MULTI-INTENT CLASSIFICATION IN CHATBOTS

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

in

Computer Science and Engineering

By

Asif Mohammad (106118058)

Sampurn Anand (106118085)

Supria Basak (106118116)

COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI-620015

MAY 2022

BONAFIDE CERTIFICATE

This is to certify that the project titled **MULTI-INTENT CLASSIFICATION IN CHATBOTS** is a bonafide record of the work done by

Asif Mohammad (106118058)

Sampurn Anand (106118085)

Supria Basak (106118116)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2021-22.

Dr. E. Sivasankar	**Dr. S. Mary Saira Bhanu**
Guide	Head of the Department

Project Viva-voce held on _____

Internal Examiner	**External Examiner**

ABSTRACT

Chat-bots or Chatter-bots in simple words are a program made to simulate and process human conversation (either written or spoken), allowing humans to interact with digital devices as if they are conversing with a real person. As queries from the users might not always be clear, intent classification aims to understand the user's goals in order to deliver the suitable response. In this project, we aim to test different sets of algorithms which can be used for intent classification. Towards the end of the project we aim to identify the best performing natural language processing model and develop a chat-bot for our institute's website. Recent works towards multi-intent classification, have concentrated on transfer learning using embeddings pre-trained on unrelated tasks. So, We take on an alternative approach using prototypical networks under the meta-learning paradigm by transfer learning on an ensemble of related tasks . We experiment with simple RNN, BiLstm, GRU, Naive bayes, Logistic Regression, SVM and BERT. For the dataset, we have used two standard benchmark open-source dataset named ATIS and SNIPS. The effect of various word embedding techniques including Bag of Words, Word2Vec etc. is considered as well. Each paradigm is compared using accuracy, macro f1 score, micro f1 score, and weighted f1 score as the metrics. From the results, we can say the Logistic Regression together with BERT works the best. We then develop a chat-bot using flask as back-end and JS, CSS, HTML as front-end to help the users to navigate through the websites available.

Keywords: Natural Language Processing; Chat-Bot; Multi-intent Classification, Recurrent Neural Networks, Word Embedding, Machine Learning

ACKNOWLEDGEMENTS

This project is the result of a dedicated team effort. It gives us immense pleasure to prepare this project report. To begin with, we would like to express our sincere gratitude to our guide, *Dr. E. Sivasankar*, Professor, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, for his continuous guidance and constructive suggestions on the matter through the course of this project. We also like to extend our heartfelt gratitude to *Mr. Nagarajan S*, PhD Scholar, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, for guiding us throughout the project, that enabled us to overcome our disruptions and complete our project successfully.

Our hearty thanks to all members of Departmental Project Evaluation Committee Members *Dr. Rajeswari Sridhar*, *Dr. M. Sridevi*, and *Dr. Sai Krishna Mothku* for their precious input and constructive criticism during the project reviews.

We are deeply grateful to *Dr. S. Mary Saira Bhanu*, HoD, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli for her constant encouragement.

Last but not least, we would also like to thank our parents and peers who have been a pillar of support.

Asif Mohammad (106118058)

Sampurn Anand (106118085)

Supria Basak (106118116)

TABLE OF CONTENTS

Title **Page No.**

ABSTRACT . . . . . **ii**

ACKNOWLEDGEMENTS . . . . . **iii**

TABLE OF CONTENTS . . . . . **iv**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

ABBREVIATIONS . . . . . **xi**

CHAPTER 1 INTRODUCTION **1**

1.1 Motivation 1

1.2 Basic Idea 2

1.2.1 What is Intent Classification? 2

1.2.2 Challenges with Intent Classification 3

1.2.3 Advantage of chat-bots 3

1.3 Objectives 4

1.4 Organization of the Report 6

CHAPTER 2 LITERATURE REVIEW 8

2.1 Natural Language Processing Tasks 8

2.2 Intent Classification 9

CHAPTER 3 IMPLEMENTATION DETAILS 15

3.1 Pre-Processing 17

3.1.1 Data Cleaning 17

3.1.2 Tokenization 18

3.1.3 Encoding 19

3.1.4 Train and Validate 19

3.1.5 Word Embedding Techniques 20

3.1.6 TF-IDF Weighted Averaging of Word Embeddings 20

3.1.7 Doc2Vec Training 21

3.2 Models for Text Classification 22

3.2.1 Naive Bayes Classifier 22

3.2.2 Support Vector Machines (SVM) 23

3.2.3 Recurrent Neural Network (RNN) 26

3.2.4 Long Short Term Memory (LSTM) Networks 27

3.2.5 Gated Recurrent Unit (GRU) 30

3.2.6 Logistic Regression Classifier 31

3.2.7 BERT 32

3.3 Chat-bot Development 39

3.3.1 Introduction 39

3.3.2 Step 1: Front-end 40

3.3.3 Step 2: Back-end 40

3.3.4 Step 3: Front-end and Back-end Integration 40

CHAPTER 4 DATASET, RESULTS AND ANALYSIS 41

4.1 Running Environment 41

4.1.1 Software Configuration 41

4.1.2 Hardware Configuration 41

4.2 Dataset 42

4.2.1 ATIS Dataset 42

4.2.2 SNIPS Dataset 43

4.3 Evaluation Metrics 44

4.4 Performance Analysis 46

4.4.1 Results on ATIS dataset 46

4.4.2 Results on SNIPS dataset 48

4.5 ChatBot Result 50

CHAPTER 5 CONCLUSION AND FUTURE WORKS 51

5.1 Summary 51

5.2 Conclusion 52

5.3 Future Scope 53

APPENDIX 54

1 Baseline Models 54

2 State of the Art Models 55

REFERENCES 59

PLAGIARISM **62**

List of Tables

4.1 Evaluation Metrics - ATIS 46

4.2 Evaluation Metrics - SNIPS 48

List of Figures

1.1 Sample chat-bot for College Query 4

1.2 Layered Input-Output 5

3.1 Methodology Flowchart - Baseline Models 15

3.2 Methodology Flowchart - State-of-the-art Model 16

3.3 Intent Classification Flowchart 19

3.4 Working of Support Vector Machine 24

3.5 An unrolled recurrent neural network 26

3.6 The repeating module in a standard RNN 28

3.7 The repeating module in a LSTM 28

3.8 Gated Recurrent Unit 30

3.9 General Architecture of The Transformer 35

3.10 The High-level Architecture of Encoder and Decoder in The Transformer 35

3.11 Detailed Working of the Encoder 37

3.12 Input Embeddings of BERT 38

3.13 NITT chat-bot 39

4.1 ATIS Dataset 42

4.2 SNIPS Dataset 43

4.3 Comparing Models for ATIS Dataset 47

4.4 Logistic Regression Accuracy - ATIS 47

4.5 Logistic Regression Loss - ATIS 48

4.6 Comparing Models for SNIPS Dataset 49

4.7 Logistic Regression Accuracy - SNIPS 49
4.8 Logistic Regression Loss - SNIPS 49
4.9 NITT ChatBot Demonstration 1 50
4.10 NITT ChatBot Demonstration 2 50
5.1 Plagiarism Report 62

