques in identifying similar texts.

In this report, taking Quora for our case study, we explored and applied different machine learning and deep learning techniques on the task of identifying duplicate questions on Quora's question pair dataset. Machine learning models are trained with features that can help solve this problem. That is why Quora decided to publish the challenge of classifying duplicate and non-duplicate questions using natural language processing and machine learning on Kaggle website where they can get many good solutions for their problem. The task is a binary classification problem where we need to classify duplicate and non-duplicate questions. To solve this problem various machine learning models are trained with features that can help solve this problem.

Keywords : Neural Network, CNN, RNN, BiLSTM, Siamese Neural Network, keras, TensorFlow

Contents

List of Figures

List of Tables

Chapter 1

Profile : Indian Institute of Information Technology Dharwad

The Indian Institute of Information Technology Dharwad is one of the 20 IIITs proposed under non-profit, Public-Private Partnership model set up by the Ministry of Human Resource Development (MHRD), Government of India. It is an academic and research institute funded by the Government of India, Government of Karnataka and industrial partner Keonics.
Recently, the institute has been declared as an Institute of National Importance under the Indian Institutes of Information Technology Public-Private Partnership Act of Parliament (No. 23 of 2017).

Figure 1.1: IIIT Dharwad

1.1 Faculty

Faculty at IIIT Dharwad are highly qualified with PhDs and Postdocs from institutes of repute in India and abroad and have the right blend of teaching, research, and industrial experience. The Director has ample experience in both academia and industry in India as well as USA. With energy and intent, they are working to set high standards in both teaching/learning and R&D.

1.2 Innovation

IIIT Dharwad aims to innovate in all its aspects: teaching, curriculum, campus design and R&D. The idea is to ensure that there is nothing boring or ordinary about IIIT Dharwad!

Teaching-Learning : Being a technical institute of national importance, a special emphasis is given to practice-based teaching and learning process.

Curriculum Design : The curriculum at IIIT Dharwad is evolving and dynamic to ensure that the courses taught are current and relevant.

Campus Design : The Master Plan for IIIT Dharwad ensures that the Institute will have a unique, modern, green, and smart IT campus. About 70% of the area will be left for greenery even after all the buildings are constructed at the end of four phases. Needless to say that the campus will feature Wi-Fi, outdoor sports facilities, fully equipped academic and research laboratories, and modern IT-enabled classrooms.

R&D : The hallmark of a good institute is its strength in research. Aptly IIIT Dharwad is being nurtured as a research institute to have close collaborations with IT industry and other R&D institutes.

1.3 Campus and Location

The permanent campus of IIIT Dharwad is coming up in a 60-acre plot between the two cities of Hubballi and Dharwad. It is close to both cities and also the Hubballi Airport (UBX) yet away from all the traffic, noise and pollution. The location is indeed pristine, right where the hills start on the way to Western Ghats. It is also the catchment area of river Shalmala that flows down to join Arabian Sea near Karwar. The transit campus is located at IT Park in Hubballi near the famous Rani Chennamma circle.

Dharwad is a well-known center for higher learning including music and arts. Hubballi-Dharwad are having a significant concentration of Higher Educational institutions and universities of repute including Indian Institute of Technology-Dharwad (IIT-Dharwad), Karnataka University, University of Agricultural Sciences Dharwad (UAS-Dharwad), Karnataka State Law University, KLE Technological University, Karnataka Institute of Medical Sciences and Hospital (KIMS-Hubballi), SDM College of Medical Sciences and Hospital, and SDM College of Engineering and Technology.

Hubballi-Dharwad are twin cities, now being developed as smart cities in the state of Karnataka and together form the 2nd largest urban city next to Bangalore. The cities are located 410 kms from Bangalore city with excellent connectivity by air, rail and road to Bangalore, thereby enabling IIIT Dharwad to reap rich dividends from the strengths of Bangalore as the nation's IT hub.

There are several significant historical places located nearby viz., Badami, Aihole, Pattadkal, Hampi, Kudalasangama, and Vijayapura. Also, the cities are very close to

Western Ghats, declared as UNESCO World Heritage Site and are one of the eight "hottest hot-spots" of rich biological diversity, flora, and fauna in the world. Dharwad city is 160 kms away from Panjim city, Goa which is a popular tourist destination.

Chapter 2

Introduction

Social media platforms are a great success as can be witnessed by the number of the active user base. In the age of internet and social media, there has been a plethora of social media platforms, for example, we have Facebook, for user interaction, LinkedIn, for professional networking, WhatsApp for chat and video calling, Stack Overflow for technical queries, Instagram for photo sharing. Along the line, Quora is a Question & Answer platform and builds around a community of users to share knowledge and express their, opinion and expertise on a variety of topics. Question Answering sites like Yahoo and Google Answers existed over a decade however they failed to keep up the content value of their topics and answers due to a lot of junk information posted and thus their user base declined.

On the other hand, Quora is an emerging site for the quality content, launched in 2009 and as of 2019, it is estimated to have 300 million active users . Quora has 400,000 unique topics and domain experts as its user so that the users get the first-hand information from the experts in the field. Quora is a platform where any user can post any question or query on their website so that people who have answers to that questions can share their unique insights and experiences regarding the query.

Quora serves as a question-answer platform and provides value to its users by allowing them to discuss any topic. Users can collaborate to give suggestions on edits to the answers that may not give complete solution to the question.

The community experience users get on Quora is very rich. That is why Quora is very popular platform among many internet users who visit the website looking for answers to their questions.

About 100 million people visit Quora every month. Because the user base is high and from many different countries, people asks similar questions without knowing that the question they are asking has already been answered or is very similar to the question that is already been asked by other users. This creates a problem of duplicate questions for both users and the website.

For users the problem arises because there are many similar questions on the platform which confuses users on which answer to check to get correct answers.
And for the website it is problematic to manage similar questions and their answers that takes extra space on their servers and creates confusion among the users which

makes them inefficient.

Detection of duplicate or similar questions becomes very necessary to solve this problem. Detection and classification of similar question pairs is one of the most difficult task because there are different words to describe the same meaning in normal human language which makes this problem very difficult to solve.

To solve this task, Quora has published the dataset on Kaggle as a challenge. With the growing repository of the knowledge base, there is a need for Quora to preserve the trust of the users, maintain the content quality, by discarding the junk, duplicate and insincere information. Quora has successfully overcome this challenge by organizing the data effectively by using modern data science approach to eliminate question duplication.

Effectively detecting duplicate questions not only saves time for seekers to find the best answer to their questions, but also reduces the effort of writers in terms of answering multiple versions of the same question.

Chapter 3

Related Works

The task of identifying duplicated questions can be viewed as an instance of the paraphrase identification problem, which is a well-studied NLP task that uses natural language sentence matching (NLSM) to determine whether two sentences are paraphrase or not.[1]

With the renaissance of neural networks,[1] two types of neural-based frameworks have been proposed for the task of paraphrase identification. The first framework is based on a "siamese" neural network consisting of two sub-networks joined at their outputs, where the sub-networks share the same weights at all levels and are responsible for extracting features from the input, and the output level computes the distance (cosine of the angle) between the two feature vectors generated by the sub-networks.[2]

Although the siamese structure is lightweight and easy to train given that the sub-networks share parameters, there is no interaction between the two sentences during the training process, which might cause information loss.[1]

To cope with the limitations of the siamese framework, a second framework named "compare-aggregate" is proposed,[3] which captures the interaction between two sentences by performing a word-level matching and aggregating the matching results into a vector for the final classification.

However, the "compare-aggregate" model fails to account for other types of granular matchings (e.g. phrase-by-sentence) and only performs matching in a single direction, which also neglects information in the sentence pairs.[1]

To tackle the limitations of the neural-based frameworks, Wang et al. proposed a bilateral multiperspective matching model (BiMPM) [4] which also applies the "compare-aggregate" framework, but instead matches the two sentences bidirectionally.

Specifically, the model encodes the input sentences using a Bidirectional Long-Short Term Memory Network (BiLSTM), matches the encodings in two directions as well as multiple perspectives in each direction, and aggregates the matching results into a fixed-length vector using another BiLSTM layer .[1]

Tomar et al. later used character-based n-gram embeddings instead of word embeddings to improve the performance of BiMPM . By matching the sentences from multiple

perspectives, the model captures more interactive features between the sentence pairs and achieves state-of-the-art performance in paraphrase identification .

Aside from the "siamese" and "compare-aggregate" frameworks, another effective approach to the paraphrase identification problem is by using the "attention-based" framework, which models the interdependence between the two sentences in a sentence pair.[5]

One of the best-performing attention-based models is the attention-based convolutional neural network (ABCNN), which achieves attention at multiple levels of granularity (word-level, phrase-level and sentence-level) by stacking multiple convolutional layers, thus capturing the information in the sentences at different levels of abstraction .

We have extracted different features from the existing question pair dataset and applied various machine learning techniques. After employing feature engineering upon raw dataset, we experimented with different machine learning algorithms to draw our baseline.

We also showed that not all features were useful in predicting duplicate question and after analyzing and dropping a few of the features, our result for ML models slightly improved but did not degrade at all. We also have the existing baseline from the works of literature, which we have surpassed.

We then tried many deep learning methods like CNN, Bi-LSTM & BiLSTM with third feature of common words between question pair to finally experiment with our three best deep learning architectures.

With our experiment results, we have shown that deep learning methods are suitable for solving the problem of detecting semantically similar questions. Our deep learning neural networks performed better than baselines from previous research studies.

Chapter 4

Machine Learning

4.1 What is Machine Learning?

According to SAS, "Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention."

Figure 4.1: Machine Learning

Even though the term machine learning has been under the spotlight only recently, the concept of machine learning has existed since a long time, the earliest example of it being Alan Turing's Enigma machine that he developed during World War II. Today, machine learning is almost everywhere around us, right from the ordinary things in our lives to the more complicated calculations involving Big Data. For instance, Tesla self-driving car and the personalized recommendations on sites such as Netflix, Amazon, and Spotify, are all outcomes of Machine Learning.[6]

4.1.1 How Does Machine Learning Work?

Machine Learning is, undoubtedly, one of the most exciting subsets of Artificial Intelligence. It completes the task of learning from data with specific inputs to the machine. It's important to understand what makes Machine Learning work and, thus, how it

can be used in the future.

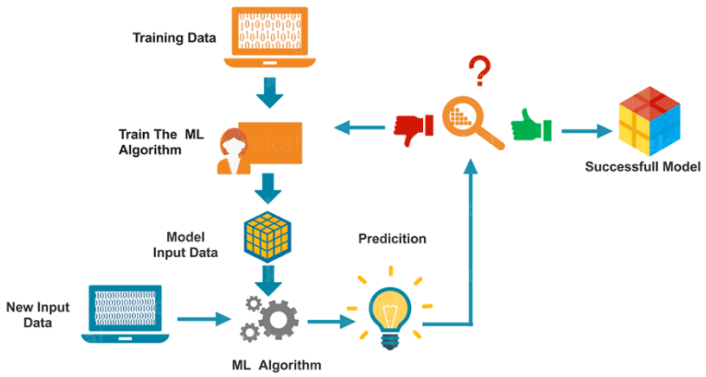


Figure 4.2: Simple Machine Learning working

The Machine Learning process starts with inputting training data into the selected algorithm. Training data being known or unknown data to develop the final Machine Learning algorithm. The type of training data input does impact the algorithm, and that concept will be covered further momentarily.

To test whether this algorithm works correctly, new input data is fed into the Machine Learning algorithm. The prediction and results are then checked.

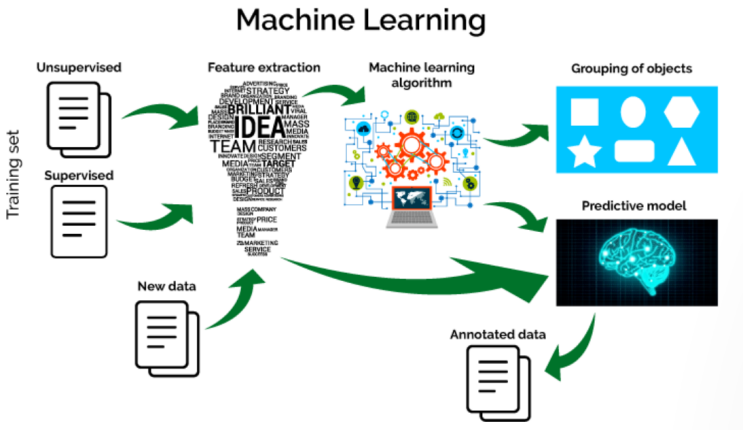


Figure 4.3: Another Simple Machine Learning working

If the prediction is not as expected, the algorithm is re-trained multiple numbers of times until the desired output is found. This enables the Machine Learning algorithm to continually learn on its own and produce the most optimal answer that will gradually increase in accuracy over time.[7]

4.2 History of machine learning

Today, machine learning algorithms enable computers to communicate with humans, autonomously drive cars, write and publish sport match reports, and find terrorist

suspects. I firmly believe machine learning will severely impact most industries and the jobs within them, which is why every manager should have at least some grasp of what machine learning is and how it is evolving.

- **18th century — Development of statistical methods :** Several vital concepts in machine learning derive from probability theory and statistics, and they root back to the 18th century. In 1763, English statistician Thomas Bayes set out a mathematical theorem for probability, which came to be known as Bayes Theorem that remains a central concept in some modern approaches to machine learning.

- **1950 — The Turing Test :** English mathematician Alan Turing's papers in the 1940s were full of ideas on machine intelligence. "Can machines think?" He asked. In 1950, he suggested a test for machine intelligence, later known as the Turing Test, in which a machine is called "intelligent" if its responses to questions could convince a human.

- **1952 — Game of Checkers :** In 1952, researcher Arthur Samuel created an early learning machine, capable of learning to play checkers. It used annotated guides by human experts and played against itself to learn to distinguish right moves from bad.

- **1956 — The Dartmouth Workshop :** The term 'artificial intelligence' was born during the Dartmouth Workshop in 1956, which is widely considered to be the founding event of artificial intelligence as a field. The workshop lasted six to eight weeks and was attended by mathematicians and scientists, including computer scientist John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon.

- **1957 — The Perceptron :** Noted American psychologist Frank Rosenblatt's Perceptron was an early attempt to create a neural network with the use of a rotary resistor (potentiometer) driven by an electric motor. The machine could take an input (such as pixels of images) and create an output (such as labels).

- **1967 — Nearest neighbor algorithm:** The Nearest Neighbor (NN) rule is a classic in pattern recognition, which appeared in several research papers in the 1960s, especially in an article written by T. Cover and P. Hart in 1967. The algorithm mapped a route for traveling salespeople, starting at a random city but ensuring they visit all cities during a short tour.

- **1973 — The Lighthill report and the AI winter :** The UK Science Research Council published the Lighthill report by James Lighthill in 1973, presenting a very pessimistic forecast in the development of core aspects in AI research. It stated that "In no part of the field have the discoveries made so far produced the major impact that was then promised." As a result, the British government cut the funding for AI research in all but two universities. This period of reduced

funding and interest is known as an AI winter.

- **1979 — Stanford Cart :** The students at Stanford University invented a robot called the Cart, radio-linked to a large mainframe computer, which can navigate obstacles in a room on its own. Though the entire room crossing took five hours due to barely adequate maps and blunders, the invention was state of the art at the time.

- **1981 — Explanation Based Learning (EBL) :** Gerald Dejong introduced the concept of Explanation Based Learning (EBL), which analyses data and creates a general rule it can follow by discarding unimportant data.

- **1985 — NetTalk :** Francis Crick Professor Terry Sejnowski invented NetTalk, NETtalk, a program that learns to pronounce written English text by being shown text as input and matching phonetic transcriptions for comparison. The intent was to construct simplified models that might shed light on human learning.

- **1986 — Parallel Distributed Processing and neural network models :** David Rumelhart and James McClelland published Parallel Distributed Processing, which advanced the use of neural network models for machine learning.

- **1992 — Playing backgammon :** Researcher Gerald Tesauro created a program based on an artificial neural network, which was capable of playing backgammon with abilities that matched top human players.

- **1997 — Deep Blue :** IBM's Deep Blue became the first computer chess-playing system to beat a reigning world chess champion. Deep Blue used the computing power in the 1990s to perform large-scale searches of potential moves and select the best move.

- **2006 — Deep Learning :** Geoffrey Hinton created the term "deep learning" to explain new algorithms that help computers distinguish objects and text in images and videos.

- **2010 — Kinect :** Microsoft developed the motion-sensing input device named Kinect that can track 20 human characteristics at a rate of 30 times per second. It allowed people to interact with the computer through movements and gestures.

- **2011 — Watson and Google Brain :** IBM's Watson won a game of the US quiz show Jeopardy against two of its champions. In the same year, Google Brain was developed its deep neural network which could discover and categorize objects in the way a cat does.

- **2012 — ImageNet Classification and computer vision :** The year saw the publication of an influential research paper by Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever, describing a model that can dramatically reduce the error rate in image recognition systems. Meanwhile, Google's X Lab developed a machine learning algorithm capable of autonomously browsing YouTube videos to identify the videos that contain cats.

- **2014 — DeepFace:** Facebook developed a software algorithm DeepFace, which can recognize and verify individuals on photos with an accuracy of a human.

- **2015 — Amazon Machine Learning :** AWS's Andy Jassy launched their Machine Learning managed services that analyze users' historical data to look for patterns and deploy predictive models. In the same year, Microsoft created the Distributed Machine Learning Toolkit, which enables the efficient distribution of machine learning problems across multiple computers.

- **2016 — AlphaGo :** AlphaGo, created by researchers at Google DeepMind to play the ancient Chinese game of Go, won four out of five matches against Lee Sedol, who has been the world's top Go player for over a decade.

- **2017 — Libratus and Deepstack :** Researchers at Carnegie Mellon University created a system named Libratus, and it defeated four top players at No Limit Texas Hold 'em, after 20 days of play in 2017. Researchers at the University of Alberta also reported similar success with their system, Deepstack.

4.3 Types of Machine Learning

4.3.1 Supervised Learning

Supervised learning deals in clearly defined and outlined inputs and outputs and the algorithms here are trained through labelled tags. In supervised learning, the learning algorithm receives both the defined set of inputs along with the correct set of outputs. So, the algorithm would then modify the structure according to the pattern it perceives in the inputs and outputs received. This is a pattern recognition model of learning that involving methods such as classification, regression, prediction, and gradient boosting.[6]

Supervised learning is usually applied in cases involving historical data. For instance, using the historical data of credit card transactions, supervised learning can predict the future possibilities of faulty or fraudulent card transactions.

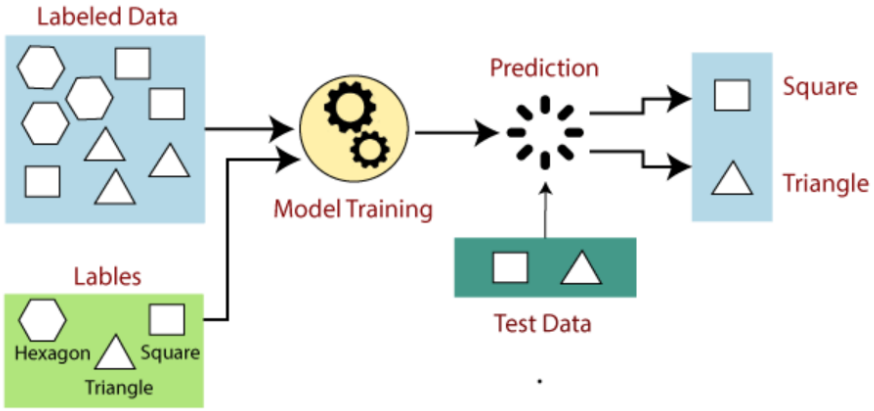


Figure 4.4: Supervised learning

The top algorithms currently being used for supervised learning are:

- Polynomial regression
- Random forest
- Linear regression
- Logistic regression
- Decision trees
- K-nearest neighbors
- Naive Bayes

4.3.2 Unsupervised Learning

Contrary to supervised learning that uses historical data sets, unsupervised learning is apps that lack any historical data whatsoever. In this method, the learning algorithm goes beyond data to come up with the apt structure – although the data is devoid of tags, the algorithm splits the data into smaller chunks according to their respective characteristics, most commonly with the aid of a decision tree. Unsupervised learning is ideal for transactional data applications, such as identifying customer segments and clusters with specific attributes.[6]

Unsupervised learning algorithms are mostly used in creating personalized content for individual user groups. Online recommendations on shopping platforms and identification of data outliers are two great examples of unsupervised learning.

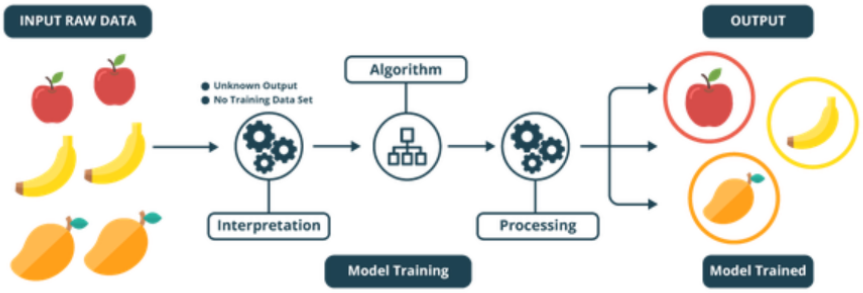


Figure 4.5: Unsupervised learning

The top algorithms currently being used for unsupervised learning are:

- Polynomial regression
- Partial least squares
- Fuzzy means
- Singular value decomposition
- K-means clustering
- Apriori
- Hierarchical clustering
- Principal component analysis

4.3.3 Reinforcement Learning

Reinforcement learning is quite similar to traditional data analysis method where the algorithms learn through trial and error method, after which it declares the outcomes with the best possible results. Reinforcement learning is comprised of three fundamental components – agent, environment, and actions. The agent here refers to the learner/decision-maker; the environment consists of all that which the agent interacts with, and the actions refer to the things that the agent can perform.

This type of learning helps improve the algorithm over time because it continues to adjust the algorithm as and when it detects errors in it. Google Maps routes are one of the most excellent examples of reinforcement learning.[6]

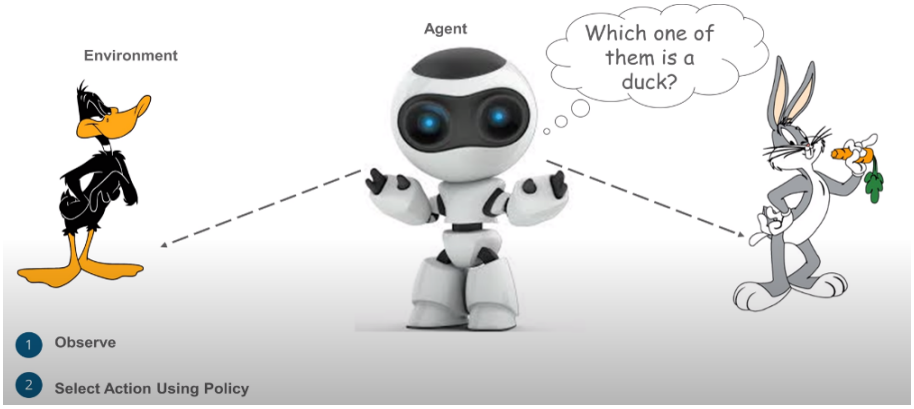


Figure 4.6: Reinforcement Learning

Now that you're aware of what is Machine Learning, including the types in which you can make the machines learn, let's now look at the various applications of Machine Learning in the world today.