Abbreviations

LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
GRU	Gated Recurrent Unit
RNN	Recurrent Neural Networks
SVM	Support Vector Machine
LR	Logistic Regression
FAQ	Frequently Asked Question
ML	Machine Learning
NLP	Natural Language Processing
AUC	Area under the ROC Curve
MT	Machine Translation
ATIS	Airline Travel Information System
SNIPS	SNIPS Natural Language Understanding benchmark
ANN	Artificial Neural Network
TF-IDF	Term Frequency–Inverse Document Frequency

CBOW	Continuous Bag Of Word
MMH	Maximum Marginal Hyperplane
K-NN	K-Nearest Neighborhood
BERT	Bidirectional Encoder Representations from Transformers
BERT-NER	Bidirectional Encoder Representations from Transformers - Name Entity Recognition
ELMo	Embeddings from Language Models
OpenAI GPT	OpenAI Generative Pre-trained Transformer
MLM	Masked Language Model

CHAPTER 1
INTRODUCTION

1.1 Motivation

The internet has spurred a shift in the way information and services are accessed. Every information is there on the web but scattered in its own way. Thus, Chat-bots are programs that is designed to simulate a human conversation or in fact ”chatter” through text or voice conversations. They are used to make any website or app more interactive. Based on the type of businesses for which the website is made, chat-bots can be of many types. They act as virtual assistants to handle simple tasks or provide valuable information just by a single query. Adding chat-bot assistants reduces overhead costs, uses additional staff time better and supports organizations to provide user service during hours when live agents aren't available.[2]

Every big organization which has their own website, do keep a chat-bot to handle simple tasks. Similarly, we are aiming to make a chat-bot for our very own college's website for answering college related queries that are frequently asked by students. As a fresher or even senior, it's really hard to keep on track of all the academic or other information at once and it takes a toll to inquire about them by visiting academic/hostel office or different websites when they need answers to their queries quickly. In a situation like this, a chat-bot can help by answering the most commonly asked queries, directing users to the right pages or sites and, if all else fails, gauging the type and seriousness of their issue before transferring them to a human support executive. [3]

Hence, to facilitate this process, we need to automate this process. Our chat-bot will help any user to navigate effortlessly through the website. Any task such as paying

fees, or showing results, or downloading full document will be handled effectively by the chat-bot in the blink of an eye.

The world of chat-bots has seen a constant increase in capability and adoption ever since its usefulness to us in speeding up; otherwise mundane tasks became evident. In the coming future, this simple chat-bot can be improvised to be the Educational-Bot which will change the teaching model. Students have a lot of questions and doubts and they need an immediate assistance on it. Appointing a dedicated human employee to address this may not be valuable as human resources have certain limitations. Chat-bots in education will enable teachers to reach, teach and counsel their students, address students questions on-the-go and help them learn faster with an innovative approach. In this proposed and experimented work that fills the rest of this report, we attempt to look into many of the existing machine learning models and methods, in order to identify what is suitable for tasks of intent classification. After this investigation of intent classification in chat-bots, we proceed to implement a chat-bot in the RASA framework according to our findings from the preceding investigation.

1.2 Basic Idea

1.2.1 What is Intent Classification?

Intent classification is the sub-task within chat-bot engineering, wherein the intent of the user (i.e. what answer/task the user expects from the chat-bot) is determined by analyzing the text provided by the user using machine learning and natural language processing techniques. The number of intents that a user might have is previously defined and it is the chat-bot's task to determine which of these finite categories does the user's query belong to. A chat-bot's accuracy in understanding what the user wants

is one of the most important considerations made in the development of a contextual assistant.

1.2.2 Challenges with Intent Classification

The goal of any chat-bot is to build human-like comprehensive conversations to meet the user's expectations. But sometimes, chat-bots deal with limitations in understanding the user's intent because –

- We represent our intents in varying forms and structures.
- Users can make spelling mistakes and might not provide a complete sentence.
- Language barrier might be there

1.2.3 Advantage of chat-bots

Chat-bot have certain features that make them preferable. They can be active at all times, also updates are done in a few minutes. They also allow organizations to gain insights and gather user data to provide more effective service. Unlike humans, chat-bots do not have emotions so they can provide consistent answers regardless of how many times a question is asked and do not lose their patience. Thus, a college chat-bot will be really helpful for students as well as the officials.

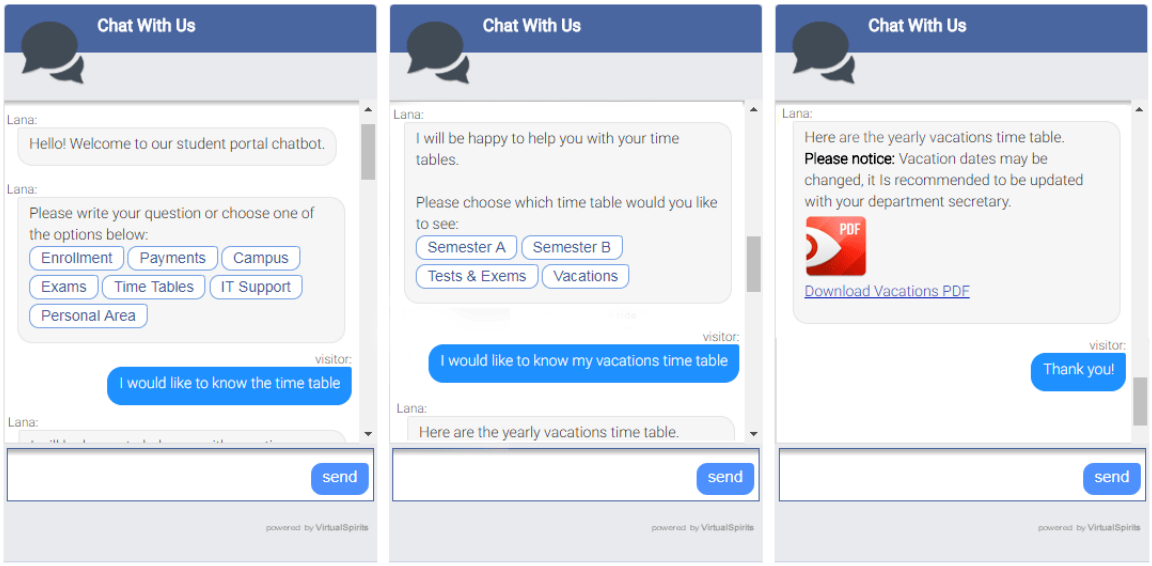


Figure 1.1: Sample chat-bot for College Query

1.3 Objectives

Our objective is to achieve multiple intents for our chat-bot from the sentence given to us from a dataset and then predict intents from a completely unknown field. We train the model on a set of different algorithms:

RNN(Recurrent Neural Network)

The RNNs are one of the the states of the algorithm in the deep learning since it is good for sequential data. RNN have internal memory so they can remember the input from previous layers. This is the reason RNN have edge over regular RNN when it comes to sequential data like text data, sentiment analysis, music generation,weather forecasting, time series. Here in this model, the first layer will be the embedding layer. Sentences will be represented as max_length by embedding_dim vectors and the next layer is a simple RNN layer. At last, the dense layers.

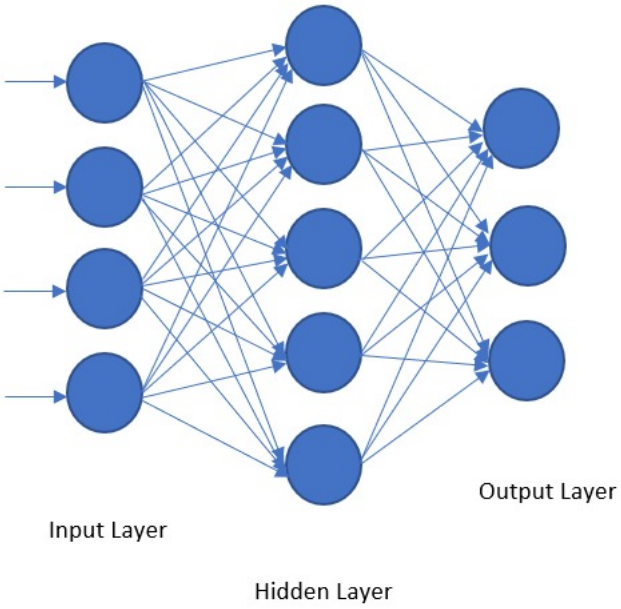


Figure 1.2: Layered Input-Output

GRU

GRU is an improved version of the RNN model and it is more efficient than SimpleRNN models. GRU has two gates: reset and update. We have replaced the SimpleRNN layer with a bidirectional GRU layer and kept the number of units same.

Bi-LSTM

We wanted to demonstrate the implementation of an Bi-LSTM model as well. The primary difference between an LSTM model and a GRU model is, LSTM model has three gates: input, output, and forget gates whereas the GRU model has two gates.[8] Being an extension of the traditional LSTMs, Bidirectional LSTMs can improve model performance on sequence classification problems. Bidirectional LSTMs train two instead of one LSTMs on the input sequence in the problems where all time=steps of the input

sequence are available.

SVM

Support Vector Machine(SVM) is a supervised machine learning algorithm which is used for classification and regression. Here we have taken different kernels like Linear, Poly, Rbf and Sigmoid.

Logistic Regression

Given an input variable, Logistic regression is the process for modeling the probability of a discrete outcome and the most usual logistic regression models a binary outcome that can take two values such as true/false, yes/no, and so on.

1.4 Organization of the Report

The remainder of the report/thesis is organized as follows:

- **Chapter 2** will cover the relevant background topics that are necessary to comprehend the remaining content of this work. First, this chapter will provide information related to the different natural language processing tasks that are used in this work. Second, it will cover the various intent classification tools and models that have been utilized previously. Finally, this chapter also includes a literature review on the conceptual ideas.

- **Chapter 3** will cover all the implementation details. At first, the steps of Data Pre-processing with formula have been discussed. Later on, all the models for text classification has been explained with respect to our project with all necessary

details. Then, BERT Model has been demonstrated with proper diagram. Finally, our college chat-bot's complete development has been shown with screenshots.

- **Chapter 4** will cover the details of our running environment, dataset analysis, evaluation metrics explanation with formula, and performance analysis with comparing models.

- Finally, in **Chapter 5**, a constructed summary of our project has been given along with conclusion and future scopes.

CHAPTER 2
LITERATURE REVIEW

This chapter focuses on providing key background information to help in the understanding of the Natural Language Processing Tasks that are most commonly used in research (with an emphasis on the tasks that have been used in this project), the ChatBot architecture, and its model variants such as BiLSTM and Few Shot Learning. It also discusses the methods of intent classification that have been in existence.

2.1 Natural Language Processing Tasks

Natural Language Processing (NLP) is a subset or a sub-field of Artificial Intelligence, Computer Science, and Linguistics which is used to circumscribe the gap of communication between the human and the computer. The idea of NLP originated from Machine Translation (MT) which came into existence in the 1940s (during the second world war). NLP has led a computer to be able to understand the contents of a document along with the contextual nuances of the language that is within it. This results in the computer being able to accurately extract information and insights that are contained in the document and also categorize, organize the documents themselves.

Even though the majority of the NLP tasks are closely related, they can be divided into categories for ease of understanding. The major NLP tasks can be broadly classified as follows: Text and Speech Processing tasks, Morphological Analysis, Syntactic Analysis, Lexical Semantics, Relational Semantics, and Higher Level NLP applications. A fraction of these tasks have straight through real-world applications whereas the others are used as sub-tasks and are used for solving the larger problems at hand.

2.2 Intent Classification

Intent classification is the automated categorization of text data based on users requirement. In addition, an intent classifier analyzes texts automatically and categorizes them into intents. So, Intents are collections of identified user intents (as in what does the user want).

There are a number of existing algorithms that have proven to be largely accurate in classifying the intent of conversations and questions of everyday life. However, even within these algorithms, some are found to be more effective for certain tasks. In order to discover the strengths and weaknesses of existing solutions for the goal of this project, we referenced some publications.

In **[13]**, Schuurmans and Frasincar attempt an investigation into several machine learning models that tackle the problem of intent classification for dialogue utterances.In this work, the authors have made use of both flat classifiers (distinguish between all classes at once) and hierarchical classifiers. A hierarchical classifier first performs a local classification at the parent node and then decides upon a sub section of relevant features for the classification process. This allows for the creation of subclasses within a category.