4.4 Why Is Machine Learning Important In Today's World?

The main focus of machine learning is to help organizations enhance their overall functioning, productivity, and decision-making process by delving into the vast amounts of data reserves. As machines begin to learn through algorithms, it will help businesses to unravel such patterns within the data that can help them make better decisions without the need for human intervention.

4.5 Advantages of machine Learning

1. Timely Analysis And Assessment

By sifting through massive amounts of data such as customer feedback and interaction, ML algorithms can help you conduct timely analysis and assessment of your organizational strategies. When you create a business model by browsing through multiple sources of data, you get a chance to see the relevant variables. In this way, machine learning can help you to understand the customer behaviour, thereby allowing you to streamline your customer acquisition and digital marketing strategies accordingly.

2. Real-time Predictions Made Possible Through Fast Processing

One of the most impressive features of ML algorithms is that they are super fast, as a result of which data processing from multiple sources takes place rapidly. This, in turn, helps in making real-time predictions that can be very beneficial for businesses. For instance,

- **Churn analysis** –

It involves identifying those customer segments that are likely to leave your brand.

• **Customer leads and conversion** –
ML algorithms provide insights into the buying and spending patterns of various customer segments, thereby allowing businesses to devise strategies that can minimize losses and fortify profits.

• **Customer retention** –
ML algorithms can help identify the backlogs in your customer acquisition policies and marketing campaigns. With such insights, you can adjust your business strategies and improve the overall customer experience to retain your customer base.

3. Transforming Industries

Machine learning has already started to transform industries with its ability to provide valuable insights in real-time. Finance and insurance companies are leveraging ML technologies to identify meaningful patterns within large data sets, to prevent fraud, and to provide customized financial plans for various customer segments. In healthcare, wearables and fitness sensors powered by ML technology are allowing individuals to take charge of their health, consequently minimizing the pressure on health professionals. Machine learning is also being used by the oil and gas industry to find out new energy sources, analyzing the minerals in the ground, predict system failures, and so on.

Wide Range of Applications

ML has a wide variety of applications. This means that we can apply ML on any of the major fields. ML has its role everywhere from medical, business, banking to science and tech. This helps to create more opportunities.

It plays a major role in customer interactions. Machine Learning can help in the detection of diseases more quickly. It is helping to lift up businesses. That is why investing in ML technology is worth it.

4.6 Disadvantages of Machine Learning

1. **Possibility of High Error**

   In ML, we can choose the algorithms based on accurate results. For that, we have to run the results on every algorithm. The main problem occurs in the training and testing of data.

   The data is huge, so sometimes removing errors becomes nearly impossible. These errors can cause a headache to users. Since the data is huge, the errors take a lot of time to resolve.[8]

2. **Algorithm Selection**

   The selection of an algorithm in Machine Learning is still a manual job. We have to run and test our data in all the algorithms. After that only we can decide what

algorithm we want. We choose them on the basis of result accuracy. The process is very much time-consuming.[8]

3. **Data Acquisition**

   In ML, we constantly work on data. We take a huge amount of data for training and testing. This process can sometimes cause data inconsistency. The reason is some data constantly keep on updating.

   So, we have to wait for the new data to arrive. If not, the old and new data might give different results. That is not a good sign for an algorithm.[8]

4. **Time and Space**

   Many ML algorithms might take more time than you think. Even if it's the best algorithm it might sometimes surprise you. If your data is large and advanced, the system will take time. This may sometimes cause the consumption of more CPU power.

   Even with GPUs alongside, it sometimes becomes hectic. Also, the data might use more than the allotted space.[8]

4.7 Machine Learning Prerequisites

For those interested in learning beyond what is Machine Learning, a few requirements should be met to be successful in pursual of this field. These requirements include: [7]

1. Basic knowledge of programming and scripting languages.
2. Intermediate knowledge of statistics and probability.
3. Basic knowledge of linear algebra. In the linear regression model, a line is drawn through all the data points, and that line is used to compute new values.
4. Understanding of calculus.
5. Knowledge of how to clean and structure raw data to the desired format to reduce the time taken for decision making.

Each of these prerequisites will help you quickly succeed in transitioning into Machine Learning.

4.8 Top Applications of Machine Learning in real World

4.8.1 Image Recognition

It is one of the most common machine learning applications. There are many situations where you can classify the object as a digital image. For digital images, the measurements describe the outputs of each pixel in the image.

In the case of a black and white image, the intensity of each pixel serves as one measurement. So if a black and white image has N*N pixels, the total number of pixels and hence measurement is N2.

In the coloured image, each pixel considered as providing 3 measurements of the intensities of 3 main colour components ie RGB. So N*N coloured image there are 3 N2 measurements.[9]

- **For face detection** –
  The categories might be face versus no face present. There might be a separate category for each person in a database of several individuals.
- **For character recognition** –
  We can segment a piece of writing into smaller images, each containing a single character. The categories might consist of the 26 letters of the English alphabet, the 10 digits, and some special characters.

Figure 4.7: Image Recognition

4.8.2 Traffic Alerts

It is a combination of multiple factors like how many people are currently using the services of Google Maps, historic data of that route, and some real-time techniques. While you use Google maps, you are allowing the app to use data like:

- Your location

- Your average traveling speed
- Answers to the questions like 'does the route still have traffic'?
- Day, time, and any specific occasion

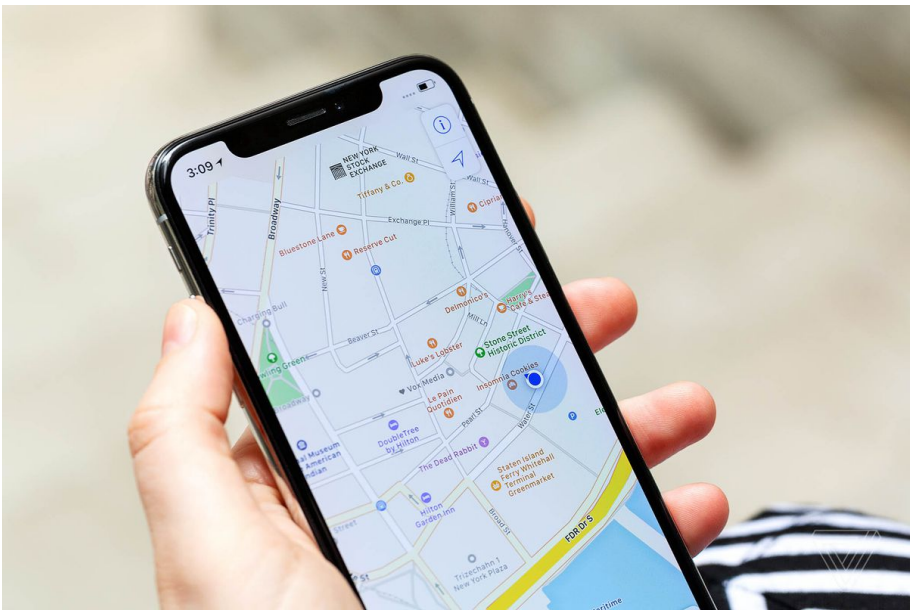


Figure 4.8: Google Maps

All such data is captured and stored by the application. By using this data, AI & machine learning algorithms make the right conclusions and give you the exact information.[13]

The updated feature of Google Maps also helps us to know how far is the upcoming bus from a specific stop and even make predictions on the bus delays. Further, the machines are so intelligent that they can even tell you how crowded the bus or train is so that you can make a call to board the bus!

4.8.3 Video Surveillance

This is one of the most advanced applications of machine learning and AI. Videos give a better opportunity to fetch valuable information from automated surveillance devices compared to any other source. This is only possible because machines keep a better outlook for the objects compared to human minds.

Figure 4.9: Video Surveillance

Video surveillance is used for different purposes like:

- Copper theft prevention
- Abnormal event detection
- Facility protections
- Operation monitoring
- Parking lots
- Traffic monitoring
- Shopping patterns

Surveillance footages are the best machine learning datasets because of their accuracy but these footages are hard to obtain. Thus, training the object detector comes into existence so that these objects can easily recognize the targets from normal images.[10]

4.8.4 Sentiment Analysis

Sentiment analysis is a top-notch machine learning application that refers to sentiment classification, opinion mining, and analyzing emotions. Using this model, machines groom themselves to analyze sentiments based on the words. They can identify if the words are said in a positive, negative, or neutral notion Also, they can define the magnitude of these words.

With the help of the process called Natural Language Processing (NLP), data-miners automatically extract and conclude the opinion by analyzing both types of machine learning algorithms – supervised and unsupervised data. Companies that are dealing with customers use this model to improve customer experience based on the

feedback.

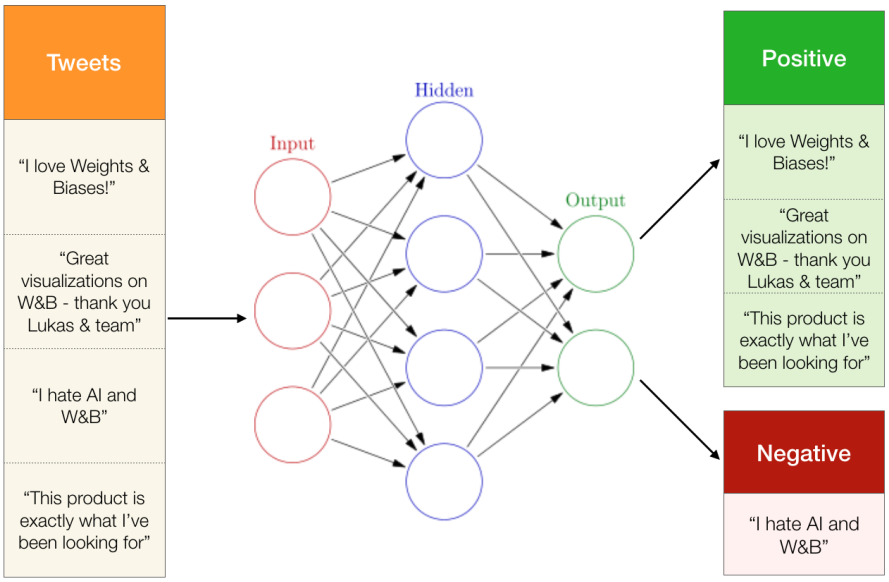


Figure 4.10: Sentiment analysis

In the below example Machine Learning model interprets the clients' tweets and bifurcates it into positive and negative notions. This gives the brand a quick view of what their clients are saying about them and accordingly they can make efforts to resolve any negative feedback.[10]

Another machine learning example is – Music applications. Apps like Ganna.com, Jiosaavn also suggest music based on user sentiments by analyzing the history of songs played, favorite playlists, and even time of listings music.

Benefits of Adopting Sentiment Analysis

- Effective brand and social media monitoring
- Enhanced customer support
- Competitive analysis
- Better tracking of employee feedbacks and UGC (user-generated content like reviews)

4.8.5 Product Recommendation

While shopping on eCommerce brands like Amazon and Flipkart, you get to see recommended items or options like 'users who bought this product also bought'; 'users also buy this along with this product'!

All these are the outcomes of advanced machine learning training wherein the system learns individual patterns of users and suggests new or additional products to buy.

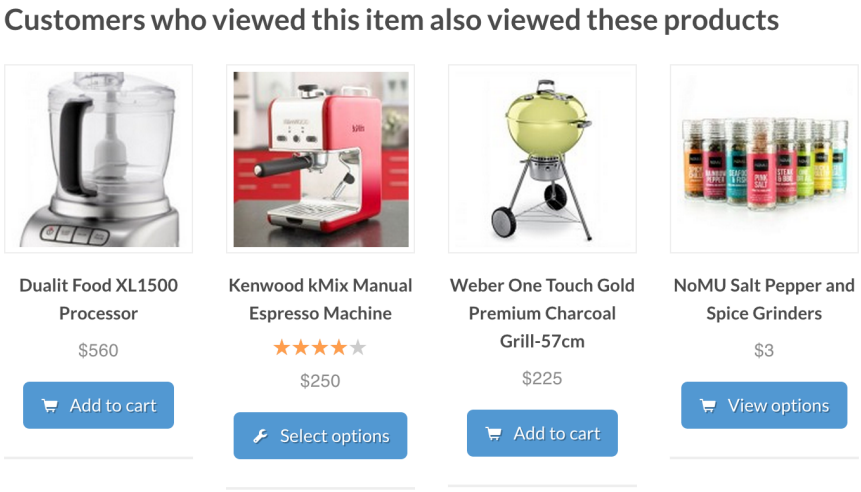


Figure 4.11: Product Recommendation

Not only on ecommerce apps, but even you will also get to see similar product suggestions on Google, YouTube, and freemium apps. Yes, this is a real-world application of machine learning.[10]

4.8.6 Google Translate

Traveling to a new place is always thrilling but the only enigma is to understand the common language of that place. To solve this dilemma, Google has launched an app that can help in easy translation of any language.

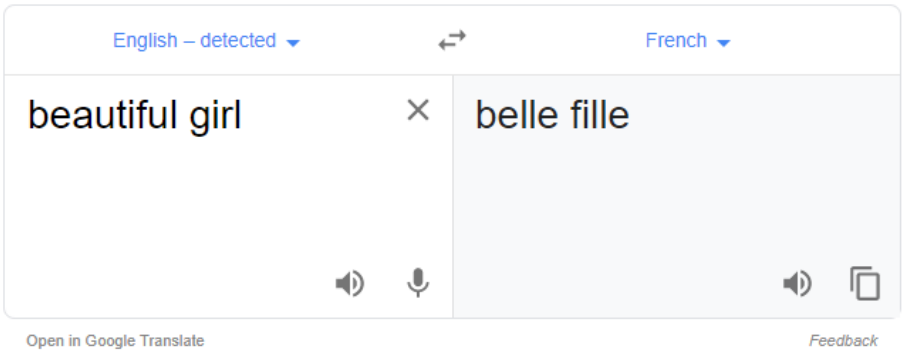


Figure 4.12: Google Translate

Google uses 'Google Neural Machine Translation' that has the ability to absorb thousands of languages, words, and dictionaries and transmute any sentence in the desired language.[10]

4.8.7 Virtual Professional Assistants

Machine learning-based VPA is among the most popular examples of machine learning applications. With a surge in smart devices usage, machines are becoming smarter in adopting human behaviors.[10]

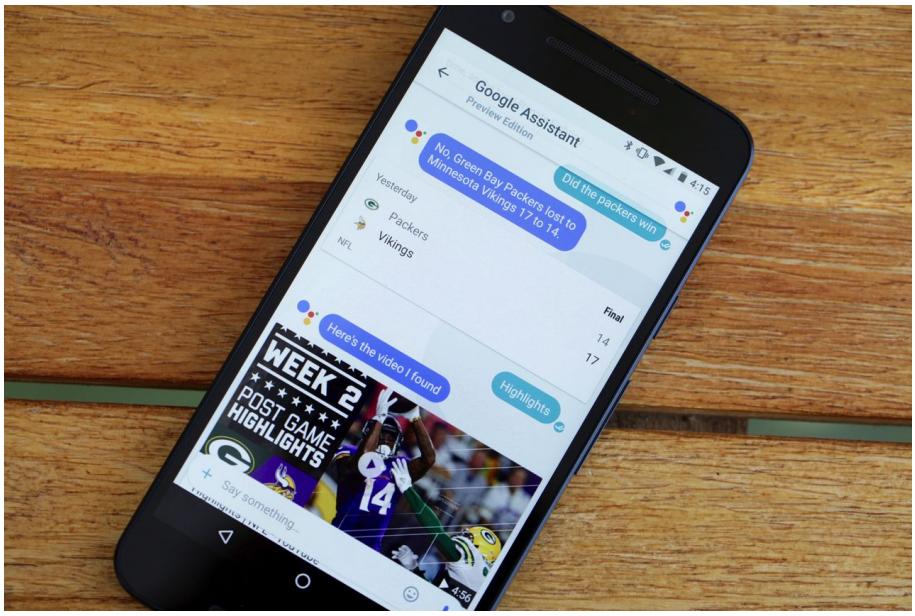


Figure 4.13: Virtual Professional Assistants

These are all examples of machine learning. From turning on smart appliances to booking an Uber on command, it is all revolving around machine learning algorithms.

4.8.8 Auto-Driven Cars

We all have heard that self-driving cars are the future of the automobile industry. Yes, the concept of self-driving is also implied based on machine learning, deep learning, and AI.

Some of the common machine learning algorithms used in autonomous driving are:[10]

- Scale-invariant feature transform (SIFT)
- AdaBoost
- TextonBoost
- You only look once (YOLO)

Figure 4.14: Auto-Driven Cars

Chapter 5

Tools and Technologies

5.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Figure 5.1: Python

5.1.1 Why Python?

- **HighLevel**
  Python derives components from the **natural language** that we humans use to communicate with each other. This makes it easier for anyone to try and relate what exactly could be happening without the burden of going through tons of **machine code**.

- **Interpreted**
  Python codes are compiled **line-by-line** which makes debugging errors much easier and efficient. But this comes at a cost as it is much slower than other programming languages.

- **Easy syntax**
  Python makes use of **indentations** instead of braces to distinguish what blocks of code come under which class or function. This makes the code look well **distributed** and makes it easy for anyone to read it.

- **Dynamic Semantics**
  If you are an old school coder, you would know that before using anything, you would need to **initialize** it. It does all of this **dynamically**.

5.1.2 What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

You can have a really good career with Python as your go-to language for solving problems. Let me name a few of the titles that you can pursue.

1. **Data Scientist**
   A Data Scientist is someone who cracks **complex** problems which relate to the field of math, statistics and brings around a solution to these problems in a logical manner.

2. **Software Engineer**
   Software engineers design, develop, test and maintain software applications that they create for their clients according to the requirements.

3. **Web Developer**
   Web developers create web applications to serve their users using the **client-server** model. There are applications such as information sharing, social network platforms, entertainment which are just a few to name.

5.1.3 Python Features

Python is slow. However, its popularity does not stop to grow as it can achieve better productivity with lesser code, making Python one of the most liked languages. Python has a list of features that attract almost anyone to start coding with it.

- Simplicity

Python is its own kind of simple. All you need to know is how the indentations work and you can code the most complex of problems in the fewer lines of code.

- Open Source

Python is free for anyone to use. You even have the freedom to modify the code of Python to better your own needs without facing any repercussions.

• Portability

Code writing can be done once and run across different systems without any changes. This makes it super helpful when a team works on a project.

• Embedding Properties

Python allows the code of other languages such as C, C++ to be embedded, which makes it much more powerful and versatile.

• Interpretation

As you know, Python is compiled line-by-line, making debugging easier and memory management much more efficient.

• Library Support

Python supports libraries that you can use and get started off to obtain your solutions much faster and easier. And the community for these libraries is very active and helpful.

• OOPS

Object-Oriented concepts help you replicate real-world scenarios in your code and also provide security to them so that you can obtain a well-made application.

5.2 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.[11]

Figure 5.2: Anaconda

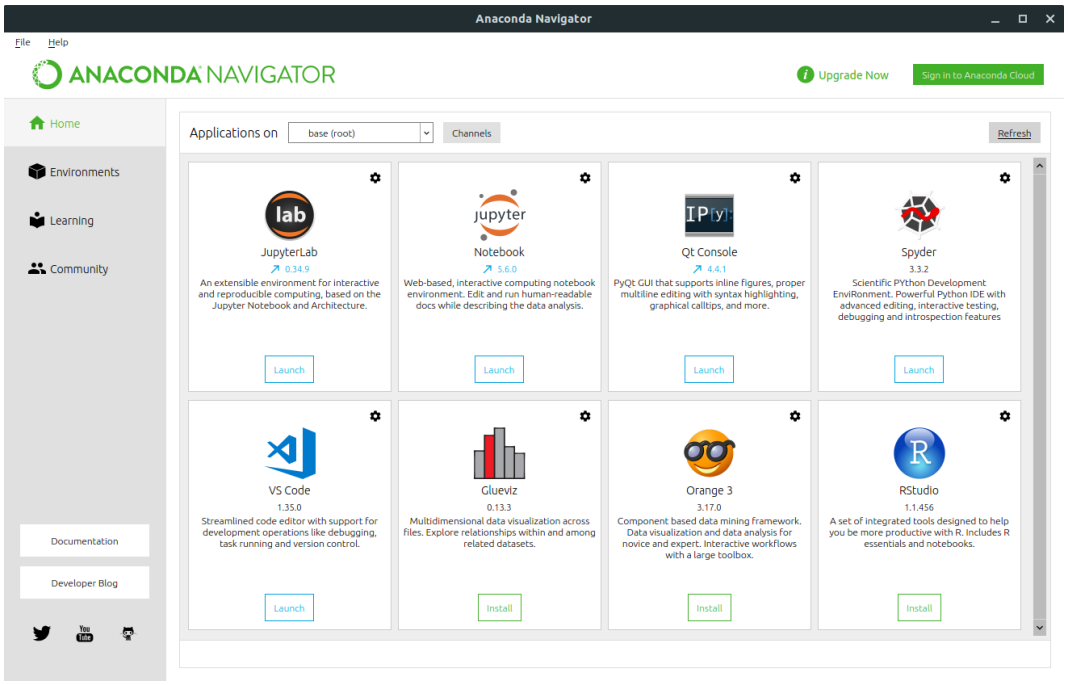


Figure 5.3: Anaconda Interface

5.2.1 Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.[11]

5.2.2 Applications of Navigator

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

5.2.3 Managing Environments

Navigator uses conda to create separate environments containing files, packages, and their dependencies that will not interact with other environments.[12]

Create a new environment named snowflakes and install a package in it:

- In Navigator, click the Environments tab, then click the Create button. The Create new environment dialog box appears.
- In the Environment name field, type a descriptive name for your environment.

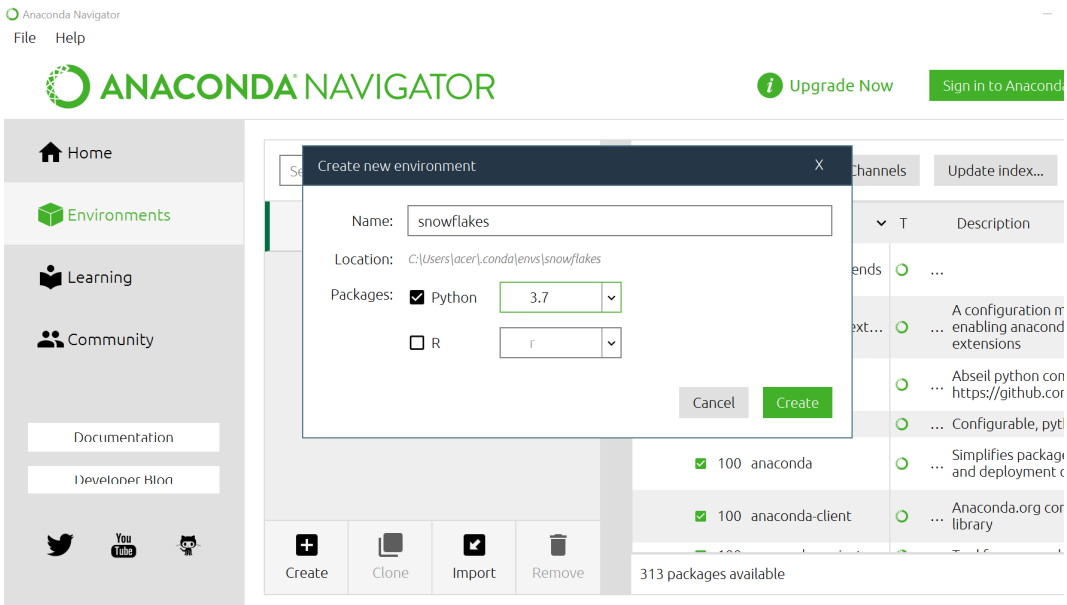


Figure 5.4: snowflakes Environment

5.2.4 Managing Python

When you create a new environment, Navigator installs the same Python version you used when you downloaded and installed Anaconda. If you want to use a different version of Python, for example Python 3.5, simply create a new environment and specify the version of Python that you want in that environment.[12]

Create a new environment named "Py 3.5" that contains Python 3.5:

1. In Navigator, click the Environments tab, then click the Create button. The Create new environment dialog box appears.
2. In the Environment name field, type the descriptive name "Py 3.5" and select the version of Python you want to use from the Python Packages box (3.8, 3.7, 3.6, 3.5, or 2.7). Select a different version of Python than is in your other environments, base or snowflakes.
3. Click the Create button.
4. Activate the version of Python you want to use by clicking the name of that environment.