The models considered in this study are - Bag of Words with Multinomial Naive Bayes, Continuous Bag of Words with Support Vector Machines, Long Short Term Memory networks and a Bidirectional Long Short Term network. In addition, the authors have experimented using popular natural language understanding frameworks such as IBM's Watson, Microsoft's LUIS, the RASA open source chat-bot framework and Google's Dialog Flow. The effect of word embedding techniques such as

Word2Vec, GloVe and FastText on accuracy is also considered.

For the dataset, they used chat-bot Corpus on Travel Scheduling, and the StackExchange Corpus on Ask Ubuntu and Web Applications. For validation, Twofold cross validation using stratified sampling was done. Their F1 score was 0.812.The macro F1 score was measured for each model and the analysis revealed that the Support Vector Machine performs best, independent of the type of word embedding. Further, with regard to utterance representation, averaging the word embeddings proves to be better than just summing them. The best hierarchical SVM outperforms the best flat SVM was clearly depicted by their works.

In future, they wanted to deal with the more number of training observations. In their work, intermediate certainties could be exploited by the dialogue system, with specific follow-up questions. Instead of local hierarchical classifier per parent node, a global hierarchical classifier could be constructed by modifying a flat classifier to take the taxonomy into account at once.

In **[12]**, Vraj Desai, Sidarth Wadhwa, Anurag A and Bhisham Bajaj attempted to not only classify user intent but to also deliver an appropriate response. This work undertakes the understanding of user intent in two steps; first, the text is classified into the corresponding field and then an analysis is performed to gauge the similarity of the text with the classified field. Initially, the data is preprocessed to remove unwanted characters, similar words are grouped and then the data is lemmatized. The bag of words approach is used to represent the text and a sigmoid activation function is used to normalize and adjust the error rate. To classify a large number of queries, Artificial Neural Networks are used which can recognize patterns in the data. For their work, the authors have used a two layer neural network which translates to 1 input layer, 2 hidden layers and 1 output layer. The number of neurons used is fine-tuned starting from a base of 20.

To classify a large number of queries, Artificial Neural Networks are used which can recognize patterns in the data. Upon validation of their work, the authors have realized a 77.6 percent accuracy. They state that other constructs manage to achieve a higher accuracy, particularly LSTM with CNN, but the system becomes highly complex and the trade-off in terms of time and computational power is not advantageous. Possible areas of application include Spam detection, Advertising via social media, Customer Handling, etc. The conclusion lays out possible future research prospects particularly in dealing with emojis and slang language used on social media.

In **[11]**, Gangadharaiah and Narayanaswamy aimed to investigate approaches to the problem of multi-intent classification performing joint multi-intent classification both at sentence level and at token-level. Thier ATIS dataset contains audio recordings of people requesting flight reservations, with 21 intent types and 120 slot labels. The SNIPS data was collected from the SNIPS personal voice assistant, with 7 intent types and 72 slot labels. In their model, a bidirectional LSTM is used for the encoder layer. Multiple intents are predicted both at the sentence level (y_I) and at the token level (yM_I). y_I uses a feedforward network. Slot labels (y_S) and token level intent prediction (yM_I) both use LSTM layers, which have skip connections to the encoder states. They used F1 scores for intent detection at the token-level and accuracy for sentence-level intent detection. With ATIS dataset, F1 scores were 94.22(Slot), 95.82 (Intent- T level). With SNIPS dataset, F1 scores were 88.03(Slot), 97.89 (Intent T-level). With Internal Dataset, F1 scores were 90.94 (Slot), 94.54(Intent T level). They want to explore other architectures to directly model dependencies between slot labels and intents. We can further test the proposed approaches against real-world scenarios to understand their generality across various domains. Multi-intent classification both at sentence-level and at token-level can be performed using their works.

In **[15]**, Bihani and Rayz, proposed a scheme to address the ambiguity in single-intent as well as multi-intent natural language utterances. They used Modified ATIS dataset that has 1,066 utterances and 5 intents. The SNIPS dataset is considerably larger, containing 14,484 utterances and 7 intents. They did assessment of fuzzy membership. In this work, they proposed a framework towards fuzzy intent classification for unseen multi-intent utterances, without the need for the existence of prior multi-intent utterance data to learn intent memberships. Created degree memberships over fuzzified intent classes and the results reveal the impact of lexical overlap between utterances of different intents, and the underlying data distributions, on the fuzzification of intent memberships. The accuracy of the paper is influenced by the lexical similarity between utterances of different intents and the underlying distribution of data used to generate memberships.

Fuzzy classification accuracy:

ATIS: Data driven 70.18,

SNIPS: Knowledge based 91.52

This work can be improved by using softmax score distributions. Mapping can be adopted for our work. Mapping is taking the underlying data distribution into account when generating memberships yields more consistent results in mapping and emulating binary memberships.

In **[20]**, Saraswat, Abhishek, and Kumar explored how Zero-shot learning works and grounded on the idea that the sentence embedding of two sentences of different languages having the same sense must be similar. Their goal was to assess how useful these sentence representations are in the context of a particular scenario. For dataset, they used conversation logs were curated to create an intent list and multiple question variants used by the users querying about that intent. The dataset had 48 intents and 920 question variants The data collected was in English, it was converted to Hindi,

Arabic and Spanish using Google Translate. Analysis of multiple question used by the users querying about the intent was done. Random Forest Classifier, Support Vector Machine and Multi-layer Perceptron model was used. The embedding of questions in each language was calculated using the LASER model. It was simulated that for one question the embedding of two languages is identical, i.e. their cosine similarity should be 1.0. To prove this point the cosine similarity between pairs of two languages was calculated. The accuracy of the model was decided by the learning-rate, dropout and number of epochs. These are known as hyper-parameters. Stratified K Fold was used for training different models with different training data. The accuracy of the final model was 0.89. While training - the random forest gave better accuracy, but in testing the accuracy of multi-layer perceptron network was very promising. SVM was not that efficient but consistency can be seen in training and testing accuracy. Considering the results of the classifiers it was observed that the embedding obtained by using the encoder from LASER it can be deduced that the embeddings are efficiently representing the sentence in various languages. Concept of how to use zero shot learning can be adopted for our work.

In **[18]**, Kumar et al. approached by transfer learning on an ensemble of related tasks using prototypical networks under the meta-learning paradigm. They applied data augmentation in conjunction with meta-learning to reduce sampling bias and made use of a conditional generator for data augmentation that is trained directly using the meta-learning objective and simultaneously with prototypical networks, hence ensuring that data augmentation is customized to the task. They optimized data augmentation model using task-specific objective, and combined it with meta learning to improve intent classification performance in the few-shot setting. Modeling setup involves learning latent representations of text (e.g. sentence embeddings) using an encoder network, followed by classification using ProtoNets. They explored augmentation in the sentence embed-

ding space as well as prototypical embedding space. Combining meta-learning with augmentation provides upto 6.49% and 8.53% relative F1-score improvements over the best performing systems in the 5-shot and 10-shot learning, respectively. Specifically, the 5-shot and 10-shot settings result in 20.87% and 15.73% absolute improvement over the single-task, in comparison to 5.12% and 3.92% during seen intents. Single task transfer does not provide significant gains over ConvTL, even failing to outperform in the 5-shot case, ProtoNets benefit with increased task variability during multiple tasks, where transfer learning happens across corpora. Few-shot learning can be adopted from this project.