5.2.5 Conda

Conda helps in Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN, and more.

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.[13]

Conda as a package manager helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because conda is also an environment manager. With just a few commands, you can set up a totally separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment.[13]

Conda can be combined with continuous integration systems such as Travis CI and AppVeyor to provide frequent, automated testing of your code.

The conda package and environment manager is included in all versions of Anaconda and Miniconda.

Conda is also included in Anaconda Enterprise, which provides on-site enterprise package and environment management for Python, R, Node.js, Java and other application stacks. Conda is also available on conda-forge, You may also get conda on PyPI, but that approach may not be as up to date.

5.3 Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.[14]

Figure 5.5: Jupyter Notebook

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.[14]

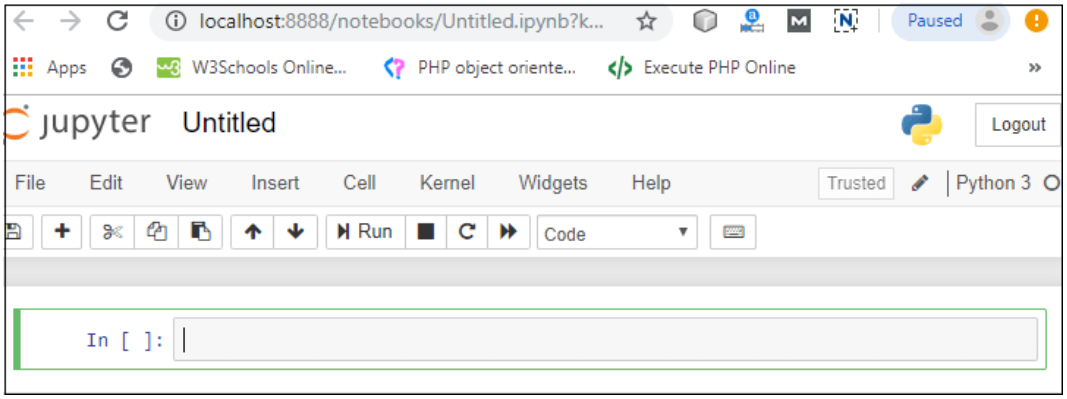


Figure 5.6: Jupyter Notebook interface

5.3.1 Components

The Jupyter Notebook combines three components:[8]

- The notebook web application :
  An interactive web application for writing and running code interactively and authoring notebook documents.

- Kernels :
  Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection.
- Notebook documents :
  Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

5.3.2 Notebook web application

The notebook web application enables users to:[15]

- Edit code in the browser, with automatic syntax highlighting, indentation, and tab completion/introspection.
- Run code from the browser, with the results of computations attached to the code which generated them.
- See the results of computations with rich media representations, such as HTML, LaTeX, PNG, SVG, PDF, etc.
- Create and use interactive JavaScript widgets, which bind interactive user interface controls and visualizations to reactive kernel side computations.
- Author narrative text using the Markdown markup language.
- Include mathematical equations using LaTeX syntax in Markdown, which are rendered in-browser by MathJax.

5.3.3 Kernels

Through Jupyter's kernel and messaging architecture, the Notebook allows code to be run in a range of different programming languages. For each notebook document that a user opens, the web application starts a kernel that runs the code for that notebook. Each kernel is capable of running code in a single programming language.

The default kernel runs Python code. The notebook provides a simple way for users to pick which of these kernels is used for a given notebook.

5.3.4 Notebook documents

Notebook documents contain the inputs and outputs of an interactive session as well as narrative text that accompanies the code but is not meant for execution. Rich output generated by running code, including HTML, images, video, and plots, is embeddeed in the notebook, which makes it a complete and self-contained record of a computation.

When you run the notebook web application on your computer, notebook documents are just files on your local filesystem with a ".ipynb" extension. This allows you to use familiar workflows for organizing your notebooks into folders and sharing them with others.[8]

Notebooks consist of a linear sequence of cells. There are three basic cell types :

- **Code cells :** Input and output of live code that is run in the kernel
- **Markdown cells :** Narrative text with embedded LaTeX equations.
- **Raw cells :** Unformatted text that is included, without modification, when notebooks are converted to different formats using nbconvert.

5.4 Colab

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs and TPUs.

Figure 5.7: colaboratory

Here's what I truly love about Colab. It does not matter which computer you have, what it's configuration is, and how ancient it might be. You can still use Google Colab! All you need is a Google account and a web browser. And here's the cherry on top – you get access to GPUs like Tesla K80 and even a TPU, for free.[16]

TPUs are much more expensive than a GPU, and you can use it for free on Colab. It's worth repeating again and again – it's an offering like no other.

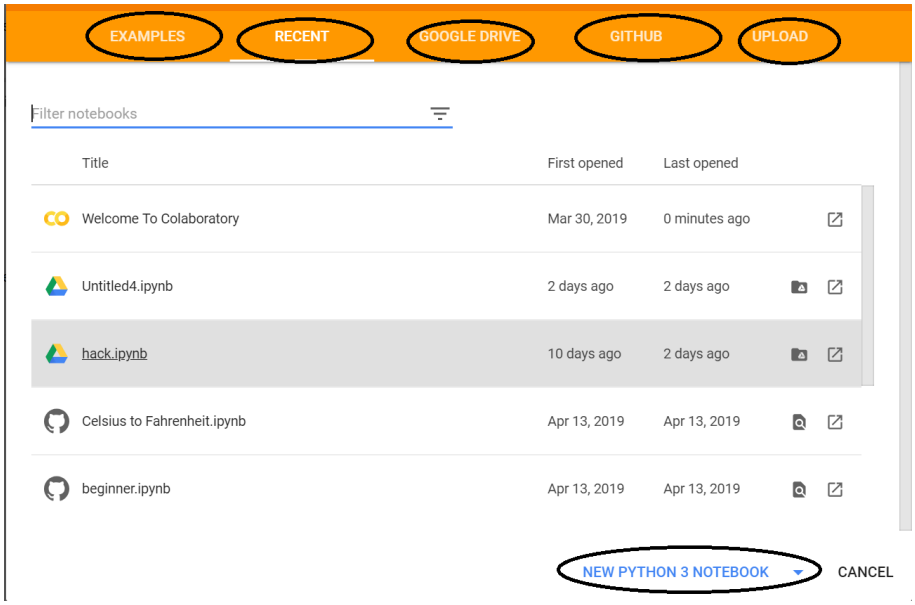


Figure 5.8: colaboratory interface interface

The availability of free GPUs and TPUs. Training models, especially deep learning ones, takes numerous hours on a CPU. We've all faced this issue on our local machines. GPUs and TPUs, on the other hand, can train these models in a matter of minutes or seconds.

Google Colab has almost every packages pre-installed, so we don't have to think about installation for the packages which will be use while coding.

I always prefer a GPU over any other CPU because of the sheer computational power and speed of execution. But, not everyone can afford a GPU because they are expensive. That's where Google Colab comes into play.

It gives you a decent GPU for free, which you can continuously run for 12 hours. For most data science folks, this is sufficient to meet their computation needs. Especially if you are a beginner, then I would highly recommend you start using Google Colab.[16]

Google Colab gives us three types of runtime for our notebooks:

- CPUs
- GPUs
- TPUs

As I mentioned, Colab gives us 12 hours of continuous execution time. After that, the whole virtual machine is cleared and we have to start again. We can run multiple CPU, GPU, and TPU instances simultaneously, but our resources are shared between these instances.

CPU	GPU	TPU
Intel Xeon Processor with two cores @ 2.30 GHz and 13GB RAM	Up to Tesla K80 with 12 GB of GDDR5 VRAM Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM	Cloud TPU with 180 teraflops of computation Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM

Table 5.1: Comparison of CPU, GPU, TPU

5.4.1 What Colab Offers You?

As a programmer, you can perform the following using Google Colab : [17]

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

5.4.2 Conclusion on Colab

Google Colab is a powerful platform for learning and quickly developing machine learning models in Python. It is based on Jupyter notebook and supports collaborative development. The team members can share and concurrently edit the notebooks, even remotely. The notebooks can also be published on GitHub and shared with the general public. Colab supports many popular ML libraries such as PyTorch, TensorFlow, Keras and OpenCV. The restriction as of today is that it does not support R or Scala yet. There is also a limitation to sessions and size. Considering the benefits, these are small sacrifices one needs to make.[18]

Chapter 6

Packages

6.1 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). It was developed by François Chollet, a Google engineer.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks.It supports other common utility layers like dropout, batch normalization, and pooling.

6.1.1 keras.preprocessing.sequence ()

: Utilities for preprocessing sequence data.

Sequence are a safer way to do multiprocessing. This structure guarantees that the network will only train once on each sample per epoch which is not the case with generators.

6.1.2 keras.preprocessing.sequence.pad_sequences ()

: Pads sequences to the same length.

6.1.3 keras.preprocessing.text.sequence ()

: Utilities for text input preprocessing.

6.1.4 keras.preprocessing.text.Tokenizer ()

: This class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count.

6.1.5 keras.utils.to_categorical ()

: Converts a class vector (integers) to binary class matrix.

6.1.6 keras.utils.plot_model ()

: Converts a Keras model to dot format and save to a file.

6.1.7 keras.backend.abs ()

: Element-wise absolute value.

6.1.8 keras.callbacks ()

: utilities called at certain points during model training.

6.1.9 keras.callbacks.EarlyStopping ()

: Stops training when a monitored metric has stopped improving.

Assuming the goal of a training is to minimize the loss. With this, the metric to be monitored would be 'loss', and mode would be 'min'. A model.fit() training loop will check at end of every epoch whether the loss is no longer decreasing, considering the min_delta and patience if applicable. Once it's found no longer decreasing, model.stop_training is marked True and the training terminates.

patience : Number of epochs with no improvement after which training will be stopped.

6.1.10 keras.callbacks.ReduceLROnPlateau ()

: Reduce learning rate when a metric has stopped improving.

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

6.1.11 keras.callbacks.ModelCheckpoint ()

: Callback to save the Keras model or model weights at some frequency.

ModelCheckpoint callback is used in conjunction with training using model.fit() to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved.

A few options this callback provides include:

- Whether to only keep the model that has achieved the "best performance" so far, or whether to save the model at the end of every epoch regardless of performance.
- Definition of 'best'; which quantity to monitor and whether it should be maximized or minimized.
- The frequency it should save at. Currently, the callback supports saving at the end of every epoch, or after a fixed number of training batches.
- Whether only weights are saved, or the whole model is saved.

6.1.12 keras.optimizers.Adam ()

: Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

6.1.13 keras.Input ()

: Input() is used to instantiate a Keras tensor.

6.1.14 keras.Model ()

: Model groups layers into an object with training and inference features.

6.1.15 keras.Model ()

: Model groups layers into an object with training and inference features.

Sequential provides training and inference features on this model.

6.1.16 keras.layers.Activation ()

: Applies an activation function to an output. Activation function, such as tf.nn.relu, or string name of built-in activation function, such as "relu".

6.1.17 keras.layers.Add ()

: Layer that adds a list of inputs to the model.

6.1.18 keras.layers.BatchNormalization ()

: Normalize and scale inputs or activations.

Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

Batch normalization differs from other layers in several key aspects:

1. Adding BatchNormalization with training=True to a model causes the result of one example to depend on the contents of all other examples in a minibatch. Be careful when padding batches or masking examples, as these can change the minibatch statistics and affect other examples.

2. Updates to the weights (moving statistics) are based on the forward pass of a model rather than the result of gradient computations.

3. When performing inference using a model containing batch normalization, it is generally (though not always) desirable to use accumulated statistics rather than mini-batch statistics. This is accomplished by passing training=False when calling the model, or using model.predict.

6.1.19 keras.layers.Bidirectional ()

: Bidirectional wrapper for RNNs.

The call arguments for this layer are the same as those of the wrapped RNN layer. Beware that when passing the initial_state argument during the call of this layer, the first half in the list of elements in the initial_state list will be passed to the forward RNN call and the last half in the list of elements will be passed to the backward RNN call.

6.1.20 keras.layers.Concatenate ()

: Layer that concatenates a list of inputs.

It takes as input a list of tensors, all of the same shape except for the concatenation axis, and returns a single tensor that is the concatenation of all inputs.

6.1.21 keras.layers.Conv1D ()

: This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide an input_shape argument (tuple of integers or None, e.g. (10, 128) for sequences of 10 vectors of 128-dimensional vectors, or (None, 128) for variable-length sequences of 128-dimensional vectors.

6.1.22 keras.layers.Dense ()

: Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

6.1.23 keras.layers.Dropout ()

: The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference. When using model.fit, training will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting trainable=False for a Dropout layer. trainable does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

6.1.24 keras.layers.Embedding ()

:Turns positive integers (indexes) into dense vectors of fixed size.

6.1.25 keras.layers.Flatten ()

: Flattens the input. Does not affect the batch size.

Inherits From: Layer

6.1.26 keras.layers.LSTM ()

: Based on available runtime hardware and constraints, this layer will choose different implementations (cuDNN-based or pure-TensorFlow) to maximize the performance. If a GPU is available and all the arguments to the layer meet the requirement of the CuDNN kernel (see below for details), the layer will use a fast cuDNN implementation.

The requirements to use the cuDNN implementation are:
1. activation == tanh
2. recurrent_activation == sigmoid
3. recurrent_dropout == 0
4. unroll is False
5. use_bias is True
6. Inputs, if use masking, are strictly right-padded.
7. Eager execution is enabled in the outermost context.

6.1.27 keras.layers.MaxPool1D ()

: The input representation by taking the maximum value over the window defined by pool_size. The window is shifted by strides. The resulting output when using "valid" padding option has a shape of: output_shape = (input_shape - pool_size + 1) / strides)

The resulting output shape when using the "same" padding option is: output_shape = input_shape / strides.

6.1.28 keras.layers.Lambda ()

: The Lambda layer exists so that arbitrary TensorFlow functions can be used when constructing Sequential and Functional API models. Lambda layers are best suited for simple operations or quick experimentation.

6.1.29 Verbose ()

: Verbose mode is an option available in many computer operating systems and programming languages that provides additional details as to what the computer is doing and what drivers and software it is loading during startup or in programming it would produce detailed output for diagnostic purposes thus makes a program easier to debug.

- verbose = 0 : will show you nothing (silent).
- verbose = 1 : will show you both progress bar and one line per epoch like this.

Figure 6.1: Verbose line

Figure 6.2: Verbose epoch

- verbose = 2 : will show you one line per epoch i.e. epoch no./total no. of epochs like this.

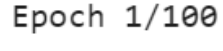


Figure 6.3: Verbose epoch

6.2 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 can run on multiple CPUs and GPUs.

TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multi-dimensional data arrays, which are referred to as tensors.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

6.3 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.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Pandas DataFrames

Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

A DataFrame is simply a 2-D array. It can be created through the pd.DataFrame constructor.

6.4 NumPy

NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object.

NumPy is a python library used for working with arrays.

Operations using NumPy

Using NumPy, a developer can perform the following operations :

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.

- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster that traditional Python lists.

6.5 NLTK (Natural Language Toolkit)

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. Tokenization, Stemming, Lemmatization, Punctuation, Character count, word count are some of these packages contains NLTK.

Natural Language Processing is manipulation or understanding text or speech by any software or machine. An analogy is that humans interact, understand each other views, and respond with the appropriate answer. In NLP, this interaction, understanding, the response is made by a computer instead of a human.

1. **spaCy** - This is completely optimized and highly accurate library widely used in deep learning.

2. **TextBlob** - This is an NLP library which works in Pyhton2 and python3. This is used for processing textual data and provide mainly all type of operation in the form of API.

3. **Gensim** - Genism is a robust open source NLP library support in python. This library is highly efficient and scalable. It uses top academic models and modern statistical machine learning to perform various complex tasks such as

    - Building document or word vectors
    - Corpora
    - Performing topic identification
    - Performing document comparison (retrieving semantically similar documents)
    - Analysing plain-text documents for semantic structure

4. **Pattern** - It is a light-weighted NLP module. This is generally used in Web-mining, crawling or such type of spidering task.

5. **Vocabulary** - This library is best to get Semantic type information from the given text.

6.6 CSV

A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.

CSV Functions

The CSV module includes all the necessary functions built in. They are:

- csv.reader
- csv.writer
- csv.register_dialect
- csv.unregister_dialect
- csv.get_dialect
- csv.list_dialects
- csv.field_size_limit

6.7 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+. SciPy makes use of Matplotlib.

Toolkits of Matplotlib

Several toolkits are available which extend Matplotlib functionality. Some are separate downloads, others ship with the Matplotlib source code but have external dependencies.

- **Basemap:** map plotting with various map projections, coastlines, and political boundaries
- **Cartopy:** a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities.
- **Excel tools:** utilities for exchanging data with Microsoft Excel
- **GTK tools:** interface to the GTK+ library
- **Qt interface**

- **Mplot3d:** 3-D plots
- **Natgrid:** interface to the natgrid library for gridding irregularly spaced data.
- **matplotlib2tikz:** export to Pgfplots for smooth integration into LaTeX documents

6.7.1 matplotlib.pyplot

Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python and the advantage of being free and open-source. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. The various plots we can utilize using Pyplot are **Line Plot, Histogram, Scatter, 3D Plot, Image, Contour, and Polar.**

6.8 Itertools

Python's Itertool is a module that provides various functions that work on iterators to produce complex iterators. This module works as a fast, memory-efficient tool that is used either by themselves or in combination to form **iterator algebra**.

For instance, let's suppose there are two lists and you want to multiply their elements. There can be several ways of achieving this. One can be using the naive approach i.e by iterating through the elements of both the list simultaneously and multiply them. And another approach can be using the map function i.e by passing the 'mul' operator as a first parameter to the map function and Lists as the second and third parameter to this function

6.9 Scikit-Learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

The functionality that scikit-learn provides include:

- Regression, including Linear and Logistic Regression
- Classification, including K-Nearest Neighbors
- Clustering, including K-Means and K-Means++
- Model selection
- Preprocessing, including Min-Max Normalization

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy:** Base n-dimensional array package
- **SciPy:** Fundamental library for scientific computing
- **Matplotlib:** Comprehensive 2D/3D plotting
- **IPython:** Enhanced interactive console
- **Sympy:** Symbolic mathematics
- **Pandas:** Data structures and analysis

Some popular groups of models provided by scikit-learn include:

- **Clustering:** for grouping unlabeled data such as KMeans.
- **Cross Validation:** for estimating the performance of supervised models on unseen data.
- **Datasets:** for test datasets and for generating datasets with specific properties for investigating model behavior.
- **Dimensionality Reduction:** for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
- **Ensemble methods:** for combining the predictions of multiple supervised models.
- **Feature extraction:** for defining attributes in image and text data.
- **Feature selection:** for identifying meaningful attributes from which to create supervised models.
- **Parameter Tuning:** for getting the most out of supervised models.
- **Manifold Learning:** For summarizing and depicting complex multi-dimensional data.
- **Supervised Models:** a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

6.9.1 sklearn.model_selection ()

: Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.

6.9.2 sklearn.model_selection.train_text_split ()

: train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually.

By default, Sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

6.9.3 sklearn.metrics.log_loss ()

: This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of a logistic model that returns y_pred probabilities for its training data y_true. The log loss is only defined for two or more labels.
For a single sample with true label yt in 0,1 and estimated probability yp that yt = 1, the log loss is

-log P(yt—yp) = -(yt log(yp) + (1 - yt) log(1 - yp))

6.9.4 sklearn.metrics.classification_report ()

: A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes.

6.9.5 sklearn.metrics.precision_recall_fscore_support ()

: Compute precision, recall, F-measure and support for each class

The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights recall more than precision by a factor of beta. beta == 1.0 means recall and precision are equally important.

The support is the number of occurrences of each class in y_true.

If pos_label is None and in binary classification, this function returns the average precision, recall and F-measure if average is one of 'micro', 'macro', 'weighted' or 'samples'.

6.9.6 sklearn.metrics.roc_auc_score ()

: Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

Note: this implementation can be used with binary, multiclass and multilabel classification.

6.9.7 sklearn.metrics.roc_curve ()

: Compute Receiver operating characteristic (ROC)

6.9.8 sklearn.metrics.accuracy_score ()

: Accuracy classification score.

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

6.9.9 sklearn.multiclass.OneVsOneClassifier ()

: One-vs-one multiclass strategy,

This strategy consists in fitting one classifier per class pair. At prediction time, the class which received the most votes is selected. Since it requires to fit n_classes * (n_classes - 1) / 2 classifiers, this method is usually slower than one-vs-the-rest, due to its O(n_classes$\hat{2}$) complexity. However, this method may be advantageous for algorithms such as kernel algorithms which don't scale well with n_samples. This is because each individual learning problem only involves a small subset of the data whereas, with one-vs-the-rest, the complete dataset is used n_classes times.

6.9.10 sklearn.preprocessing.label_binarize ()

: Binarize labels in a one-vs-all fashion

Several regression and binary classification algorithms are available in scikit-learn. A simple way to extend these algorithms to the multi-class classification case is to use the so-called one-vs-all scheme.

This function makes it possible to compute this transformation for a fixed set of class labels known ahead of time.

6.10 SciPy

SciPy is an open-source Python library which is used to solve scientific and mathematical problems. It is built on the NumPy extension and allows the user to manipulate and visualize data with a wide range of high-level commands. As mentioned earlier, SciPy builds on NumPy and therefore if you import SciPy, there is no need to import NumPy.

Use of SciPy

- SciPy contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.
- SciPy is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

Special Function package

- scipy.special package contains numerous functions of mathematical physics.
- SciPy special function includes Cubic Root, Exponential, Log sum Exponential, Lambert, Permutation and Combinations, Gamma, Bessel, hypergeometric, Kelvin, beta, parabolic cylinder, Relative Error Exponential, etc.

6.11 Binary Crossentropy

Binary crossentropy is a loss function that is used in binary classification tasks. These are tasks that answer a question with only two choices (yes or no, A or B, 0 or 1, left or right). Several independent such questions can be answered at the same time, as in multi-label classification or in binary image segmentation. Formally, this loss is equal to the average of the categorical crossentropy loss on many two-category tasks.

6.11.1 Math in Binary Crossentropy

The binary crossentropy loss function calculates the loss of an example by computing the following average:

$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figure 6.4: Binary Crossentropy equation

where y cap i is the ith scalar value in the model output, yi is the corresponding target value, and output size is the number of scalar values in the model output.

This is equivalent to the average result of the categorical crossentropy loss function applied to many independent classification problems, each problem having only two possible classes with target probabilities yi and (1-yi) .

6.12 Word Embedding

A word embedding is a learned representation for text where words that have the same meaning have a similar representation.

It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

Key to the approach is the idea of using a dense distributed representation for each word.

Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one-hot encoding.

6.12.1 Embedding Layer

An embedding layer, for lack of a better name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification.

It requires that document text be cleaned and prepared such that each word is one-hot encoded. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.

The one-hot encoded words are mapped to the word vectors. If a multilayer Perceptron model is used, then the word vectors are concatenated before being fed as input to the model. If a recurrent neural network is used, then each word may be taken as one input in a sequence.

This approach of learning an embedding layer requires a lot of training data and can be slow, but will learn an embedding both targeted to the specific text data and the NLP task.

6.12.2 Word2Vec

Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus.

It was developed by Tomas Mikolov, et al. at Google in 2013 as a response to make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pre-trained word embedding.

Additionally, the work involved analysis of the learned vectors and the exploration of vector math on the representations of words. For example, that subtracting the "man-ness" from "King" and adding "women-ness" results in the word "Queen", capturing the analogy "king is to queen as man is to woman".

We find that these representations are surprisingly good at capturing syntactic and semantic regularities in language, and that each relationship is characterized by a relation-specific vector offset. This allows vector-oriented reasoning based on the offsets between words. For example, the male/female relationship is automatically learned, and with the induced vector representations, "King – Man + Woman" results in a vector very close to "Queen."

Two different learning models were introduced that can be used as part of the word2vec approach to learn the word embedding; they are:

- Continuous Bag-of-Words, or CBOW model.
- Continuous Skip-Gram Model.

The CBOW model learns the embedding by predicting the current word based on its context. The continuous skip-gram model learns by predicting the surrounding words given a current word.

The continuous skip-gram model learns by predicting the surrounding words given a current word.

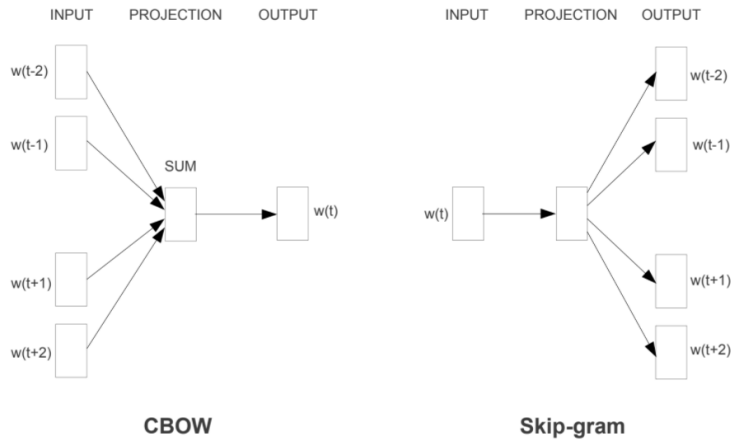


Figure 6.5: CBOW & Skip-gram

Both models are focused on learning about words given their local usage context, where the context is defined by a window of neighboring words. This window is a configurable parameter of the model.

The size of the sliding window has a strong effect on the resulting vector similarities. Large windows tend to produce more topical similarities [_], while smaller windows tend to produce more functional and syntactic similarities.

The key benefit of the approach is that high-quality word embeddings can be learned efficiently (low space and time complexity), allowing larger embeddings to be learned (more dimensions) from much larger corpora of text (billions of words).

6.12.3 GloVe

GloVe stands for global vectors for word representation, or GloVe, algorithm is an extension to the word2vec method for efficiently learning word vectors, developed by Pennington, et al. at Stanford.

Classical vector space model representations of words were developed using matrix factorization techniques such as Latent Semantic Analysis (LSA) that do a good job of using global text statistics but are not as good as the learned methods like word2vec at capturing meaning and demonstrating it on tasks like calculating analogies (e.g. the King and Queen example above).

GloVe is an approach to marry both the global statistics of matrix factorization techniques like LSA with the local context-based learning in word2vec.

Rather than using a window to define local context, GloVe constructs an explicit word-context or word co-occurrence matrix using statistics across the whole text corpus. The result is a learning model that may result in generally better word embeddings.

GloVe, is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity,

and named entity recognition tasks.

We have used GloVE Embedding with vector size of 300 Dimensions for our model.

Chapter 7

Neural Networks

In the past 10 years, the best-performing artificial-intelligence systems — such as the speech recognizers on smartphones or Google's latest automatic translator — have resulted from a technique called "deep learning." [19]

Deep learning is in fact a new name for an approach to artificial intelligence called neural networks, which have been going in and out of fashion for more than 70 years.

7.1 What is a Neural Network?

Neural Networks are inspired by biological neural network (human brain).

It's a computing system with interconnected nodes called neurons. Using algorithms, they can recognize hidden patterns and correlations in raw data, cluster and classify it and can countinuously learn and improve.

A neural network having more than one hidden layer is generally referred to as a **Deep Neural Network.**

NNs began as an attempt to exploit the architecture of the human brain to perform tasks that conventional algorithms had little success with. They are more properly referred to as an 'artificial' neural network (ANN).

Neural nets are a means of doing machine learning, in which a computer learns to perform some task by analyzing training examples. Usually, the examples have been hand-labeled in advance. An object recognition system, for instance, might be fed thousands of labeled images of cars, houses, coffee cups, and so on, and it would find visual patterns in the images that consistently correlate with particular labels.[19]]

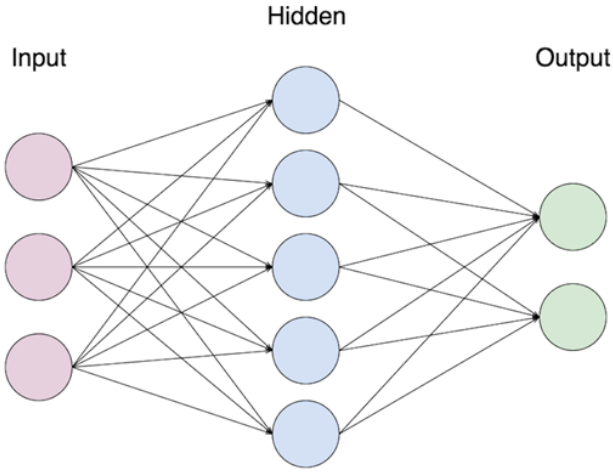


Figure 7.1: A Simple Neural Network

7.1.1 History of Neural Network

- In 1943 , Warren mcCulloch and Walter Pitts created first NN model. It was purely based on mathematics and algorithms but couldn't be tested due to lack of computational resources.
- In 1958, Frank Rosenblatt created the first ever model that could do pattern recognition. He introduced "Perceptron" (A Neural Network Model or an Artificial Neuron).
- he first NNs that could be tested and had many layers were published by Alexey Ivakhnenko and Lapa in 1965.

7.1.2 Basic Structure of Neural Network

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

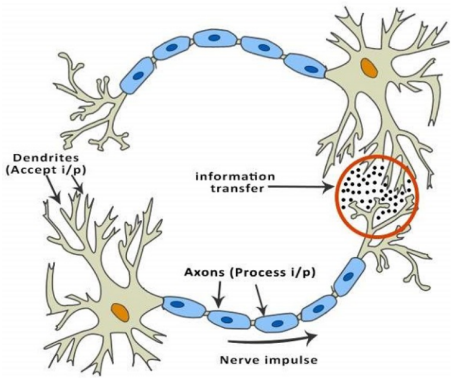


Figure 7.2: Biological Neuron

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values.
There are 3 parts that make up the architecture of a basic Neural Network. These are:

1. Units / Neurons

Neuron is a mathematical function that works same as biological neuron. So what does a neuron do?
It takes the inputs and multiplies them by their weights,then it sums them up after that it applies the activation function (eg sigmoid function) to the sum.

2. Connections / Weights / Parameters

- Weights (commonly referred to as w) are the learnable parameters of a machine learning model.
- When the inputs are transmitted between neurons, the weights are applied to the inputs along with the bias. A neuron. Weights control the signal (or the strength of the connection) between two neurons.

3. Biases

Bias is like the intercept added in a linear equation. It is an additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron. Thus, Bias is a constant which helps the model in a way that it can fit best for the given data.

It is commonly referred to as 'b' and like weights, bias are also the learnable parameters of a machine learning model.

7.1.3 Layers

1. These are what help an NN gain complexity in any problem. Increasing layers(with units) can increase the non-linearity of the output of an NN.
2. Each layer contains some amount of units or neurons.
3. There are 2 layers which every NN has. Those are the input and output layers.
4. Any layer in between those is called a hidden layer.

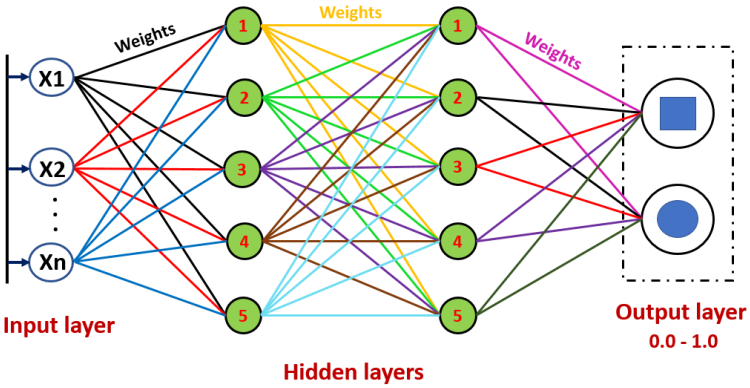


Figure 7.3: Structure of Basic Neural Network

7.1.4 Activation Function

These are also known as **mapping functions**. They take some input on the x-axis and output a value in a restricted range(mostly).
The function used in a neuron is generally termed as an activation function. There have been 5 major activation functions tried to date : step, sigmoid, tanh, ReLU and leaky ReLU. Each of these is described in detail below.[20]

1. Step Function

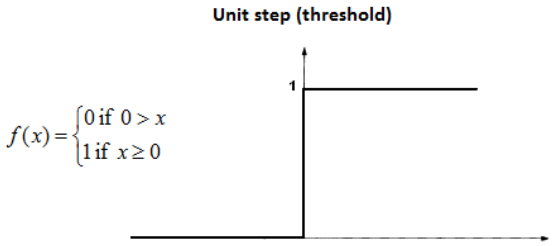


Figure 7.4: Step Function

Where the output is 1 if the value of x is greater than equal to zero and 0 if the value of x is less than zero. As one can see a step function is non-differentiable at zero. At present, a neural network uses back propagation method along with gradient descent to calculate weights of different layers. Since the step function is non-differentiable at zero hence it is not able to make progress with the gradient descent approach and fails in the task of updating the weights.

To overcome, this problem sigmoid functions were introduced instead of the step function.

2. Sigmoid Function

A sigmoid function or logistic function is defined mathematically as Figure 7.5

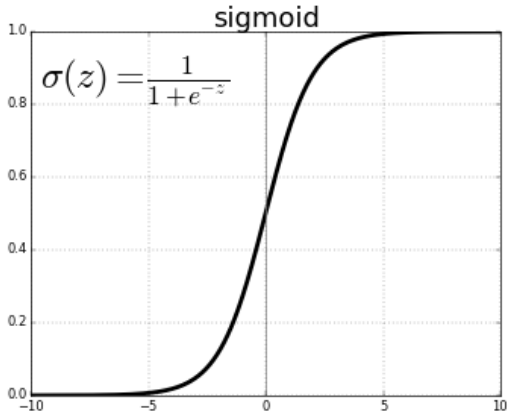


Figure 7.5: Sigmoid Function

The value of the function tends to zero when z or independent variable tends to negative infinity and tends to 1 when z tends to infinity. It needs to be kept in mind that this function represents an approximation of the behavior of the dependent variable and is an assumption. Now the question arises as to why we use the sigmoid function as one of the approximation functions. There are certain simple reasons for this.

- It captures non-linearity in the data. Albeit in an approximated form, but the concept of non-linearity is essential for accurate modeling.
- The sigmoid function is differentiable throughout and hence can be used with gradient descent and backpropagation approaches for calculating weights of different layers.
- The assumption of a dependent variable to follow a sigmoid function inherently assumes a Gaussian distribution for the independent variable which is a general distribution we see for a lot of randomly occurring events and this is a good generic distribution to start with.

However, a sigmoid function also suffers from a problem of vanishing gradients. As can be seen from the picture a sigmoid function squashes it's input into a very small output range [0,1] and has very steep gradients. Thus, there remain large regions of input space, where even a large change produces a very small change in the output. This is referred to as the problem of vanishing gradient. This problem increases with an increase in the number of layers and thus stagnates the learning of a neural network at a certain level.

Tanh Function

The tanh(z) function is a rescaled version of the sigmoid, and its output range is [1,1] instead of [0,1]. [1]

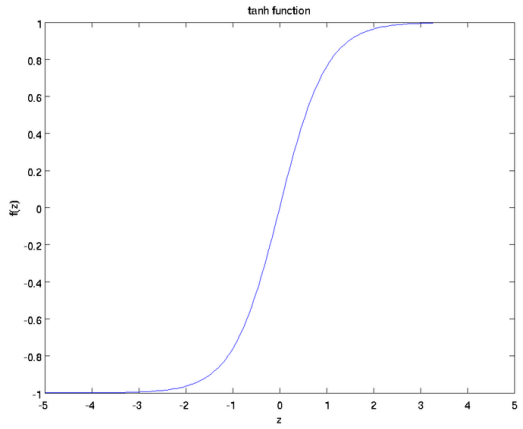
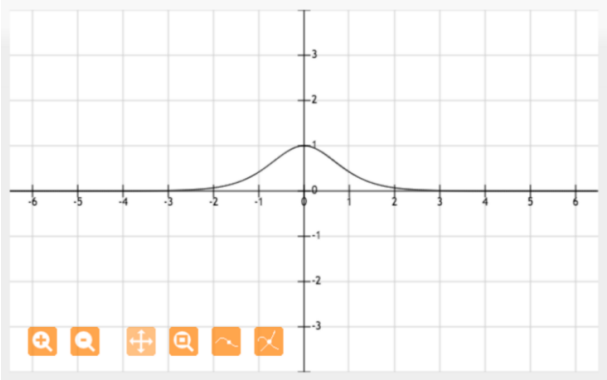


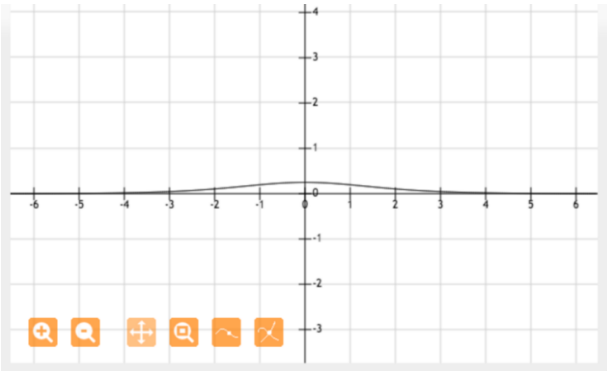
Figure 7.6: Tanh Function

The general reason for using a Tanh function in some places instead of the sigmoid function is because since data is centered around 0, the derivatives are higher. A higher gradient helps in a better learning rate. Below attached are plotted gradients of two functions tanh and sigmoid.

For tanh function, for an input between [-1,1], we have derivative between [0.42, 1].



For sigmoid function on the other hand, for input between [0,1], we have derivative between [0.20, 0.25]



As one can see from the pictures above a Tanh function has a higher range of

derivative than a Sigmoid function and thus has a better learning rate. However, the problem of vanishing gradients still persists in Tanh function.

4. ReLU Function

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. So, it can be written as f(x)=max (0, x).

Graphically it looks like this :

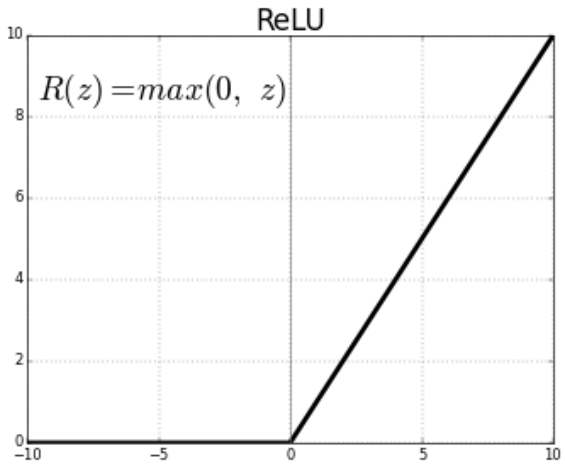


Figure 7.7: ReLU Function

The Leaky ReLU is one of the most well-known. It is the same as ReLU for positive numbers. But instead of being 0 for all negative values, it has a constant slope (less than 1.).

That slope is a parameter the user sets when building the model, and it is frequently called . For example, if the user sets =0.3, the activation function is f(x) = max (0.3*x, x) This has the theoretical advantage that, by being influenced, by x at all values, it may make more complete use of the information contained in x.

There are other alternatives, but both practitioners and researchers have generally found an insufficient benefit to justify using anything other than ReLU. In general practice as well, ReLU has found to be performing better than sigmoid or tanh functions.

7.1.5 Working of a Neural Network

A neural network has many layers. Each layer performs a specific function, and the complex the network is, the more the layers are. That's why a neural network is also called a multi-layer perceptron.

The purest form of a neural network has three layers:

1. The input layer
2. The hidden layer
3. The output layer

As the names suggest, each of these layers has a specific purpose. These layers are made up of nodes. There can be multiple hidden layers in a neural network according to the requirements. The input layer picks up the input signals and transfers them to the next layer. It gathers the data from the outside world.

The hidden layer performs all the back-end tasks of calculation. A network can even have zero hidden layers. However, a neural network has at least one hidden layer. The output layer transmits the final result of the hidden layer's calculation.

You will have to train a neural network with some training data as well, before you provide it with a particular problem.

How do Perceptron Layers Work?

A neural network is made up of many perceptron layers; that's why it has the name 'multi-layer perceptron.' These layers are also called hidden layers of dense layers. They are made up of many perceptron neutrons. They are the primary unit that works together to form a perceptron layer. These neurons receive information in the set of inputs. You combine these numerical inputs with a bias and a group of weights, which then produces a single output.

For computation, each neuron considers weights and bias. Then, the combination function uses the weight and the bias to give an output (modified input). It works through the following equation:

combination = bias +weights * inputs

After this, the activation function produces the output with the following equation:

output = activation(combination)

This function determines what kind of role the neural network performs. They form the layers of the network. The following are the prevalent activation functions:

1. Linear Function

In this function the output is only the combination of the neuron:

activation = combination

The hyperbolic Tangent Function

It is the most popular activation function among neural networks. It is a sigmoid function, and it lies between -1 and +1:

activation = tanh(combination)

2. Logistic Function

The logistic function is quite similar to the hyperbolic tangent function because it is a kind of sigmoid function, as well. However, it is different because it lies between 0 and 1:

activation = 11 + e-combination

3. The Rectified Linear Unit Function

Just like the hyperbolic tangent function, the rectified linear unit function is also prevalent. Another name for the rectified linear unit function is ReLU. ReLU is equal to the combination when it is equal to or greater than zero, and it's negative if the combination is lower than (negative) zero.

How NN works:

Neural networks learning process is not very different from humans, humans learns from experience in lives while neural networks require data to gain experience and learn. Accuracy increases with the amount of data over time. Similarly, humans also perform the same task better and better by doing any task you do over and over. [53]

The underlying foundation of neural networks is a layer and layers of connections. The entire neural network model is based on a layered architecture. Each layer has its own responsibility. These networks are designed to make use of layers of "neurons" to process raw data, find patterns into it and objects which are usually hidden to naked eyes.

1. Information is fed into the input layer which transfers it to the hidden layer,
2. The interconnections between the two layers assign weights to each input randomly,
3. A bias added to every input after weights are multiplied with them individually,
4. The weighted sum is transferred to the activation function,
5. The activation function determines which nodes it should fire for feature extraction,
6. The model applies an application function to the output layer to deliver the output,
7. Weights are adjusted, and the output is back-propagated to minimize error.

The model uses a cost function to reduce the error rate. You will have to change the weights with different training models.

Cost Function:

Cost Function is often the squared of the difference between actual and predicted outcome, because the difference sometimes can be negative. this is also known as mean squared error (MSE) function.

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

Figure 7.8: Cost Function

1. The model compares the output with the original result
2. It repeats the process to improve accuracy.

The model adjusts the weights in every iteration to enhance the accuracy of the output.

To train a neural network, data scientist put their data in three different baskets.[21]

- Training data set – This helps networks to understand and know the various weights between nodes.
- Validation data set – To fine-tune the data sets.
- Test data set – To evaluate the accuracy and records margin of error.

Layer takes input, extract feature and feed into the next layer i.e. each layer work as an input layer to another layer. This is to receive information and last layer job is to throw output of the required information. Hidden layers or core layers process all the information in between.

The neural network is a weighted graph where nodes are the neurons and the connections are represented by edges with weights. It takes input from the outside world and is denoted by x(n).

Each input is multiplied by its respective weights and then they are added. A bias is added if the weighted sum equates to zero, where bias has input as 1 with weight b. Then this weighted sum is passed to the activation function. The activation function

limits the amplitude of the output of the neuron. There are various activation functions like Threshold function, Piecewise linear function or Sigmoid function.

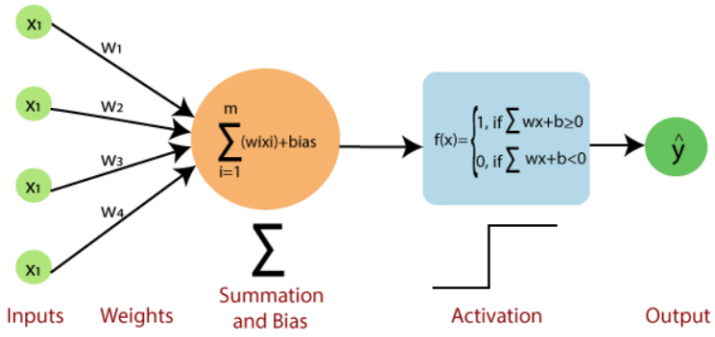


Figure 7.9: Threshold Function

7.1.6 Backpropagation

The backpropagation algorithm is commonly used for improving the performance of neural network prediction accuracy. It's done by adjusting higher weight connections in an attempt to lower the cost function.

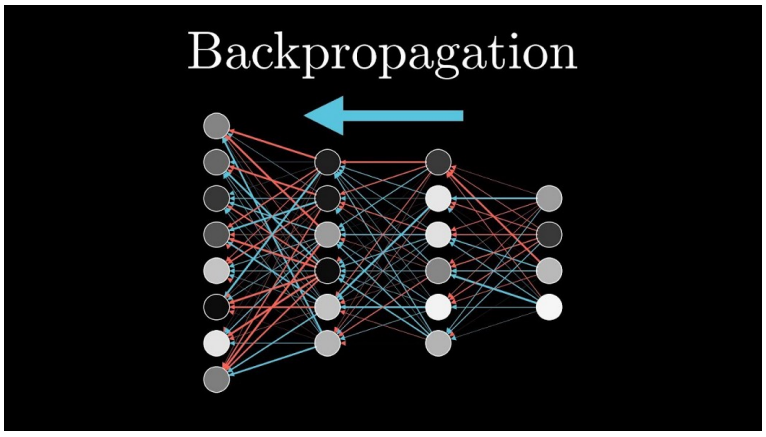


Figure 7.10: Backpropagation

Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.[22]

Why We Need Backpropagation?

Most prominent advantages of Backpropagation are:

- Backpropagation is fast, simple and easy to program.
- It has no parameters to tune apart from the numbers of input.
- It is a flexible method as it does not require prior knowledge about the network.