CHAPTER 3
IMPLEMENTATION DETAILS

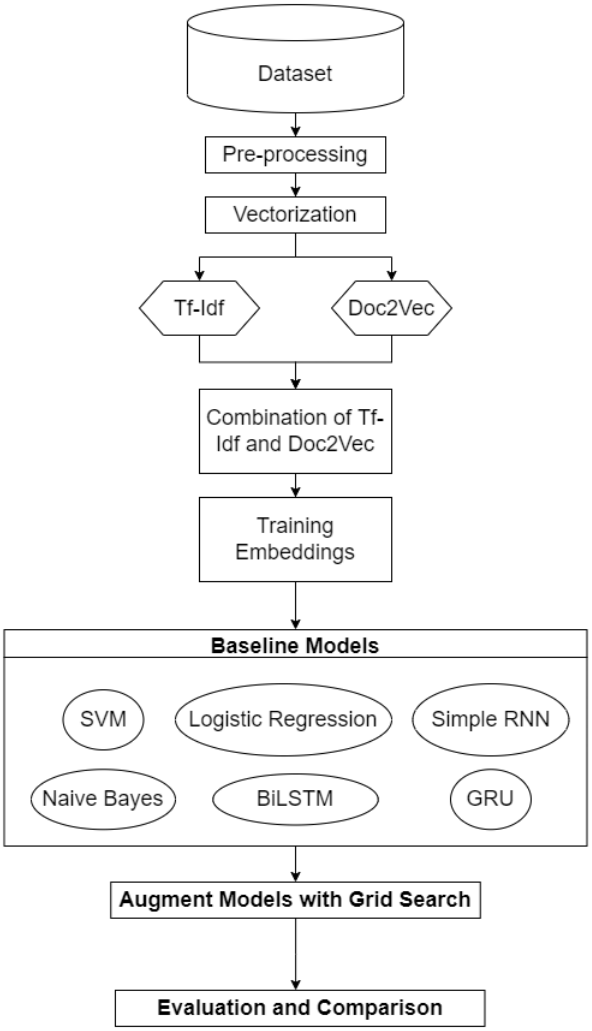


Figure 3.1: Methodology Flowchart - Baseline Models

Chat-bots can understand user requests thanks to Natural Language Processing (NLP). However, the conversation engine unit in NLP is critical for making the chat-bot more contextual and providing users with customized conversation experiences. Intent definition is a key feature of this chat-bot communication engine. It is a complex operation, despite how simple it can seem. Text input is recognized by a software feature known as a ”classifier,” which associates the data with a particular ”purpose,” resulting in a concise description of the terms for the machine to comprehend.

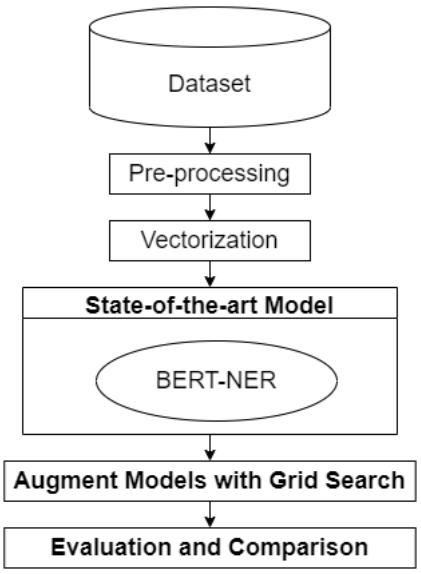


Figure 3.2: Methodology Flowchart - State-of-the-art Model

A classifier is a tool for categorizing data. Here in this case, a sentence into multiple categories. To understand the intention behind the information it has received, chat-bots can classify every part of a sentence into broken down categories. The process is similar to how humans classify items into sets, such as a guitar is an instrument, a coat is a form of clothing, and rejoice is an emotion.

- **Pattern Matching:**

Pattern matching requires using regex in order to find patterns in the text, and then classify it into different intents.

- **Machine Learning Algorithms:**

To construct multi-class classification, use a variety of machine learning algorithms. Machine learning systems allow chat-bots to be more contextual and analyze information about potential customers, process changes, and other factors.

• **Neural Networks:**

Word embedding is used to learn from text using neural networks. Deep learning is a form of machine learning technique that employs the Artificial Neural Network (ANN) principle. The artificial neural network (ANN) is a computational system based on biological neural networks. Without task-specific programming, these systems are trained and "learn" by improving their efficiency. 7 For a computer to deal with machine learning algorithms and neural networks, we need numeric representations of text. This is where sentence vectors come in. They are based on the principle of vector space models to provide a way to convert sentences typed by a user into a mathematical vector. The definition can then be represented in multi-dimensional vectors. These vectors can then be used to characterize intent and demonstrate how different sentences are linked.

3.1 Pre-Processing

Preprocessing of data is one of the crucial steps in any process. It involves transforming the raw data into an understandable format for further processing to increase the efficiency of the model.

Regardless of the model and word embedding techniques used, the steps followed for intent classification are the same.

3.1.1 Data Cleaning

Data Preprocessing is a proven method of eliminating errors in the dataset. Any real world data is prone to contain errors. By preprocessing the data, we can eliminate unwanted or irrelevant data before we train our model. Data cleaning is the process

of filling missed values, resolving inconsistency and smoothing out noisy data.– Raw data must be cleaned before it can be fed into the model. This step includes removing any empty cells and getting rid of unnecessary features. Then all the punctuations and special characters are removed since they have no effect on the result but make working with the data very difficult. All letters are changed into lowercase since case makes no difference to the model. Another vital/key step in the process is lemmatization where words with the same 'stem' are treated as the same. E.g: am, are, is $\longrightarrow$ be. These are removed by Python String manipulation functions using Regex.

3.1.2 Tokenization

Tokenization is the process of chopping text into individual units of Tokens by doing using techniques such as:

1. Removal of Punctuations and tokenizes the text.

2. Lemmatization: Stemming words and variations of the same word used in different parts of sentences into a single meaning conveying a token.