- It is a standard method that generally works well.
- It does not need any special mention of the features of the function to be learned.

Types of Backpropagation Networks

Two Types of Backpropagation Networks are: [22]

1. Static Back-propagation
2. Recurrent Backpropagation

1) Static back-propagation:

It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

2) Recurrent Backpropagation:

Recurrent backpropagation is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.
The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation.

History of Backpropagation

- In 1961, the basics concept of continuous backpropagation were derived in the context of control theory by J. Kelly, Henry Arthur, and E. Bryson.
- In 1969, Bryson and Ho gave a multi-stage dynamic system optimization method.
- In 1974, Werbos stated the possibility of applying this principle in an artificial neural network.
- In 1982, Hopfield brought his idea of a neural network.
- In 1986, by the effort of David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, backpropagation gained recognition.
- In 1993, Wan was the first person to win an international pattern recognition contest with the help of the backpropagation method.

Backpropagation Key Points

- Simplifies the network structure by elements weighted links that have the least effect on the trained network.
- You need to study a group of input and activation values to develop the relationship between the input and hidden unit layers.

- It helps to assess the impact that a given input variable has on a network output. The knowledge gained from this analysis should be represented in rules.
- Backpropagation is especially useful for deep neural networks working on error-prone projects, such as image or speech recognition.
- Backpropagation takes advantage of the chain and power rules allows backpropagation to function with any number of outputs.

Disadvantages of using Backpropagation

- The actual performance of backpropagation on a specific problem is dependent on the input data.
- Backpropagation can be quite sensitive to noisy data
- You need to use the matrix-based approach for backpropagation instead of mini-batch.

Gradient Descent

- Gradient Descent is a optimization algorithm
- Steps of gradient Descent:

1) Random Initialization
2) Partial Derivative of cost function
3) After computing the derivative ,update the parameter

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \implies \frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \cdot x_i \implies \frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

$$a_0 = a_0 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$a_1 = a_1 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

Figure 7.11: Gradient Descen

7.1.7 Types of Neural Networks

1) Recurrent Neural Network (RNN)

In this network, the output of a layer is saved and transferred back to the input. This way, the nodes of a particular layer remember some information about the past steps. The combination of the input layer is the product of the sum of weights and features. The recurrent neural network process begins in the hidden layers.

Here, each node remembers some of the information of its antecedent step. The model retains some information from each iteration, which it can use later. The system

self-learns when its outcome is wrong. It then uses that information to increase the accuracy of its prediction in back-propagation. The most popular application of RNN is in text-to-speech technology.

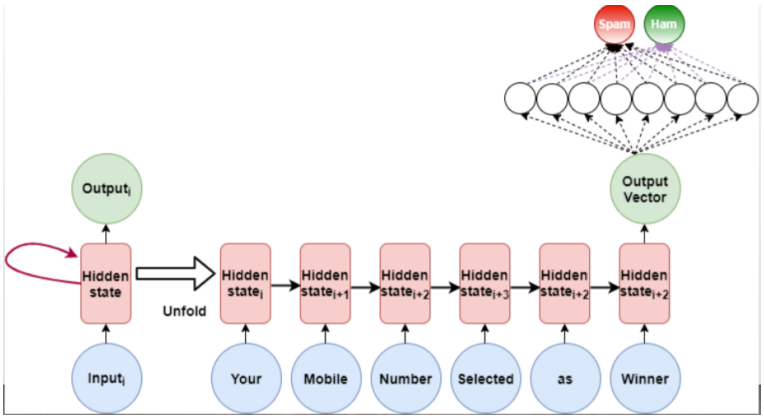


Figure 7.12: Recurrent Neural Networks

2) Convolutional Neural Network (CNN)

This network consists of one or multiple convolutional layers. The convolutional layer present in this network applies a convolutional function on the input before transferring it to the next layer. Due to this, the network has fewer parameters, but it becomes more profound. CNNs are widely used in natural language processing and image recognition.

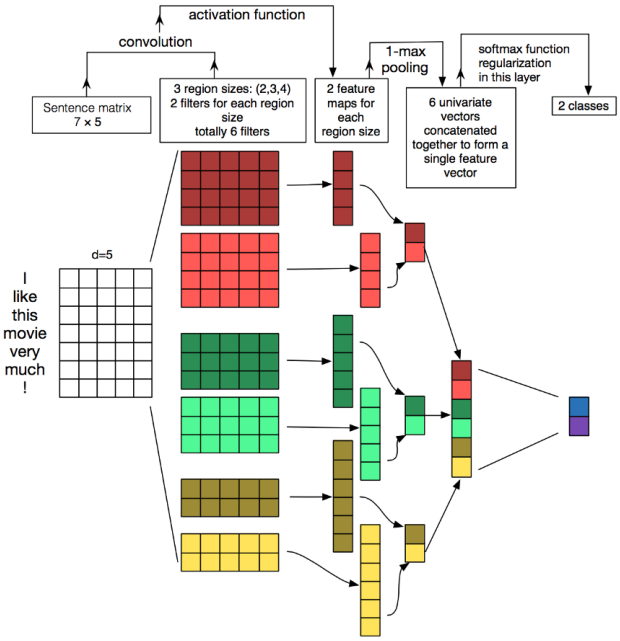


Figure 7.13: Convolutional Neural Networks

3) Feedforward Neural Network (FNN)

This is the purest form of an artificial neural network. In this network, data moves in one direction, i.e., from the input layer to the output layer. In this network, the output layer receives the sum of the products of the inputs and their weights. There's no back-propagation in this neural network. These networks could have many or zero hidden layers. These are easier to maintain and find application in face recognition.

There are no feedback loops. They are used in pattern generation/recognition/classification. They have fixed inputs and outputs.

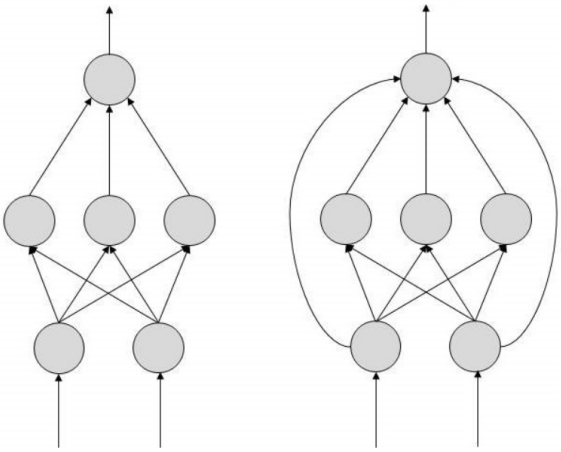


Figure 7.14: Feedforward Neural Network

5) FeedBack ANN

Here, feedback loops are allowed. They are used in content addressable memories.

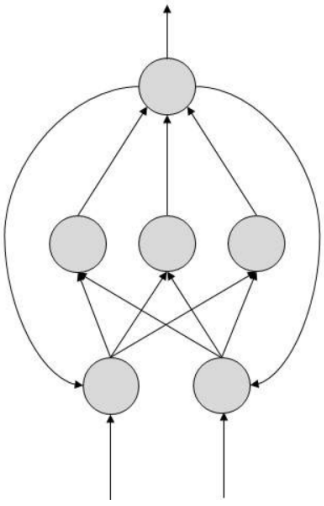


Figure 7.15: FeedBack ANN

7.1.8 Advantages of Neural Network

1. Can work with incomplete information once trained.

2. Have ability of fault tolerance.

3. Have a distributed memory.

4. Can make machine learning.

5. Parallel processing.

6. Stores information on an entire network

7. Can learn non-linear and complex relationships.

8. Ability to generalize i.e. can infer unseen relationships after learning from some prior relationships.

7.1.9 Applications of Neural Networs

They can perform tasks that are easy for a human but difficult for a machine [23]

- **Aerospace** : Autopilot aircrafts, aircraft fault detection.
- **Automotive** : Automobile guidance systems.
- **Military** : Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.
- **Electronics** : Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.
- **Financial** : Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.
- **Industrial** : Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modeling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.
- **Medical** : Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
- **Speech** : Speech recognition, speech classification, text to speech conversion.
- **Telecommunications** : Image and data compression, automated information services, real-time spoken language translation.
- **Transportation** : Truck Brake system diagnosis, vehicle scheduling, routing systems.

- **Software** : Pattern Recognition in facial recognition, optical character recognition, etc.
- **Time Series Prediction** : ANNs are used to make predictions on stocks and natural calamities.
- **Signal Processing** : Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.
- **Control** : ANNs are often used to make steering decisions of physical vehicles.
- **Anomaly Detection** : As ANNs are expert at recognizing patterns, they can also be trained to generate an output when something unusual occurs that misfits the pattern.

7.2 CNN

A convolutional neural network— also called CNN or ConvNet, is a Deep Learning algorithm. It takes an input image, assigns weights/ biases to the components of the image, and then classifies the entire image.[24]

Convolutional Neural Networks (CNNs) are the backbone of image classification, a deep learning phenomenon that takes an image and assigns it a class and a label that makes it unique. Image classification using CNN forms a significant part of machine learning experiments.

They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.[25]

7.2.1 Structure of CNN

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.[26]

A Convolutional Neural Network comprises of 3 major layers. Each layer tries to find a pattern or useful information of the data. These are:

1. The convolutional layer
2. The pooling layer
3. The fully connected layer

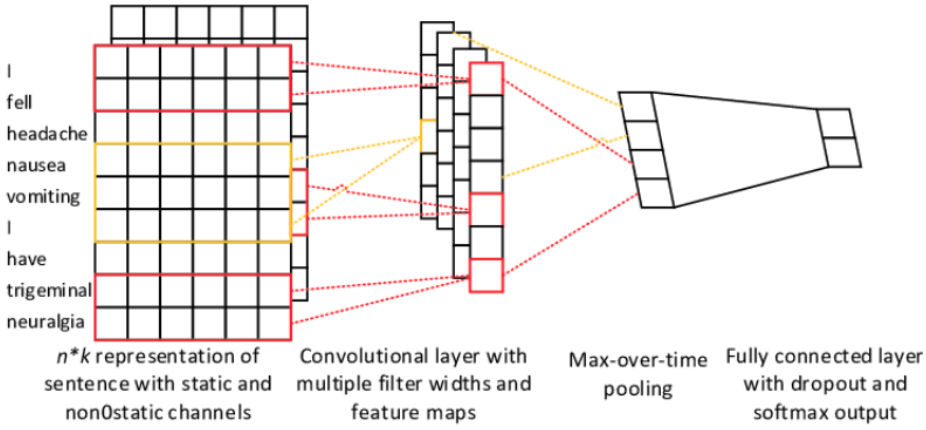


Figure 7.16: CNN

1. The convolutional layer

This is the core layer of the convolutional neural network. Its parameters are composed of a set of filters. These filters are small, but they cover the full depth of the input volume.

Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.

The main task performed at the convolutional layer is the extraction of high-level features. The first one is responsible for extracting low-level features like color, edges, etc. The subsequent convolutional layers take out the high-level features, thus, leading to a complete understanding/ perusal of the image.

The Pooling Layer

This layer is meant to reduce the spatial size of the image representation. As such, it also helps to reduce the computation and processing amount in the neural network. Additionally, it also extracts dominant features that are positionally and rotationally invariant.

There are two types of Pooling: Max Pooling and Average Pooling.

1. **Max Pooling** returns the maximum value from the portion of the image covered by the Kernel.
2. **Average Pooling**returns the average of all the values from the portion of the image covered by the Kernel.

There are multiple pooling layers in addition to convolutional layers. Greater the number of these layers, the more low-level features will be extracted. However, computational power expended will also increase.

The Fully Connected Layers (FCL)

As the last layer, the FC layer is simply a feed-forward neural network. The input to the fully connected layer is the flattened output of the last pooling/ convolutional layer. To flatten means that the 3-dimensional matrix or array is unrolled into a vector.

For each FC layer, a specific mathematical calculation takes place. After the vector has passed through all the fully connected layers, the softmax activation function is used in the final layer. This is used to compute the probability of the input belonging to a particular task.

Thus, the end result is the different probabilities of the input image belonging to different classes.

The process is repeated for different types of images and individual images within those types. This trains the network and teaches it to differentiate between a dog and a cat, and a rose and a sunflower.

7.2.2 Hyperparameters of CNN

Model Parameters are the properties of training data that will learn on its own during training by the classifier or other ML model. For example, weights and biases, or split points in Decision Tree.

Model Hyperparameters are instead properties that govern the entire training process. They include variables which determines the network structure (for example, Number of Hidden Units) and the variables which determine how the network is trained (for example, Learning Rate). Model hyperparameters are set before training (before optimizing the weights and bias).[27]

Hyperparameters are important since they directly control behavior of the training algo, having important impact on performance of the model under training.

Choosing appropriate hyperparameters plays a key role in the success of neural network architectures, given the impact on the learned model. For instance, if the learning rate is too low, the model will miss the important patterns in the data; conversely, if it is high, it may have collisions.[27]

Choosing good hyperparameters provides two main benefits:

- Efficient search across the space of possible hyperparameters; and
- Easier management of a large set of experiments for hyperparameter tuning.

For example, here are some model inbuilt configuration variables :

- Learning Rate

- Number of Epochs
- Hidden Layers
- Hidden Units
- Activations Functions

Hyperparameters can be roughly divided into 2 categories:

1. Optimizer hyperparameters
2. Model Specific hyperparameters

1. Optimization Hyperparameters

They are related more to the optimization and training process.

A) Learning rate:

If the model learning rate is way too smaller than optimal values, it will take a much longer time (hundreds or thousands) of epochs to reach an ideal state. On the other hand, if the learning rate is much larger than optimal value, then it would overshoot the ideal state and the algorithm might not converge. A reasonable starting learning rate = 0.001.

The model will have have hundreds and thousands of parameters each with its own error curve. And learning rate has to shepherd all of them.

B) Batch size:

Batch size has an effect on the resource requirements of the training process, speed and number of iterations in a non-trivial way.

Historically, there has been a debate to do stochastic training where you fit a single example of the dataset to the model and, using only one example, perform a forward pass, calculate the error / backpropagate & set adjusted values for all the hyperparameters. And then do this again for each example in the dataset.

Or, perhaps better to feed the entire data to the
training step and calculate the gradient using the error generated by looking at all the examples in the dataset. This is called **batch training.**

Recommended starting values for experimentation: 1, 2, 4, 8, 16, 32, 64, 128, 256. Commonly used technique today is to set a **mini-batch size**. A fair value for mini-batch size= 32.

C) Number of Epochs:

To choose the right number of epochs for the training step, the metric we should pay attention to is the **Validation Error.**

There's a technique that can be used named **Early Stopping** to determine when to stop training the model; it is about stopping the training process in case the validation error has not improved in the past 10 or 20 epochs.

2. Model Specific Hyperparameters

They are more involved in the structure of the model:

A) Number of hidden units:

Number of hidden units is one of the more mysterious hyperparameters. The number of hidden units is the main measure of model's learning capacity.

For a simple function, it might need fewer number of hidden units. The more complex the function, the more learning capacity the model will need. Slightly more number of units then optimal number is not a problem, but a much larger number will lead to overfitting (i.e. if you provide a model with too much capacity, it might tend to overfit, trying to "memorize" the dataset, therefore affecting capacity to generalize)

B) First hidden layer:

Another heuristic involving the first hidden layer is that setting the number of hidden units larger than the number of inputs tends to enable better results in number of tasks, according to empirical observation.

C) Number of layers:

It is often the case that 3-layer Neural Net will outperform a 2-layer one. But going even deeper rarely helps much more. (exception are Convolutional Neural Networks, where the deeper they are, the better they perform).

7.2.3 Stride Size

Another hyperparameter for your convolutions is the stride size, defining by how much you want to shift your filter at each step. In all the examples above the stride size was 1, and consecutive applications of the filter overlapped. A larger stride size leads to fewer applications of the filter and a smaller output size.[28]

The following shows stride sizes of 1 and 2 applied to a one-dimensional input:

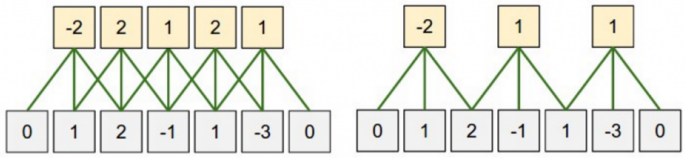


Figure 7.17: Convolution Stride Size. Left: Stride size 1. Right: Stride size 2

In the literature we typically see stride sizes of 1, but a larger stride size may allow you to build a model that behaves somewhat similarly to a Recursive Neural Network, i.e. looks like a tree.

7.2.4 Working of CNN for text classification

Convolutional Neural Networks (ConvNets) have in the past years shown break-through results in some NLP tasks, one particular task is sentence classification, i.e., classifying short phrases (i.e., around 20 50 tokens), into a set of pre-defined categories.

Text classification tasks generally involve classifying a sentence into one of question types and require scanning across the sentence and building up a representation of its constituent words.

An end-to-end text classification pipeline is composed of following components:

1. Training text: It is the input text through which our supervised learning model is able to learn and predict the required class.
2. Feature Vector: A feature vector is a vector that contains information describing the characteristics of the input data.
3. Labels: These are the predefined categories/classes that our model will predict.
4. ML Algo: It is the algorithm through which our model is able to deal with text classification (In our case : CNN, RNN, HAN).
5. Predictive Model: A model which is trained on the historical dataset which can perform label predictions.

A simple CNN architecture for classifying texts

We use a pre-defined word embedding available from the library. Generally, if the data is not embedded then there are many various embeddings available open-source like **Glove and Word2Vec.** [29]

An example of a sentence convolution in a vector-concatenation notation:

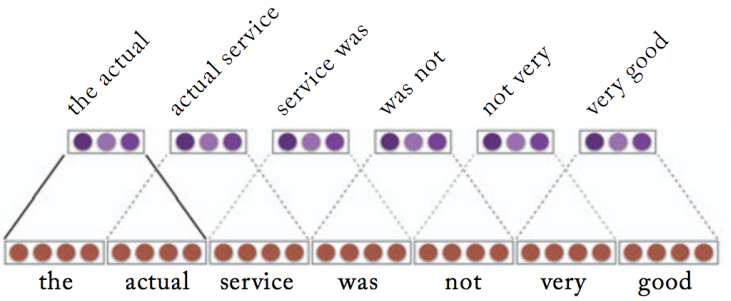


Figure 7.18: Example of a sentence convolution with k=2 and dimensional output l=3

When we do dot product of vectors representing text, they might turn out zero even when they belong to same class but if you do dot product of those embedded word vectors to find similarity between them then you will be able to find the interrelation of words for a specific class.

Then, we slide the filter/ kernel over these embeddings to find convolutions and these are further dimensionally reduced in order to reduce complexity and computation by the Max Pooling layer.

Pooling

The pooling operation is used to combine the vectors resulting from different convolution windows into a single l-dimensional vector. This is done again by taking the max or the average value observed in resulting vector from the convolutions. Ideally this vector will capture the most relevant features of the sentence/document.

This vector is then fed further down in the network. Hence, the idea that ConvNet itself is just a feature extractor, most probably to a full connected layer to perform prediction.

Lastly, we have the fully connected layers and the activation function on the outputs that will give values for each class.

7.3 BiLSTM

To understand BiLSTM , we must first go through the insights of RNN and it's limitations.

7.3.1 RNN

Recurrent Neural Networks or RNN as they are called in short, are a very important variant of neural networks heavily used in Natural Language Processing.

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks,

all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.[30]

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.

RNN has shown to be hugely successful in natural language processing especially with their variant LSTM, which are able to look back longer than RNN.

Advantages of Recurrent Neural Network

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighborhood.

Limitations of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.
4. Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. So if you are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning.[31]
5. During back propagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update a neural networks weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning.[31]

7.3.2 History

- **1997:** LSTM was proposed by Sepp Hochreiter and Jürgen Schmidhuber.[1] By introducing Constant Error Carousel (CEC) units, LSTM deals with the vanishing gradient problem. The initial version of LSTM block included cells, input and output gates.[32]

- **1999:** Felix Gers and his advisor Jürgen Schmidhuber and Fred Cummins introduced the forget gate (also called "keep gate") into LSTM architecture, enabling the LSTM to reset its own state.

- **2000:** Gers & Schmidhuber & Cummins added peephole connections (connections from the cell to the gates) into the architecture. Additionally, the output activation function was omitted.

- **2009:** An LSTM based model won the ICDAR connected handwriting recognition competition. Three such models were submitted by a team lead by Alex Graves. One was the most accurate model in the competition and another was the fastest.

- **2013:** LSTM networks were a major component of a network that achieved a record 17.7% phoneme error rate on the classic TIMIT natural speech dataset.

- **2014:** Kyunghyun Cho et al. put forward a simplified variant called Gated recurrent unit (GRU).

- **2015:** Google started using an LSTM for speech recognition on Google Voice. According to the official blog post, the new model cut transcription errors by 49%.

- **2016:** Google started using an LSTM to suggest messages in the Allo conversation app. In the same year, Google released the Google Neural Machine Translation system for Google Translate which used LSTMs to reduce translation errors by 60%.

  Apple announced in its Worldwide Developers Conference that it would start using the LSTM for quicktype in the iPhone and for Siri.

  Amazon released Polly, which generates the voices behind Alexa, using a bidirectional LSTM for the text-to-speech technology.

- **2017:** Facebook performed some 4.5 billion automatic translations every day using long short-term memory networks.

  Researchers from Michigan State University, IBM Research, and Cornell University published a study in the Knowledge Discovery and Data Mining (KDD) conference. Their study describes a novel neural network that performs better on certain data sets than the widely used long short-term memory neural network.

  Microsoft reported reaching 94.9% recognition accuracy on the Switchboard corpus, incorporating a vocabulary of 165,000 words. The approach used "dialog session-based long-short-term memory".

- **2019:** Researchers from the University of Waterloo proposed a related RNN architecture which represents continuous windows of time. It was derived using the Legendre polynomials and outperforms the LSTM on some memory-related benchmarks.

7.3.3 What is BiLSTM?

LSTM networks were designed for long term dependencies, therefore the idea which makes it different from other neural network is that it is able to remember information for a long span of time without learning, again and again, making this whole process simpler and faster. This type of recurrent neural network includes an inbuilt memory system for storing information.[33]

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems.[34] The purpose of the Bi-LSTM is to look at a particular sequences both from front-to-back as well as from back-to-front. In this way, the network creates a context for each character in the text that depends on both its past as well as its future.[35]

Where all time steps of the input sequence are available, Bi-LSTMs train two LSTMs instead of one LSTMs on the input sequence. The first on the input sequence as-is and the other on a reversed copy of the input sequence. By this additional context is added to network and results are faster.[36]

Bidirectional Long/Short-Term Memory (BiLSTM) look exactly the same as its unidirectional counterpart. The difference is that the network is not just connected to the past, but also to the future. As an example, unidirectional LSTMs might be trained to predict the word "fish" by being fed the letters one by one, where the recurrent connections through time remember the last value. A BiLSTM would also be fed the next letter in the sequence on the backward pass, giving it access to future information. This trains the network to fill in gaps instead of advancing information, so instead of expanding an image on the edge, it could fill a hole in the middle of an image.[35]

7.3.4 Architecture of Bi-LSTM

A common LSTM unit is composed of five main components:-

1. Forget Gate
2. Input Gate
3. Cell State
4. Output Gate
5. Hidden State Output

These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions.[31] and remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.[32]

Review of Recurrent Neural Networks

To understand how LSTM's works, let's review the recurrent neural network. An RNN works like this; First words get transformed into machine-readable vectors. Then the RNN processes the sequence of vectors one by one.[31]

While processing, it passes the previous hidden state to the next step of the sequence. The hidden state acts as the neural networks memory. It holds information on previous data the network has seen before.