3. Removal of stop words: Some words of the English language do not contribute much to conveying the core meaning of the language are eliminated.

4. Create bi-gram words if required from the text. All of the above functionalities were implemented using a python class called DocProcess that operated on a Spacy Object.

3.1.3 Encoding

Computers can only understand 0s and 1s. Words must be converted into numbers for computers to understand but they must not lose their meaning. For this, various word embedding techniques have been developed that we will look at in the next section. In this step, the number of unique intents and the total vocabulary is gauged. This can give the user a rough idea of what the output must look like.

3.1.4 Train and Validate

Train and Validate – The data is split into training and validation sets (usually in the 4:1 ratio). The model must be defined and based on the type of model, we have to define the number of epochs, the loss function, the activation function, number of layers and other parameters that need to be finetuned. The graphs for the loss and accuracy are generated for a visual understanding of model performance.

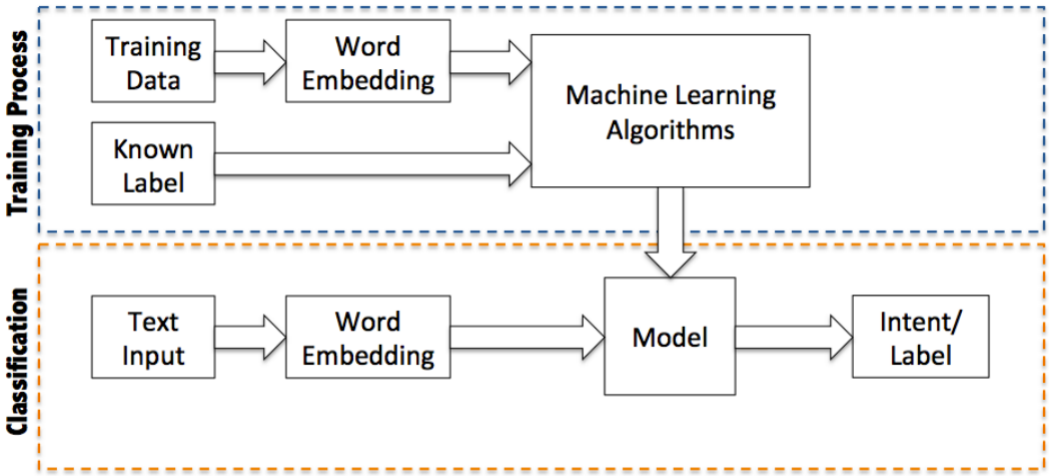


Figure 3.3: Intent Classification Flowchart

3.1.5 Word Embedding Techniques

Individual terms are interpreted as real-valued vectors in a predefined vector space in word embeddings, which is a class of techniques. Since every word is mapped to a single vector and the vector values are learned in a manner that resembles that of a neural network, the technique is often grouped with deep learning. Each word is expressed by a real-valued vector with several dimensions, often tens or hundreds. In comparison, sparse word representations, such as one-hot encoding, need thousands or millions of dimensions.

3.1.6 TF-IDF Weighted Averaging of Word Embeddings

TF stands for Term Frequency and IDF stands for Inverse Document Frequency. TF-IDF is a useful measure for feature extraction as it combines both the factors in identifying the important terms in the source document. Term frequency is a direct measure of how often the word occurs in the source document and is specified in equation 3.1. Inverse Document Frequency is a direct measure of how important a word is for the source document. The IDF of a term is the number of documents in the source divided by the number of documents containing the term. In general, words that occur very frequently like 'the', 'and', etc. are not important to the output summary. Hence to find the true inverse document frequency, we take log of the obtained IDF value as in equation 3.2.

$$tf(t,d) = \frac{Total no. of terms in the document d}{No. of times term t appears in a document d} \tag{3.1}$$

$$idf(t, D) = log\left(\frac{Totalno.ofdocumentsD}{No.ofdocumentshavingthetermt}\right) \tag{3.2}$$

The final tf-idf value is simply the product of the individual tf and idf values as calculated in 3.1 and 3.2. The final tf-idf value is obtained by equation 3.3. The larger the value, the more important is the word for the output summary. The tf-idf value is calculated for each word in the source vocabulary and appended to the basic word embeddings.

$$tfidf(t, d, D) = tf(t, d).idf(t, D) \tag{3.3}$$

TF-IDF weights calculated can be really useful to normalize the average while considering frequencies of words occurring across documents. The obtained values are used as weights to average embeddings to compute the Doc Vector. [9]

3.1.7 Doc2Vec Training

Here, we combine training of word embeddings and calculation of Doc Vector [6] by training an additional paragraph/document-specific vector every time a word is trained as a part of the continous Bag of Words (CBOW) process of training. Thus, after training, we will obtain a vector(numeric representation of the document) along with the convergence of word embeddings. This modeling strategy is introduced as the Distributed Memory Version of Paragraph Vector (PV-DM).

3.2 Models for Text Classification

3.2.1 Naive Bayes Classifier

The Naive Bayes classifier is based on the Bayes theorem which is used for predictive modeling. It is used Where the dimensionality of the inputs is high and the predictors are independent. It is a family of simple probabilistic classifiers ,and based on a common assumption that given the category variable, all features are independent of each other. It is often used as the baseline in text classification [10] The Bayes theorem is written as follows:

$$P(class/features) = \frac{P(class) * P(features/class)}{P(features)} \tag{3.4}$$

The continuous values related with each class in Gaussian Naive Bayes are distributed in the form of a Gaussian distribution. (3.2)

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(v-\mu_c)^2}{2\mu_c^2}} \tag{3.5}$$

Where c is a class, x is a constant attrte, and is the average of the x values for class c.

When it comes to document classification, these models are frequently used. The number of times a word come into view in a single document is represented by each case.

The probability that is observed for X is given by:

$$p(x|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{k_i}^{x_i} \tag{3.6}$$

where X_i represents the number of times the event I occurs in a given case, and X is a function vector.

There are various variants of this algorithm including:-

Multinomial Naive Bayes: This is commonly used to solve the document classification dilemma, such as determining if a document falls under the sports, politics, or technology categories. The frequency of the terms present in the text is one of the features/predictors used by the classifier.

Gaussian Naive Bayes: We can assume that the values are sampled from a gaussian distribution when the predictors have a continuous value not being discrete.

Bernoulli Naive Bayes: Features are basically the independent booleans-binary variables, which describe inputs in the multivariate Bernoulli event model. This model is like the multinomial model being common for document classification tasks where binary term occurrence features are used instead of term frequencies.