How to calculate the hidden state : First, the input and previous hidden state are combined to form a vector. That vector now has information on the current input and previous inputs. The vector goes through the tanh activation, and the output is the new hidden state, or the memory of the network.[31]

Tanh activation

The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1.

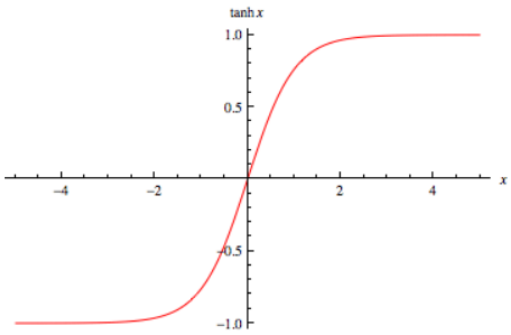


Figure 7.19: Tanh Function

When vectors are flowing through a neural network, it undergoes many transformations due to various math operations. So imagine a value that continues to be multiplied by let's say 3. Some values can explode and become astronomical, causing other values to seem insignificant. A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network.[31]

So that's an RNN. It has very few operations internally but works pretty well given the right circumstances (like short sequences). RNN's uses a lot less computational resources than it's evolved variants like **Long Short Term Memory Networks (LSTM)** and **Gated Recurrent Unit Networks (GRU)**.

In this report we will continue with LSTM.

LSTM

An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells.

These operations are used to allow the LSTM to keep or forget information.

Sigmoid

Gates contains sigmoid activations. A sigmoid activation is similar to the tanh activation. Instead of squishing values between -1 and 1, it squishes values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappears or be "forgotten." Any number multiplied by 1 is the same value therefore that value stay's the same or is "kept." The network can learn which data is not important therefore can be forgotten or which data is important to keep.

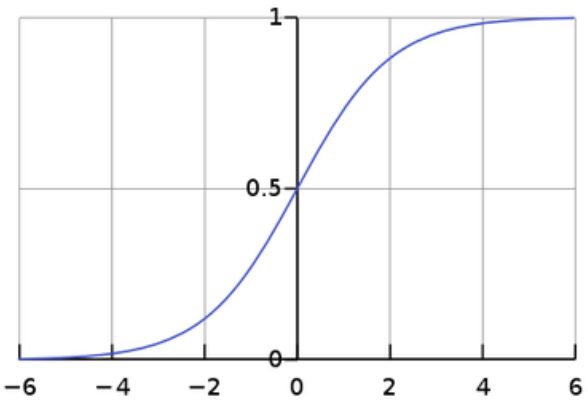


Figure 7.20: Sigmoid Function

We have three different gates that regulate information flow in an LSTM cell. A forget gate, input gate, and output gate.

1. Forget gate

First, we have the forget gate. This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

The forget gate controls the extent to which a value remains in the cell.

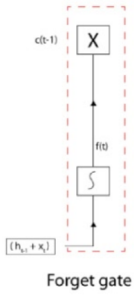


Figure 7.21: Structure of the Forget Gate of LSTM

2. Input Gate

To update the cell state, we have the input gate. First, we pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. You also pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network. Then you multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output.

The input gate controls the extent to which a new value flows into the cell.

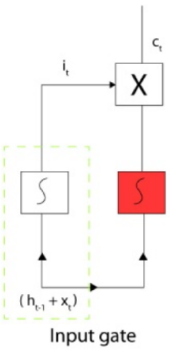


Figure 7.22: Structure of the Input Gate of LSTM

3. Cell State

Now we should have enough information to calculate the cell state. First, the cell state gets pointwise multiplied by the forget vector. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives us our new cell state.

Figure 7.23: Structure of the Cell State of LSTM

4. Output Gate Last we have the output gate. The output gate decides what the next hidden state should be. Remember that the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

The output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

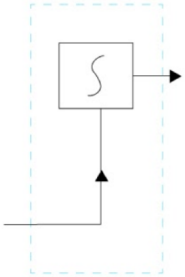


Figure 7.24: Processing of an Output Gate of LSTM

To review,

- The Forget gate decides what is relevant to keep from prior steps.
- The input gate decides what information is relevant to add from the current step.
- The output gate determines what the next hidden state should be.[31]

5. Hidden state output

Hidden state output is the multiplication of cell state information and output information.

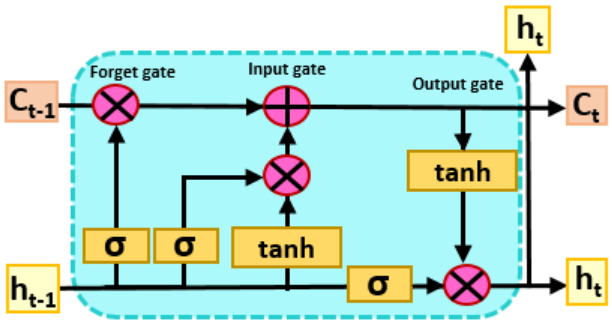


Figure 7.25: BiLSTM architecture of gates

The principle of BRNN is to split the neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). Those two states' output are not connected to inputs of the opposite direction states. The general structure of RNN and BRNN can be depicted in the right diagram. By using two time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information.

The idea behind Bidirectional Recurrent Neural Networks (RNNs) is very straightforward. Which involves replicating the first recurrent layer in the network then providing the input sequence as it is as input to the first layer and providing a reversed copy of the input sequence to the replicated layer. This overcomes the limitations of a traditional RNN.Bidirectional recurrent neural network (BRNN) can be trained using all available input info in the past and future of a particular time-step.Split of state neurons in regular RNN is responsible for the forward states (positive time direction) and a part for the backward states (negative time direction).

7.3.5 Working Of Bi-LSTM Network

Consider the phrase, 'He said, "Teddy ___". From these three opening words it's difficult to conclude if the sentence is about Teddy bears or Teddy Roosevelt. This is because the context that clarifies Teddy comes later. RNNs (including GRUs and LSTMs) are able to obtain the context only in one direction, from the preceding words. They're unable to look ahead into future words.

Bidirectional RNNs solve this problem by processing the sequence in both directions. Typically, two separate RNNs are used: one for forward direction and one for reverse direction. This results in a hidden state from each RNN, which are usually concatenated to form a single hidden state.

The final hidden state goes to a decoder, such as a fully connected network followed by softmax. Depending on the design of the neural network, the output from a BRNN can either be the complete sequence of hidden states or the state from the last time step. If a single hidden state is given to the decoder, it comes from the last states of

each RNN. [75]

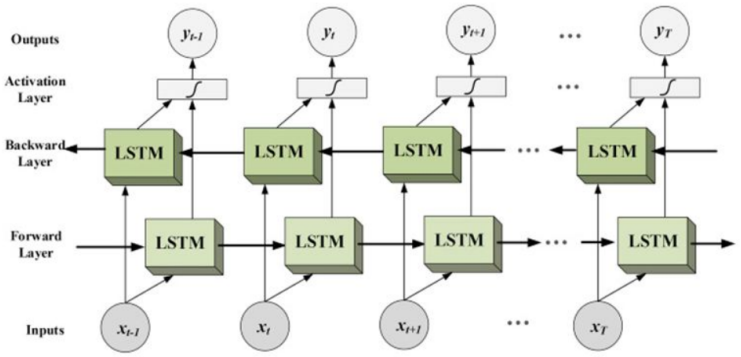


Figure 7.26: Working of Bi-LSTM Network

7.3.6 Applications of BiLSTM

Applications of BiLSTM include:

- Robot control
- Time series prediction
- Speech recognition
- Rhythm learning
- Music composition
- Grammar learning
- Handwriting recognition
- Human action recognition
- Sign language translation
- Protein homology detection
- Predicting subcellular localization of proteins
- Time series anomaly detection
- Several prediction tasks in the area of business process management
- Prediction in medical care pathways
- Semantic parsing
- Object co-segmentation
- Airport passenger management
- Short-term traffic forecast

7.3.7 Limitations of Bi-LSTM

1. One limitation with BRNN is that the entire sequence must be available before we can make predictions. For some applications such as real-time speech recognition, the entire utterance may not be available and BRNN may not be adequate.[37]

2. In the case of language models, the task is to predict the next word given preceding words. BRNN is clearly not suitable since it expects future words as well. Applying BRNN in this application will give poor accuracy. Moreover, BRNN is slower than RNN since results of the forward pass must be available for the backward pass to proceed. Gradients will therefore have a long dependency chain.

3. LSTMs capture long-term dependencies better than RNN and also solve the exploding/vanishing gradient problem. However, stacking many layers of BiLSTM creates the vanishing gradient problem. Deep neural networks so successful with CNNs are not so successful with BiLSTMs.

7.4 Siamese Neural Networks (SNNs)

Siamese networks are neural networks containing two or more identical subnetwork components.

Siamese nets were first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem [38]. Siamese networks are neural networks containing two identical subnetwork components, which accept distinct inputs but are joined by an energy function at the output.

A siamese network may look like this:

Siamese networks are neural networks containing two or more identical subnetwork components.

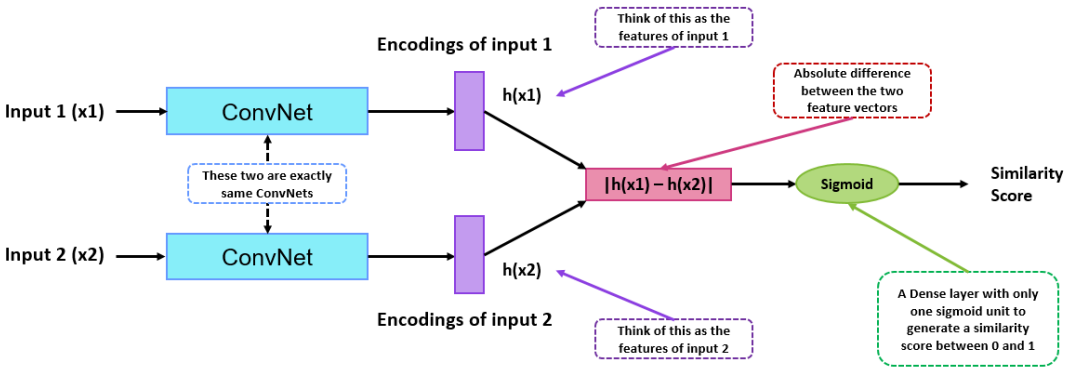


Figure 7.27: Siamese Neural Network

It is important that not only the architecture of the subnetworks is identical, but the weights have to be shared among them as well for the network to be called "siamese",

[39] which reduce the number of parameters for training as well as the tendency of overfitting.[40]

The main idea behind siamese networks is that they can learn useful data descriptors that can be further used to compare between the inputs of the respective subnetworks. Hereby, inputs can be anything from numerical data (in this case the subnetworks are usually formed by fully-connected layers), image data (with CNNs as subnetworks) or even sequential data such as sentences or time signals (with RNNs as subnetworks).[39]

7.4.1 Working of SNNs

Usually, siamese networks perform binary classification at the output, classifying if the inputs are of the same class or not. Hereby, different loss functions may be used during training. One of the most popular loss functions is the binary cross-entropy loss. This loss can be calculated as

$$L = -y \log p + (1 - y) \log(1 - p)$$

where L is the loss function, y the class label (0 or 1) and p is the prediction. In order to train the network to distinguish between similar and dissimilar objects, we may feed it one positive and one negative example at a time and add up the losses:

$$L = L_+ + L_-$$

Another possibility is to use the triplet loss: [41]

$$L = \max(d(a, p) - d(a, n) + m, 0)$$

Hereby, d is a distance function (e.g. the L2 loss), a is a sample of the dataset, p is a random positive sample and n is a negative sample. m is an arbitrary margin and is used to further the separation between the positive and negative scores.

7.4.2 Advantages of SNNs

- Sharing weights across subnetworks means fewer parameters to train.
- This means less data required and less tendency to overfit.
- It can be more robust to extreme class imbalance.
- It can yield better embeddings.
- Siamese network makes easy to use similar model to process similar kind of inputs.

7.4.3 Applications of SNNs

1. **One-shot learning :**

   In this learning scenario, a new training dataset is presented to the trained (classification) network, with only one sample per class. Afterwards, the classification performance on this new dataset is tested on a separate testing dataset. As siamese networks first learn discriminative features for a large specific dataset, they can be used to generalize this knowledge to entirely new classes and distributions as well.[38]

2. **Pedestrian tracking for video surveillance :**

   In this work, a siamese CNN network is combined with size and position features of image patches to track multiple persons in the field-of-view of the camera by detecting their position in each video frame, learning the associations between multiple frames and computing the trajectories.[42]

3. **Cosegmentation :**

   Cosegmentation is posed as a clustering problem to align the similar objects using Siamese network and segmenting them. We also train the Siamese network on non-target classes with no to little fine-tuning and test the generalization capability to target classes. Generation of visual summary of similar images based on relative relevance.[43]

4. **Matching resumes to jobs :**

   In this exotic application, the network tries to find matching job postings for applicants. In order to do this, a trained siamese CNN network extracts deep contextual information from both the postings and the resumes and computes their semantic similarity. The hypothesis is that matching resume — posting pairs will rank higher on the similarity scale than non-matching ones.[44]

Chapter 8

Dataset Analysis

8.1 Dataset Description

For this project we have used the publically available Quora Question Pairs data set on Kaggle [45][46]. For our experiment we used training data set available on the Kaggle website. This training data set for competition contains 404290 question pairs of size 60.4mb, while the test data set has 2345796 question pairs of size 455mb, which is about 6 times of the training data set. Each instance of the dataset consists of the ID of the question pair, the unique IDs of both the questions and full text of each question, as well as the target variable indicating whether the two questions are duplicate or not (i.e. have the same meaning) [40].

Columns	Description
id	unique value for the question pair
qid1	unique value for the first question pair
qid2	unique value for the second question pair
question1	full text of the first question
question2	full text of the second question
is_duplicate	If question pair is duplicate then 1, otherwise 0

Table 8.1: Description

The question pairs of the dataset cover myriad of topics such as education, technology, sports, philosophy, politics, culture, economics, social, entertainment and many more. Some questions also include special characters such as mathematical symbols and foreign language characters.

We have split our data into two sets of training and validation in ratio 70:30, training set has 283003 instances whereas validation set has 121287 in which 44937 are duplicate and 76350 are not duplicate entries.

It is worth mentioning that the duplicate labels in the training set are provided by human experts and are inherently subjective, since the true meaning of sentences can never be known with certainty. As a result, the labels on the training data set represent a reasonable consensus and are considered ground truth, but are not 100% accurate, i.e. the data set might be noisy [45].

8.2 Dataset Observations

In the training data set given, about 20.7% of questions appear multiple times in different question pairs. By simple analysis we found that duplicate question pairs have only 36.92% of whole training data set with 149263 entries and rest are non duplicates of entries 255027, indicating that the data set is heavily imbalanced with many more non-duplicated question pairs than the duplicated ones.

The first observation is that the training data set is noisy, i.e. Although the two questions share almost all the words and have the exact same meaning, they are labelled as non-duplicated. For instance, the question pair consisting of sentences "When can I expect my Cognizant confirmation mail?" and "When can I expect Cognizant confirmation mail?" is labelled as non-duplicated, despite the fact that the two questions only differ in an insignificant stop word "my".

The second observation concerns with the first word of the questions. The most common first words in the dataset (Which, How, What, Why, Is, I, Who and Can) are mostly interrogative, as expected. The first words of questions are not much significant in the detection of duplicates, since the percentage of question pairs where the two questions share the same first word for duplicated and non-duplicated classes are 62.3% and 41.9% respectively, which is not a large difference.

The third observation is that there is no significant role of Question Mark symbol "?" on semantics of the questions pairs, for an instance "How do I make friends." and "How to make friends?" is labeled as duplicate. Although one question has "?" symbol and other doesn't.

The fourth observation is that the order of words matter a lot in the identification of duplicate questions, which means simple methods involving only the word-to-word matching between sentences and neglecting the higher level abstractions will fail to capture the question semantics and falsely label some question pairs. For example, consider the question pair "what do you think about your parents?", "What do your parents think about you?"], here the two questions consist of the exact same words but they are labelled as non-duplicate because of the different order of words resulting in different semantics.

Chapter 9

Implementation with CNN Model

In this section, different basic and ensemble machine learning algorithms are trained to predict if the question pair set in the dataset is duplicate or non-duplicate of each other.
Here we describes the general setup of the framework that we have implemented with the goal of advancing the study of duplicate question pair detection. This is our first approach where we have used CNN model.

9.1 Dataset Preprocessing

Preprocessing begins with reading the text for question 1 and question 2 from the file, along with the label (1 for duplicate, 0 for non-duplicate).

There are different ways in which text can be preprocessed before being made fit into a model.

We started with dataset cleansing by filtering all the special characters
(such as '! "# $ % & \'() ∗ + , - · / : ; <=>? @ [//] ∧ _{|} ∼′
in question1 and question2 columns as they do not contribute to the context of a sentence. For example, some simple counts on Quora dataset showed us that 99.87% of questions contain the question mark symbol, while other symbols such as full stop appear 6.39% and a small number of questions (less than 1%) contain math and programming symbols such as , = etc.
For example, the relationship between the duplicates in Table 5.4 becomes ambiguous after removing stop words.

For example, the relationship between the duplicates in Table 5.4 becomes ambiguous after removing stop words.

We choose not to remove stop words as this might alter the meaning of a sentence.

Lemmatization is the process of returning a word to its base or dictionary form, while stemming removes the suffix of a word. As we want to retain as much as possible of the initial meaning of a sentence, we do not apply neither of these techniques to our datasets.

S.No.	Original	Stop words removed
Q1	Why do people not like Donald Trump running for president?	Why, people, like, Donald, Trump, running, president, ?
Q2	Why are people so against Donald Trump running for president?	Why, people, Donald, Trump, running, presdent, ?

Table 9.1: Examples of sentences losing initial meaning after stop word removal.

Then we have converted all the texts into lower case.

After the cleaning step, each sentence goes through a tokenization process, in which it is split in individual words based on blank spaces. We create a dictionary (also known as word index) of 89,983 unique words. In this dictionary (also called corpus) , each unique word is assigned an index. This dictionary is used to replace every word in a sentence with its index, thus obtaining word vectors. For example, the question becomes the vector:

Input : What is the step by step guide to invest in share market in india?

Output : [2, 3, 1, 1222, 59, 1222, 2566, 7, 579, 8, 763, 384, 8, 36]

Then we move towards encoding of sentences using text_to_sequences() function. The point to be kept in mind is that the encoding of the two columns of questions in the dataset must be done separately.

The embedding layer requires that all vectors have the same length, which is initially not the case as sentences naturally vary in length.
To achieve this , we apply padding to both the columns separately with maximum word size of 100. This is done to convert all the questions having variable length into fix length.

For instance,

Input : What's the best on-line calender/scheduling/booking system? Something that allows me to show my availability customers can 'request' to meet me. When I 'accept' their request, the appointment pops into our calendars & reminders are automated.

Output : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 26 1 19 20 784 22234 9721 3068 211 404 29 3665 62 7 442 17 11647 2965 13 1967 7 820 62 39 4 1300 79 1967 1 4475 14553 108 214 15925 17649 11 6685]

The main purpose of these steps is to remove noise i.e. the elements that do not add meaning to sentences.

GloVe
For a deep learning model to be able to learn from text data we need to create word embedding matrix. In order to obtain this matrix we use the GloVe.
The embedding matrix, which will be passed to the embedding layer, requires two

parameters: the number of unique words it should contain and the dimension of the vectors.

There are 89,983 unique words in our dataset. So we add 89,983 words to the embedding matrix and the dimension of each word vector will be 300.

It assigns each word in the corpus with a unique vector of numbers having fixed dimension (300d in our model). These resulting vectors are passed as input to the neural network.

9.2 Proposed Approach

The Deep models do not understand input in the form of text or speech. In order to make the input understandable for such models, every question must be vectorized. In our proposed model, first layer is embedding layer that accepts the question pairs as input and converts each word into a vector.

We begin by passing each question into a separate tower of the network with identical parameters and weights, producing a "siamese network". (Refer to Section 7.4)

The raw questions, represented as single dimensional vectors of vocabulary indexes, are then converted into pre-trained word in the embedding layer. The embedding matrix for each question is then passed through the Four Convolutional neural layers followed by Pooling and Dropout layers, that converts the word matrix into a single-dimensional feature vector.
These feature vectors are then passed through a Flatten Layer that transform all the features into a vector that can be fed into a fully connected layer i.e Dense layer.

The output of this dense layer from each of the two independent networks are concatenated together using Vector difference and Hadamard Multiplication. Finally, the concatenated feature vectors are passed through a fully connected Dense Layer that produces the final output.

We use cross entropy loss function and the model is trained using the Adam optimizer. The implementation1 of the model is done in python language using TensorFlow.

The training is performed using Google Colaboratory, by setting runtime environment to GPU.