3.2.2 Support Vector Machines (SVM)

A support vector machine (SVM) [5] is a supervised machine learning model that uses classification algorithms and is useful for two-group classification problem. SVM models categorizes new text after having labeled training data for each group. Support Vector Machines are typically considered as a classification method. However, they can be

used to solve both classification and regression problems, and can manage both continuous and categorical variables. To divide different groups, SVM creates a hyperplane in multidimensional space. SVM one-by-one produces the best hyperplane. The hyperplane is then used to minimize an error. Basically, the aim of SVM is to find the maximum marginal hyperplane (MMH), which divides a dataset into evenly divided classes.

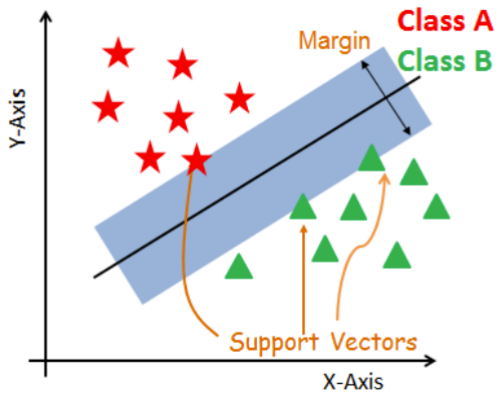


Figure 3.4: Working of Support Vector Machine

Support Vectors: The data points nearest to the hyperplane are called support vectors. By measuring margins, these points will better describe the separating rows. These points are more applicable to the classifier's construction.

Hyperplane: A hyperplane is a decision plane that divides a group of objects that belong to different classes. The equation of a hyperplane for SVM is

$$w^T + b = 0 \tag{3.7}$$

Where w is the coefficient matrix, x is the weight vector and b is the bias.

Margin: A margin is the distance between the two lines on the class points that are nearest to each other, and the perpendicular distance from the line to the support vectors

or closest points is determined. A greater margin between the groups is considered a decent margin, whereas a smaller margin is considered a poor margin.

$$(x^+ - x^-).\hat{w} = (x^+ - x^-).\frac{w}{||w||} = x^+.\frac{w}{||w||} - x^-.\frac{w}{||w||} \tag{3.8}$$

Here, x+ is a positive example for a support vector and x- is a negative example. Hinge loss is the loss function that helps maximize the margin is h.

$$\begin{aligned} max(0, 1 - y_i[w^T x_i + b]) &= 0 \\ \implies y_i[w^T x_i + b] &= 1 \\ x^+.\frac{w}{||w||} - x^-.\frac{w}{||w||} = x^+.\frac{1-b}{||w||} - x^-.\frac{-b-1}{||w||} &= \frac{2}{||w||} \\ max\frac{2}{||w||} \rightarrow max\frac{1}{||w||} \rightarrow min||w|| \rightarrow min\frac{1}{2}||w||^2 & \end{aligned} \tag{3.9}$$

Linear SVM: Linear SVM is used for linearly separable data. If a dataset can be divided into two groups using only a single straight line, then it is called linearly separable data, and the classifier used is called Linear SVM.

Non-linear SVM: Non-Linear SVM is used to classify non-linearly separated data. If a dataset cannot be categorized using a straight line, then it is classified as non-linear data, and the classifier used is a Non-linear SVM classifier.

Poly SVM: Poly SVM shows the similarity of vectors . As example, training samples in a feature space over polynomials of the original variables which allows learning of non-linear models.

Rbf SVM: Rbf is the default kernel used in the sklearn's SVM classification algorithm. Rbf's similarity to K-Nearest Neighborhood(K-NN) Algorithm made it a popular and

useful kernel. It has all the advantages of K-NN and overcomes the space complexity problem because Rbf only needs to store the support vectors during training (not the entire dataset).

Sigmoid SVM: Sigmoid SVM is identical to a two-layer, perceptron model of the neural network, that is used as an activation function for artificial neurons. Sigmoid Kernel was popular for its origin from neural network theory.

3.2.3 Recurrent Neural Network (RNN)

In the myriad of neural networks, RNN is the feed forward neural network which has an internal memory that has been generalized. It is repetitive since it iterates an equivalent operation for every input. The present input's output is usually affected by the previous computation. The output is traced and sent into the recurrent network once it's created. It considers each current input, and therefore the output it's non-heritable from the previous input while decision-making.

RNNs can use their internal state to process a succession of inputs (memory). As a result, tasks such as unsegmented, linked handwriting recognition and speech recognition can be accomplished. In other neural networks, all of the inputs are unrelated to one other. All of the inputs in an RNN, on the other hand, are connected. It initially ex-

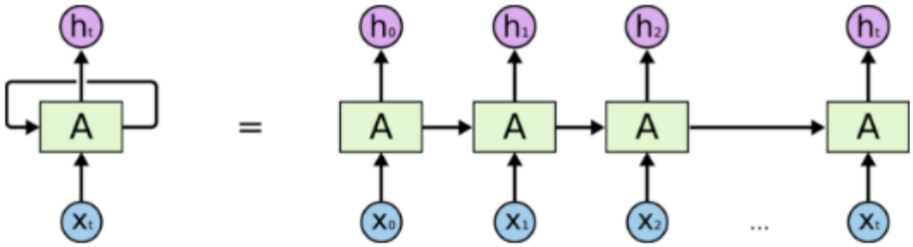


Figure 3.5: An unrolled recurrent neural network

tracts X(0) from the sequence of inputs, then produces h(0), which, together with X(1), serves as the following step's input. As a consequence, the inputs for the following step are h(0) and X. (1). Similarly, the input for the following stage is h(1), and the input for the next step is X(2), and so on. As an outcome, it recalls the context when learning. The current state is

$$h_t = f(h_{t-1}, x_t) \tag{3.10}$$

Applying Activation Function:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{3.11}$$

Where W is the weight in the above equation, h is the single hidden vector, W_{hh} is the weight at the previous hidden state, W_{xh} is the weight at the current input state, and $\tanh$ is the activation function, which implements a non-linearity that squashes the activations to the range[-1.1]

Output:

$$y_t = h_t W_{hy} \tag{3.12}$$

Here, y_t is the output state. W_{hy} is the weight at the output state.

3.2.4 Long Short Term Memory (LSTM) Networks

Long Short Term Memory (LSTM) networks are a kind of RNN capable of learning long-term dependencies. LSTMs were created expressly to avoid the problem of long-term reliance.

A set of recurring neural network modules make up both recurrent neural networks. In typical RNNs, this repeating module will be quite basic, with only one $tanh$ layer.