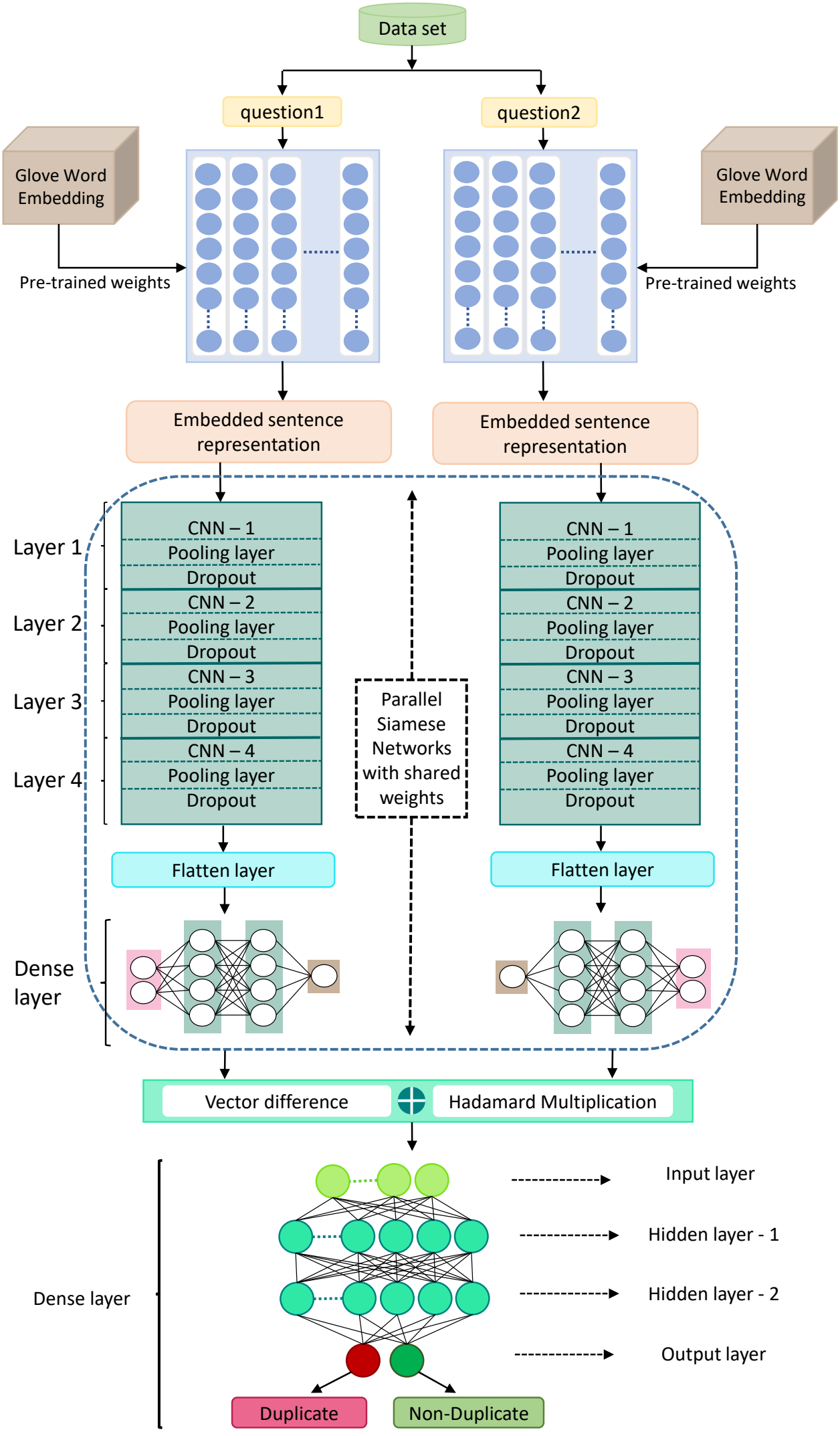


Figure : Architecture - 1, CNN Model with 4 layers

9.3 Results

We have created four different CNN models having 1, 2, 3 and 4 CNN layers where each CNN Model is trained on four different batch sizes : 32, 64, 128 and 256. Hence in total we implemented 16 different CNN models with their accuracy, log loss and f1 score as shown below :

S.No	Batch Size	Accuracy	f1-0	f1-1	Log loss
1	256	77.76	0.82	0.52	0.501
2	128	79.01	0.83	0.57	0.484
3	64	79.32	0.83	0.54	0.493
4	32	79.72	0.83	0.57	0.482

Table 9.2: Accuracy result for CNN 1 layer

S.No	Batch Size	Accuracy	f1-0	f1-1	Log loss
1	256	78.53	0.83	0.58	0.491
2	128	79.25	0.83	0.59	0.482
3	64	79.66	0.84	0.61	0.475
4	**32**	**80.41**	**0.84**	**0.65**	**0.463**

Table 9.3: Accuracy result for CNN 2 layers

S.No	Batch Size	Accuracy	f1-0	f1-1	Log loss
1	256	78.65	0.83	0.61	0.488
2	128	78.81	0.83	0.62	0.486
3	64	79.45	0.84	0.62	0.483
4	32	80.79	0.84	0.66	0.476

Table 9.4: Accuracy result for CNN 3 layers

S.No	Batch Size	Accuracy	f1-0	f1-1	Log loss
1	256	77.94	0.83	0.61	0.516
2	128	78.56	0.83	0.59	0.509
3	64	79.65	0.84	0.67	0.494
4	32	80.31	0.83	0.67	0.486

Table 9.5: Accuracy result for CNN 4 layers

From the above four tables representing accuracy results of 16 CNN models, we infer that the least log-loss value is achieved with CNN 2 layer model having 32 batch size. The log loss value is 0.463.
We have plotted the 3 graphs :

- Accuracy v/s Epoch
- Loss v/s Epoch
- Area under curve-ROC

These three graphs for the best CNN model we got are shown below :

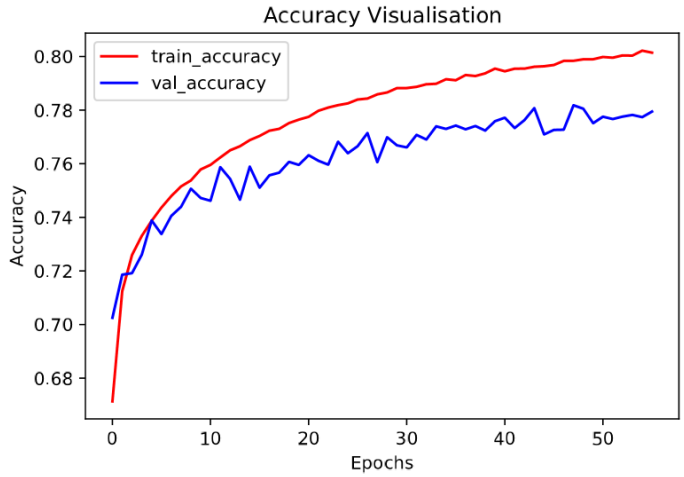


Figure 9.1: Accuracy v/s Epoch Graph of CNN Model

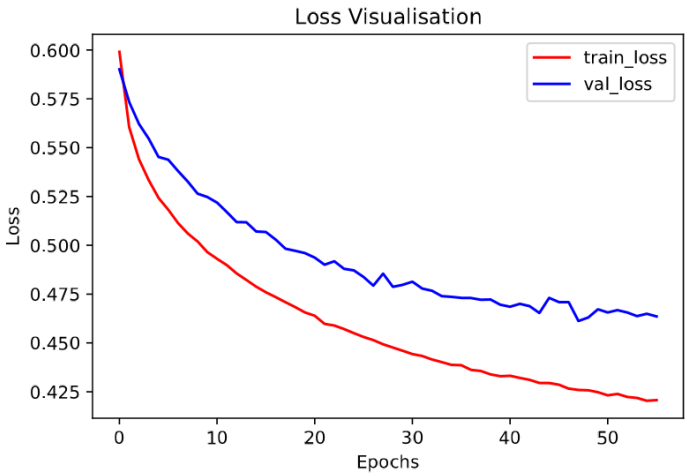


Figure 9.2: Loss v/s Epoch Graph of CNN Model

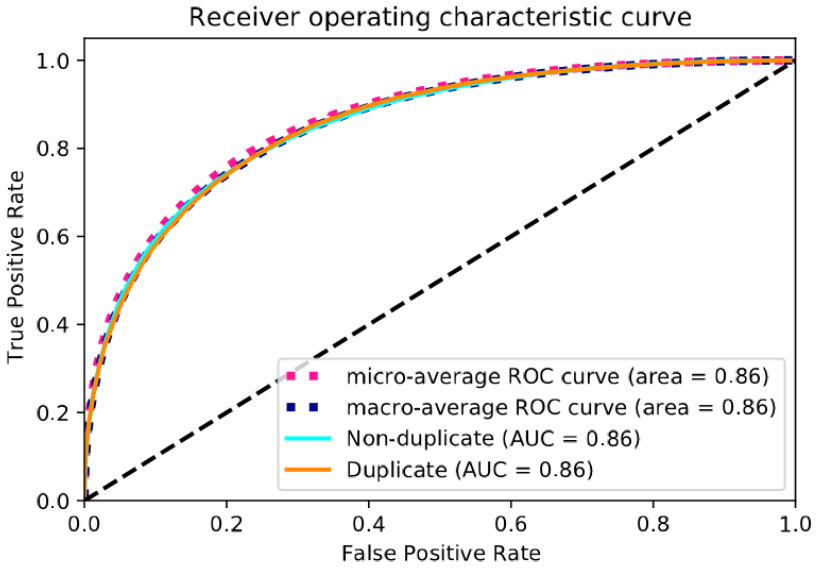


Figure 9.3: Area under curve-ROC Graph of CNN Model

Chapter 10

Implementation with BiLSTM Model

In the general setup of the framework that we have implemented with the goal of advancing the study of duplicate question pair detection, this is our second approach where we have used BiLSTM moldel.

10.1 Dataset Preprocessing

For Dataset Preprocessing , please refer to Section 9.1 as the preprocessing of the dataset is same for all the proposed models.

10.2 Proposed Approach

Aside from the simple word-level and sentence-level metrics for evaluating the dissimilarity between two questions, we want to utilize the power of neural networks in modeling the data, given the large number of data entries we are presented with. Recurrent neural networks (RNNs) are rather effective at modeling sequential data, in that they have a memory storage that captures the previous inputs as well as the corresponding computations, and allows information to cycle inside the networks for arbitrarily long time (hence the name recurrent) [12]. However, RNN has its restrictions in learning long-term dependencies, and decreases in performance with the increasing length of the input sequence.

Given the limitations of RNN, a Long-Short Term Memory (LSTM) structure is widely used in sequence modeling, which is capable of learning long-term dependencies using four interacting layers, as opposed to the one information processing layer in the RNN structure [13]. In addition to the better performance in modeling long-term dependencies, LSTM provides more flexibility in terms of controlling the amount of stored information as needed, which makes it an ideal choice for our purpose of sequence modeling.

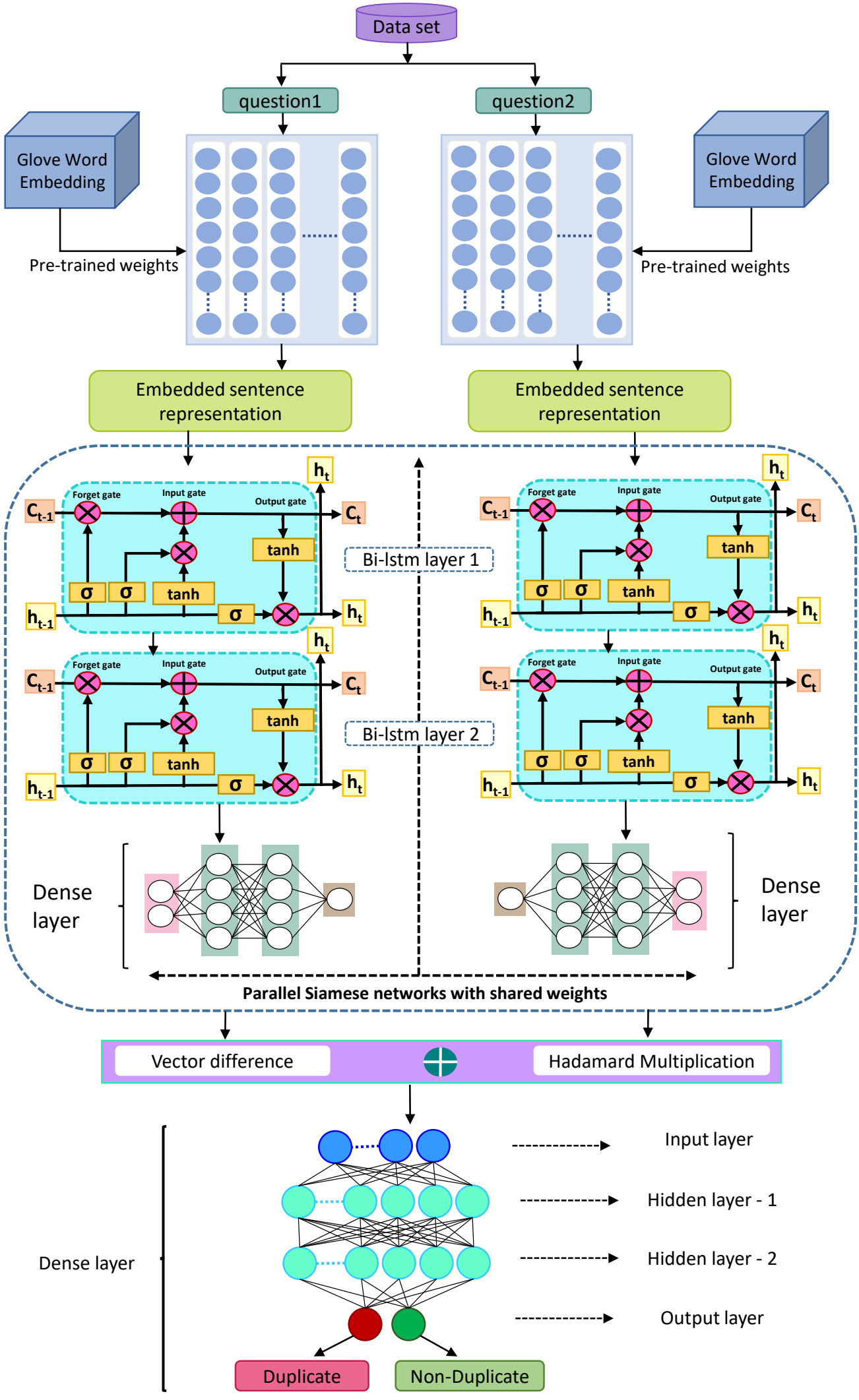


Figure : Architecture - 2, LSTM Model with 2 layers

We design a siamese structure that consists of two LSTM sub-networks (one for each question in the question pair. We begin by passing each question into a separate tower of the network with identical parameters and weights, producing a "siamese network". (Refer to Section 7.4)

Siamese neural network is an architecture that contains two identical sub-networks joined at the outputs [6], and is widely used in tasks that involve finding the similarity between two comparable patterns, e.g. paraphrase identification. A key feature of the siamese structure is the shared weights across the sub-networks, which reduce the number of parameters for training as well as the tendency of overfitting.

In addition, by processing similar inputs with similar models, the subnetworks generate input representations that share the same semantics and are easy to compare.

The Siamese LSTM model is first trained on each word embedding (GloVe) individually and later we use the blend of these trained models prediction for final prediction. The models are trained on 283K number of samples and are tested on 121k instances (validation dataset).

The training is performed using Google Colaboratory, by setting runtime environment to GPU.

We have passed our embedded sentence into the LSTM layer. The LSTM layer contains 3 gates, **Forget** Gate , **Input** gate, **Output** gate :

- The Forget gate decides what is relevant to keep from prior steps.
- The Input gate decides what information is relevant to add from the current step.
- The Output gate determines what the next hidden state should be.

After that, the output from LSTM layer is fed into the fully connected Dense Layer. The outputs from both the parallel Dense layers are further concatenated using 'Vector Difference' and 'Hadamard Multiplication'. This concatenation layer merges the outputs of the two sub-networks into a single vector representation for the final classification.
Finally, the concatenated vectors are again passed through a fully connected Dense Layer which outputs a single value indicating the probability that the two question are duplicates.

10.3 Result

We have created two different BiLSTM models having 1 and 2 LSTM layers where each BiLSTM Model is trained on four different batch sizes : 32, 64, 128 and 256. Thus in total we implemented 8 different BiLSTM models with their accuracy, log loss and f1 scores as shown below :

S.No	Batch Size	Accuracy(%)	f1-0	f1-1	Log loss
1	256	78.76	0.83	0.68	0.465
2	128	79.51	0.84	0.66	0.475
3	64	80.24	0.84	0.68	0.464
4	32	79.97	0.84	0.68	0.468

Table 10.1: Accuracy results for BiLSTM 1 layer

S.No	Batch Size	Accuracy(%)	f1-0	f1-1	Log loss
1	256	79.28	0.84	0.64	0.468
2	128	79.29	0.83	0.64	0.469
3	64	80.11	0.84	0.64	0.472
4	**32**	**80.59**	**0.84**	**0.66**	**0.463**

Table 10.2: Accuracy results for BiLSTM 2 layer

From the above 8 tables representing accuracy results of 8 BiLSTM models, we infer that the least log-loss value is achieved with BiLSTM 2 layer model having 32 batch size. The log loss value is 0.463 .

We have plotted the 3 graphs :

- Accuracy v/s Epoch
- Loss v/s Epoch
- Area under curve-ROC

These three graphs for the best BiLSTM model we got is shown below :

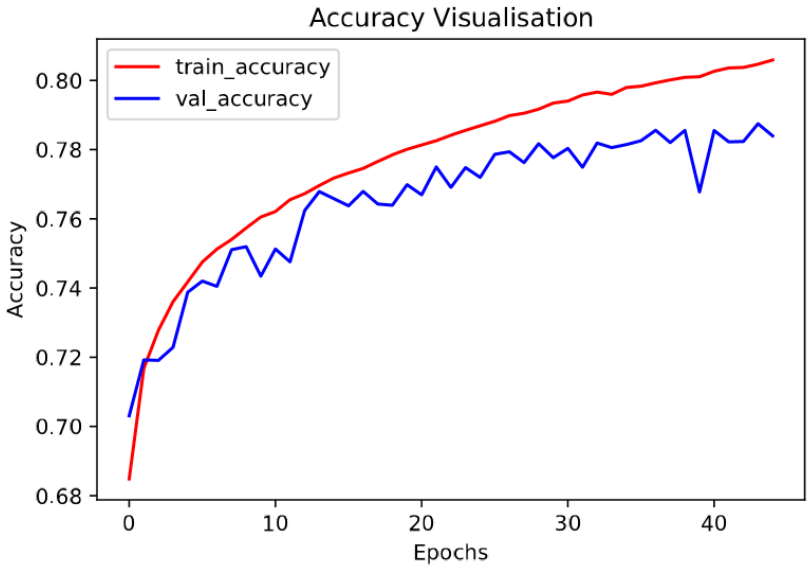


Figure 10.1: Accuracy v/s Epoch Graph of BiLSTM Model

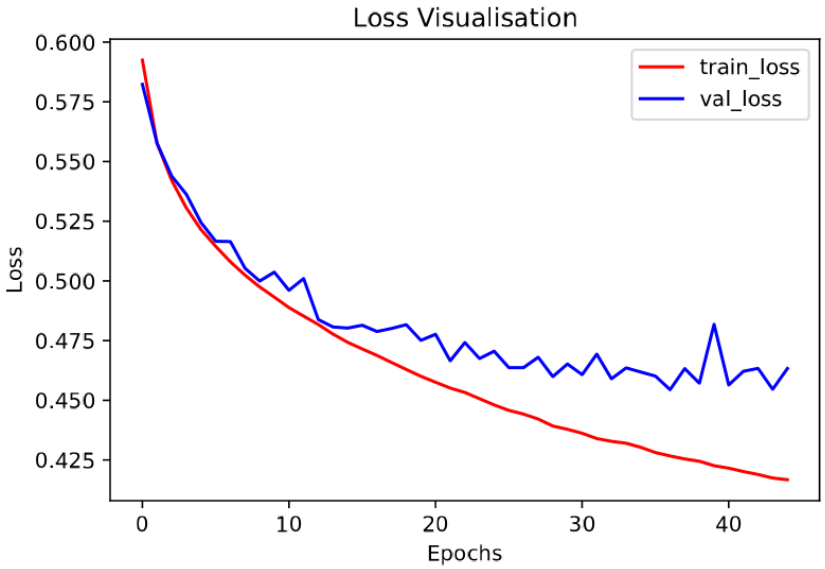


Figure 10.2: Loss v/s Epoch Graph of BiLSTM Model

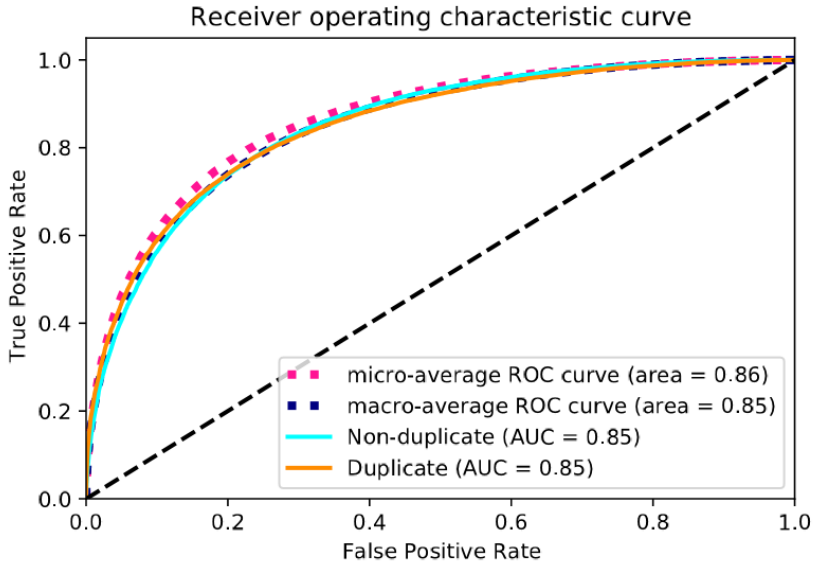


Figure 10.3: Area under curve-ROC Graph of BiLSTM Model

Chapter 11

Implementation of BiLSTM with Common Words Feature

11.1 Dataset Preprocessing

For Dataset Preprocessing , please refer to Section 9.1 as the preprocessing of the dataset is same for all the proposed models.

11.2 Proposed Approach

We have seen the implementation of CNN and BiLSTM model for the Duplicate Question Pair Dataset. Now we move on to dive into the last model which is based on BiLSTM with a slightly different additional feature.

In this simple neural network architecture, we use a pair of questions as the two inputs.
Embedding layers is the first hidden layer of this network that uses word embedding to represent a word as a dense vector, and we specify three arguments to the Embedding function, the input dimension, output dimension, and the input length as vocab_size+1 , 300 and 100 respectively.

The architecture consists of the "Embedding Layer", "LSTM Layer" applied separately on each of the question inputs in a parallel "Siamese Network".(Refer to Section 7.4) Additionally, we extract a feature of "Common words between question pair".

Now, the output of each LSTM layer and the "Common Words Feature" are separately passed through a "Dense Layer" in a parallel fashion as shown in Architecture -3

Then all the three outputs from Dense Layer are merged using the Concatenated layer.

The output from the merged model layer is then passed through the series of "Batch Normalization"(speeds up and optimizes training), "Dropout" (which prevents the overfitting of model), and Dense Layer. We used the dropout weight of 0.17 within LSTM.

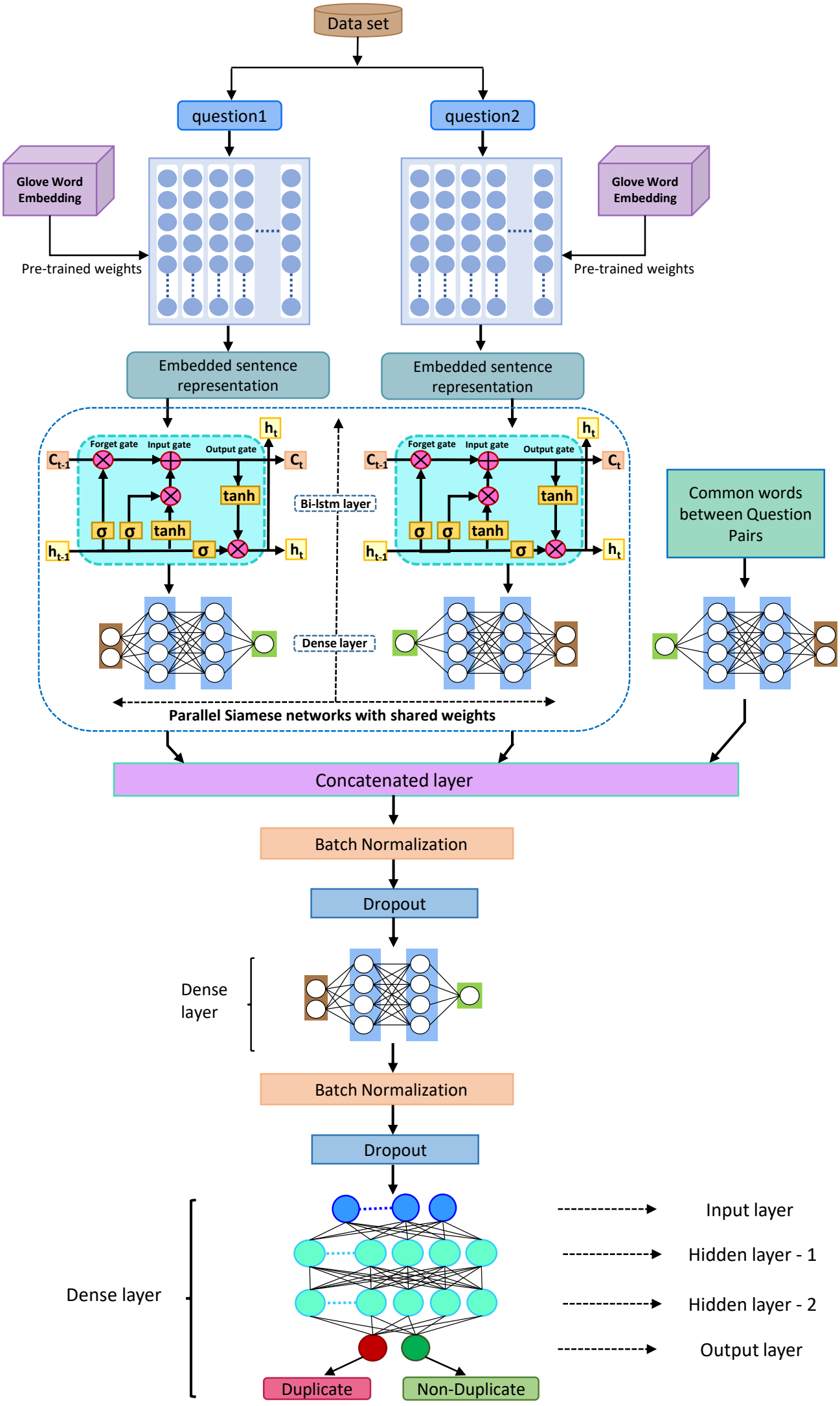


Figure : Architecture - 3, LSTM Model with Common Words feature

We pass the output again through a series of Batch Normalization and Dropout.

The output of Dropout layer is passed through the fully connected dense Layer to get final classification. Sigmoid Activation function is applied at the final output layer.

11.3 Results

Below table shows Accuracy , log loss and f1 score of the BiLSTM with Common Words model. We set the batch size to 1024.

S.No.	Batch Size	Accuracy(%)	f1-0	f1-1	Log loss
1	**1024**	**85.11**	**0.86**	**0.78**	**0.368**

Table 11.1: Accuracy result for BiLSTM with Common Words Feature

From the table representing accuracy results, we infer that the log loss value is 0.368 which is by far the least. Hence we conclude that BiLSTM with common word feature performs the best among all the models experimented by far.

We have plotted the 3 graphs :

- Accuracy v/s Epoch
- Loss v/s Epoch
- Area under curve-ROC

These three graphs for the model we got are shown below :

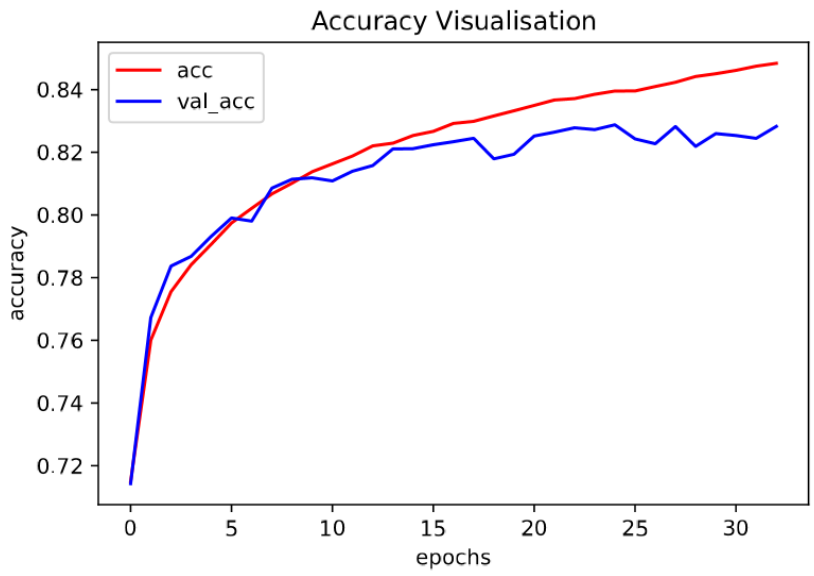


Figure 11.1: Accuracy v/s Epoch Graph of BiLSTM with Common Words Feature Model

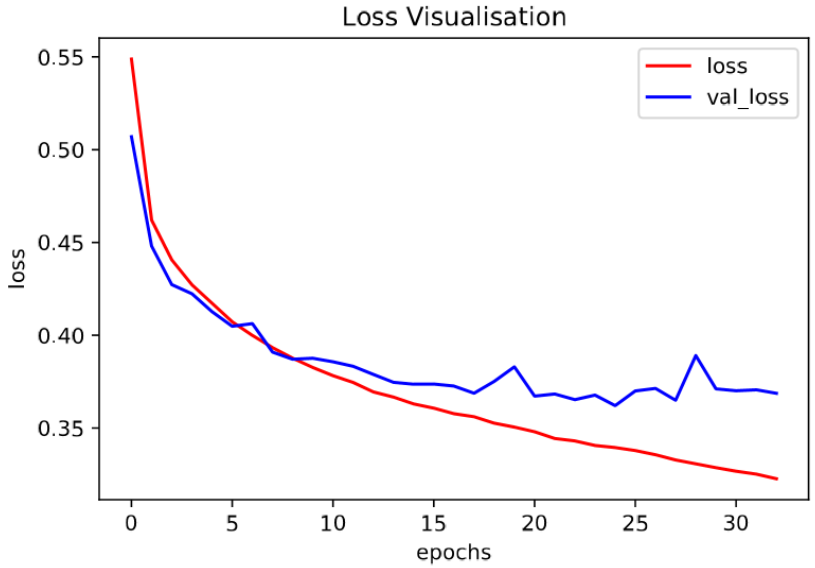


Figure 11.2: Loss v/s Epoch Graph of BiLSTM with Common Words Feature Model Model

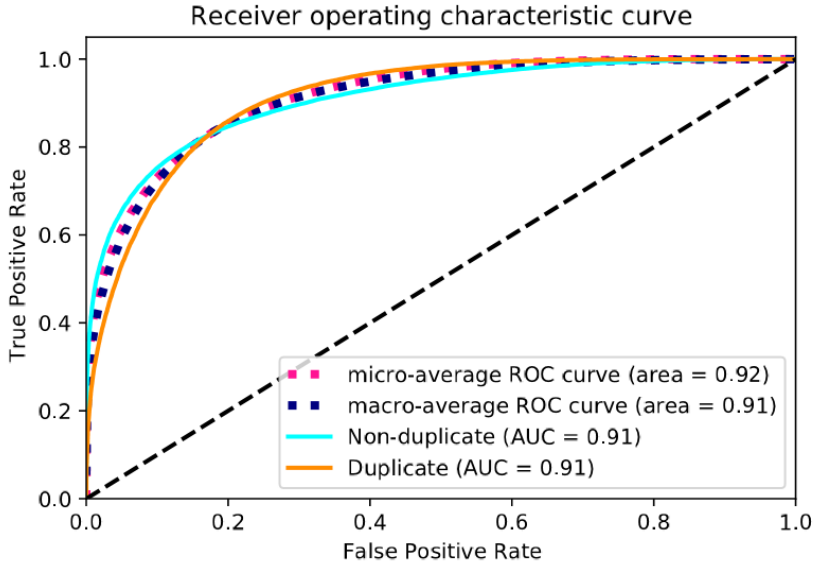


Figure 11.3: Area under curve-ROC Graph of BiLSTM with Common Words Feature Model Model

Chapter 12

Comparison among Proposed Models

In this section we compare the results of all three models we proposed and discuss our main findings from each experiment.

12.1 Discussion

Evaluation Metrics

The selection of metrics is the most crucial step in the evaluation of our models as it influences how we measure the performance of our model against each other and the baselines.

1. **Accuracy** : Accuracy is the ratio of the total number of correct predictions made by the models to the total number of predictions requested to the model.
2. **F1-Score** : F1-score or F1-measure is harmonic mean of precision and recall. To understand F1-Score, we need to understand Precision, also known as Specificity and Recall, also known as Sensitivity.
3. **Precision** : Precision or Specificity is the ratio of predicted positive samples that are actually positive to the total number of positive predictions made by the models.
4. **Recall** : Recall or Sensitivity is the ratio of predicted positive samples that are actually positive to the total number of actual positive predictions in total sample.
5. **Log loss** : Log loss is also known as cross-entropy, and when the classification type is of binary as in our research, then it is known as binary cross-entropy. Log Loss value lies in the range of 0,1 where ideal models will have log loss of 0, and the worst model will have log loss of 1. Log loss indicates how badly our model predicted the probability of our classification.

We only include results of best performing model of all 3 proposed models.

In case of CNN model, we found that the CNN with 2 layers having batch size 32 is the best performing model with training accuracy 80.41% and log loss value of 0.463 .

Similarly in BiLSTM model, we found the BiLSTM with 2 layers having batch size 32 performed best with training accuracy 80.59% and log loss value of 0.463 .

Similarly last model of BiLSTM with Common Word Feature gives the best model with batch size 1024 with training accuracy 85.11% and log loss value of 0.368 .

12.2 Results

The Classification Report for all the three best performing models in each category are represented below. Classification report shows the values of 'precision' , 'recall' and 'f1 score' for respective models.

	precision	recall	f1-score	support
0	0.78	0.91	0.84	76350
1	0.79	0.55	0.65	44937
micro avg	0.78	0.78	0.78	121287
macro avg	0.78	0.73	0.74	121287
weighted avg	0.78	0.78	0.77	121287
samples avg	0.78	0.78	0.78	121287

Figure 12.1: Classification report of CNN Model

	precision	recall	f1-score	support
0	0.78	0.91	0.84	76350
1	0.78	0.58	0.66	44937
micro avg	0.78	0.78	0.78	121287
macro avg	0.78	0.74	0.75	121287
weighted avg	0.78	0.78	0.78	121287
samples avg	0.78	0.78	0.78	121287

Figure 12.2: Classification report of BiLSTM Model

	precision	recall	f1-score	support
0	0.89	0.83	0.86	76350
1	0.74	0.83	0.78	44937
micro avg	0.83	0.83	0.83	121287
macro avg	0.82	0.83	0.82	121287
weighted avg	0.83	0.83	0.83	121287
samples avg	0.83	0.83	0.83	121287

Figure 12.3: Classification report of BiLSTM with Common Words Feature Model

We have further compared the f1 score (f1-0 and f1-1) of all the three best performing models.

S.No.	Model	f1-0	f1-1
1	CNN	0.84	0.65
2	BiLSTM	0.84	0.66
3	BiLSTM with Common Words	**0.86**	**0.78**

Table 12.1: f1 score comparison among 3 Proposed Models

The graph plot for the f1-score comparison is shown below :

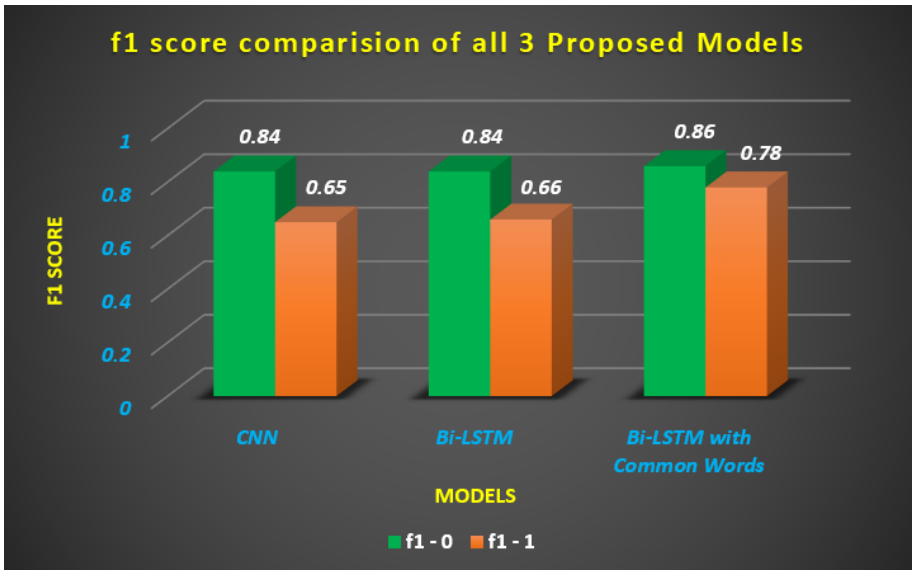


Figure 12.4: Graph of f1 score comparison for all 3 models

We also compared the accuracy and log loss values of all the three best performing models with the graph visualisation as shown below.

The graph plot for the accuracy comparison is shown below :

S.No.	Model	Accuracy(%)	log loss
1	CNN	80.41	0.463
2	BiLSTM	80.59	0.463
3	BiLSTM with Common Words	**84.84**	**0.368**

Table 12.2: Accuracy & log loss comparison among 3 proposed models

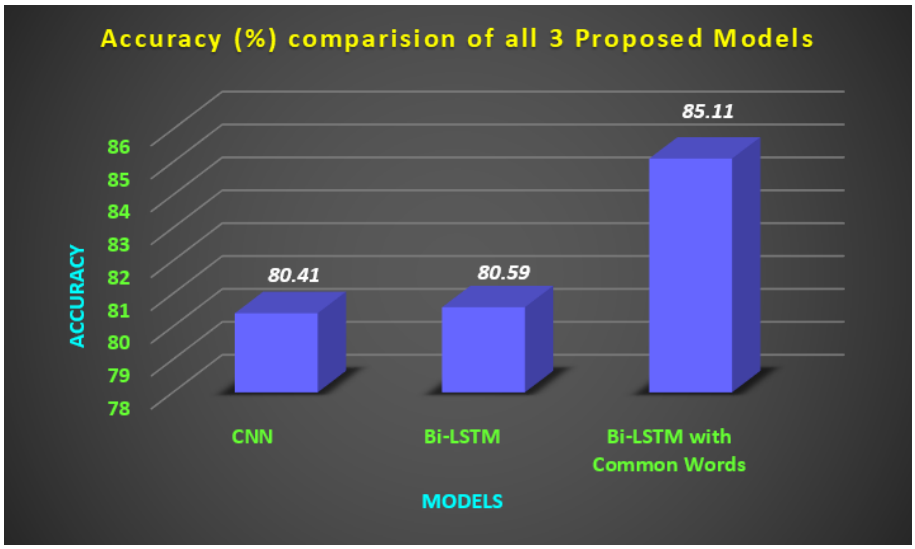


Figure 12.5: Graph of Accuracy comparison for all 3 models

The graph plot for the log loss comparison is shown below :

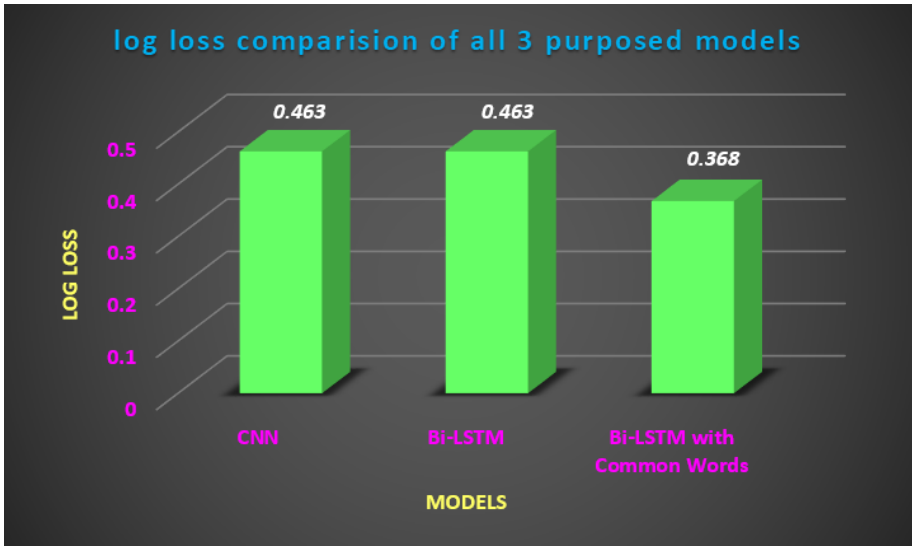


Figure 12.6: Graph of log loss comparison for all 3 models

We find that the 3rd model of BiLSTM with Common Words Feature outperformes the other two models giving accuracy of 85.11% without any data augmentation.

BiLSTM 2 layer model achieves an accuracy of 80.59% showing a slightly better performance in comparison to CNN 2 layer model with an accuracy of 80.41%.

In comparison, BiLSTM with Common Word feature achieves an accuracy which is almost 4.5% improvement over the previous best performance.

12.3 Findings

The general findings are as follows:

1. While training our model we have seen that if we increase batch size from 32 to 256, the graph becomes smooth (distortion decreases) but training accuracy decreases.

   When we decrease the batch size from 256 to 32, the graph gets spikes (distortion increases) but training accuracy increases.

2. We observed that in case of 3rd model when we add "Common Word Feature", the accuracy increases with good margin and hence log loss also decreases.

   We hence conclude that if we keep on improvising the model with more features, the accuracy will enhance further.

3. As we increase the number of layers in a model, the training time of the model also increases.

4. Also, as the batch size increases, the traning time decreases.

5. The size of the kernel is important and should be tuned for each problem.

Chapter 13

Implications

For future work, we see several possible extensions of our model. Additionally, we could spend more time designing more features for models.

We suspect that using word embeddings trained on Quora data could provide a boost in performance. Firstly, other researchers have demonstrated that weighting the word embeddings by their TF-IDF scores can improve performance on semantic textual similarity tasks. The basic idea is to give rare words greater weight. In addition there is room for improvement by using "matching aggregation" techniques, as proposed by Wang et al. , that allow interactions between the input sentences as they are encoded through the neural networks.
It could be obtained by weighing the common words by their TF-IDF (term-frequencyinverse-document-frequency) weight while measuring overlap.
Also, it could be fascinating to figure out how to accomplish multi-perspective matching by combining question features within a CNN layer.

One idea is to incorporate "Data augmentation". There are several ways of achieveing data augmentation that could be worked on such as :

1. Applying SMOTE in the BiLSTM model.
2. Using 'Reverse Question Augmentation' where we can reverse the order of question pairs and double the size of original dataset while maintaining the initial distribution of labels.
3. Thesaurus Augmentation where we find synonyms for words in a given sentence and replace them.
4. Heuristic Augmentation that includes sampling the available dataset based on a defined heuristic for word overlap and adding that sample again, but in reverse order.
5. Word embedding perturbation which is better called 'Gaussian Embedding Perturbation'. It addresses the continuous space of word embeddings, instead of the discrete space of words. This type of augmentation may make sentence encoding models more robust to noise.

We aim to find appropriate text representation, data augmentation, transfer learning and ensembling techniques to further advance the study of DQD.

In future we would like to extend our experiments to other models such as ABCNN [47] and ESIM [48] which have shown better performance than traditional NLP techniques.

Chapter 14

Conclusion

We have introduced the Quora Question Pairs for Duplicate Question Pair Detection from Kaggle competition. The adaption of machine learning system is found to be suitable for the optimization in the work. The model is built using Python language.

We elaborated Neural networks and their working. We introduced types of Neural Networks such as CNN, RNN, BiLSTM. Additionally, we discussed Siamese Neural Network in details.

We proposed CNN and BiLSTM models for our Duplicate Question Pair Detection (DQD) dataset. The implementation was based on parallel Siamese network. Apart from these two models, we experimented with another BiLSTM model adding a third feature of Common Words to it. This model outperformed all previous models and gave the best accuracy with least value of log loss which was 0.368.

Future effort is required in terms of exploring different approaches to improve our siamese LSTM model, as well as experimenting with the state-of-the-art models in paraphrase identification.

Overall, however, it's clear that deep learning is a powerful tool when applied to this problem that shows a lot of promise going forward.

Bibliography

[1] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[3] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[4] Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. Neural paraphrase identification of questions with noisy pretraining. *arXiv preprint arXiv:1704.04565*, 2017.

[5] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

[6] Sumit Shukla. Machine learning : https://rb.gy/0okprl. 2018.

[7] Priyadharshini. Machine learning working : https://rb.gy/skdpbw. 2020.

[8] Techvidvan Team. Advantages and disadvantages machine learning : https://rb.gy/s8yuze. 2020.

[9] Dataflair Team. Applications of machine learning : https://rb.gy/kqjdpf. 2018.

[10] Pratibha Roy. Applications of machine learning : https://rb.gy/1qbhry. 2019.

[11] Anaconda Documentation. Anaconda navigator : https://rb.gy/lipatj. 2020.

[12] Anaconda Documentation. Anaconda navigator : https://rb.gy/jxjmjq. 2020.

[13] Conda Documentation. Conda : https://conda.io/en/latest/. 2017.

[14] Mike Driscoll. Jupyter notebook : https://rb.gy/j0ixfi. 2020.

[15] Jupyter Team. Jupyter notebook : https://rb.gy/6dkqwf. 2015.

[16] Abhishek Sharma. Google colab : https://rb.gy/cxsz54. 2020.

[17] Tutorials Point. Google colab : https://rb.gy/fmffuw. 2020.

[18] Tutorials Point. Google colab conclusion : https://rb.gy/zncjsz. 2020.

[19] Larry Hardesty. Neural networks : https://rb.gy/cam8bj. 2017.

[20] Vibhor Nigam. Neural networks activation functions : https://rb.gy/5vxxq1. 2018.

[21] Vinod Sharma. Cost function : https://rb.gy/ccnuqu. 2018.

[22] guru99. Backpropagation : https://rb.gy/tt4xcg. 2020.

[23] Tutorials Points. Applications of neural networks : https://rb.gy/dimitb. 2020.

[24] Kechit Goyal. Cnn : https://www.upgrad.com/blog/convolutional-neural-networks/. 2020.

[25] Wikipedia. Cnn : https://en.wikipedia.org/wiki/convolutional$_n eural_n etwork$.2020.

[26] Sumit Saha. Structure of cnn : https://rb.gy/nxajwf. 2018.

[27] Jorge Leonel. Hyperparameters of cnn : https://rb.gy/fqy8a3. 2019.

[28] DENNY BRITZ. Stride of cnn : https://rb.gy/goumbj. 2015.

[29] vijay choubey. Architecture of cnn : https://rb.gy/7xoupv. 2020.

[30] aishwarya.27. Rnn : https://rb.gy/k1vr0f. 2020.

[31] Michael Phi. Limitations of rnn : https://rb.gy/blcrqo. 2018.

[32] Wikipedia. History of rnn : https://rb.gy/ebcwzb. 2020.

[33] Tanesh Balodi. Bilstm : https://rb.gy/a4o7mk. 2019.

[34] Jason Brownlee. https://rb.gy/ny3ia8. 2017.

[35] https://rb.gy/7nzyoq. 2020.

[36] https://rb.gy/usj8sj. 2019.

[37] arvindpdmn. https://devopedia.org/bidirectional-rnn.

[38] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[39] Branislav Holländer. https://rb.gy/ofpm8g. 2018.

[40] Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. Quora question pairs, 2018.

[41] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[42] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40, 2016.

[43] Prerana Mukherjee, Brejesh Lall, and Snehith Lattupally. Object cosegmentation using deep siamese network. *arXiv preprint arXiv:1803.02555*, 2018.

[44] Saket Maheshwary and Hemant Misra. Matching resumes to jobs via deep siamese network. In *Companion Proceedings of the The Web Conference 2018*, pages 87–88, 2018.

[45] Quora question pairs — kaggle [online] available: https://www.kaggle.com/c/quora-question-pairs.

[46] Kaggle: Your home for data science [online] available:https://www.kaggle.com/.

[47] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: attentionbased convolutional neural network for modeling sentence pairs. corr abs/1512.05193, 2015.

[48] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.