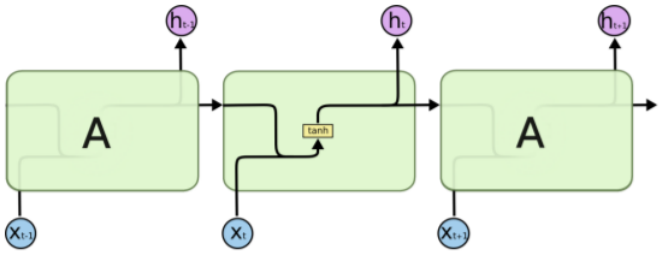


Figure 3.6: The repeating module in a standard RNN

Although LSTMs have a chain-like structure, the repeating module is unique. There are four neural network layers instead of one, each of which interacts in a different way.

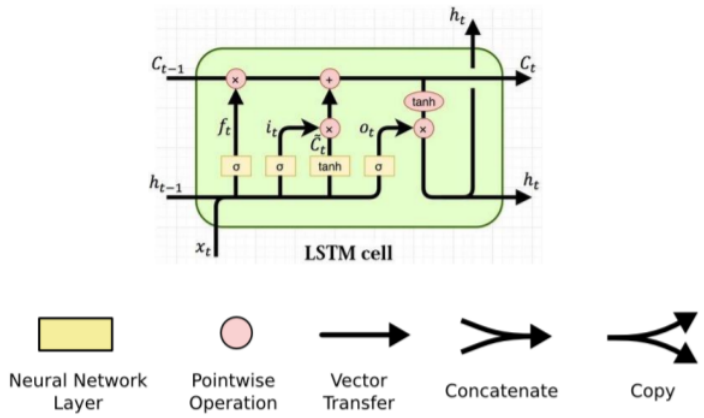


Figure 3.7: The repeating module in a LSTM

Each line in the figure above transports a full vector from the output of one node to the inputs of others. Pink circles represent point-wise operations like vector addition, whereas yellow boxes represent learned neural network layers. Concatenation happens when lines merge, whereas forking occurs when the material of a line is replicated and the copies transmitted to various locations. The equations of the LSTM are:

$$i_t = \sigma(h_{t-1}W^i + x_tU^i) \tag{3.13}$$

$$f_t = \sigma(h_{t-1}W^f + x_tU^f) \tag{3.14}$$

$$o_t = \sigma(h_{t-1}W^o + x_tU^o) \tag{3.15}$$

$$\tilde{C_t} = \tanh\left(h_{t-1}W^g + x_tU^g\right) \tag{3.16}$$

$$C_t = \sigma(i_t * \tilde{C_t} + f_t * C_{t-1}) \tag{3.17}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{3.18}$$

i_t - input gate,

f_t - forget gate,

O_t - forget gate,

σ - sigmoid function,

W^x - weight for the respective gate,

$h_t - 1$ - output of previous lstm block,

X_t - current input,

U_x - the product of previous state and bias,

C_t - represents current cell state, and

$\tilde{C_t}$ - a candidate for cell state.

The key to LSTMs is the cell condition, which is represented by the horizontal line at the top of the picture. The cell's state is comparable to that of a conveyor belt. It goes straight down the chain with only a few minor linear interactions. It's quite normal for information to pass via it without being affected. The LSTM has the capacity to delete or add information to the cell state, which is carefully controlled by structures known as gates. Gates are a method of allowing information to pass through if desired. A sigmoid neural network layer and a point-wise multiplication procedure are used to create them. The sigmoid layer generates values ranging from 0 to 1, indicating the amount of each part that should be permitted to pass. "Nothing should be let through!" says a value of zero, whereas "everything should be permitted through!" says a value of one. To safeguard and monitor the cell state, an LSTM contains three of these gates.

3.2.5 Gated Recurrent Unit (GRU)

A gated recurrent unit (GRU) is a gating mechanism for recurrent neural networks (RNNs) that works similarly to a long short-term memory (LSTM) unit but does not contain an output gate. GRUs solve the problem of vanishing gradients that can arise in traditional recurrent neural networks. A GRU may be thought of as a version of the long short-term memory (LSTM) unit since they have a similar architecture and provide comparable outcomes in some instances. GRUs can overcome the vanishing gradient problem by employing an update and a reset gate. The update gate controls data flow into memory, whereas the reset gate controls data flow out of memory. Two vectors, the update and the reset gates, govern which data is transmitted to the output. They may be taught to keep historical data while eliminating material that is irrelevant to the forecast.

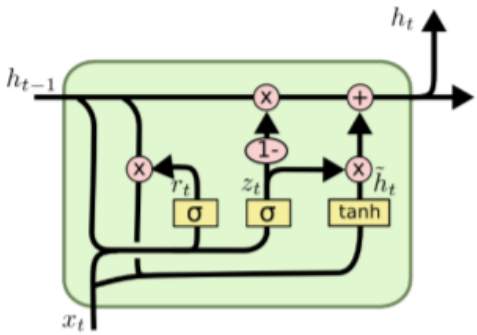


Figure 3.8: Gated Recurrent Unit

$$z_t = \sigma([x_t, h_{t-1}].W_z) \tag{3.19}$$

$$r_t = \sigma([x_t, h_{t-1}].W_r) \tag{3.20}$$

$$\tilde{h}_t = \tanh([x_t, h_{t-1}.r_t].W) \tag{3.21}$$

$$h_t = \tilde{h}_t * z_t + h_{t-1} * (1 - z_t) \tag{3.22}$$

Here, z_t - update gate and r_t - reset gate.

3.2.6 Logistic Regression Classifier

It is a machine learning predictive analysis approach for categorical classification problems that links features (input) to an outcome probability. As a result, it is utilised to allocate observation to a certain set of classes. There are two forms of logistic regression:

1. binary

2. Multilinear functions

It returns a probability value by transforming its output using the logistic sigmoid function logistic function (cost function). The hypothesis implies that the cost function should be limited (between 0 and 1).

LR Hypothesis expectation-

$$0 \leq P(X) \leq 1 \tag{3.23}$$

Sigmoid function's formula-

$$f(x) = \frac{1}{e^{-(x)+1}} \tag{3.24}$$

For using the linear regression we used the formula of the hypothesis-

$$h_\theta(x) = \beta_0 + \beta_1 x \tag{3.25}$$

For logistic regression we modified it to-

$$\sigma(Z) = \sigma(\beta_0 + \beta_1 X) \tag{3.26}$$

Then, we expected the values of our hypothesis to be between 0 and 1.

$$\begin{aligned} Z &= \beta_0 + \beta_1 X \\ h_\Theta(Z) &= sigmoid(Z) \\ \text{i.e.,} \quad h_\Theta(X) &= \frac{1}{e^{-(\beta_0+\beta_1 X)} + 1} \end{aligned} \tag{3.27}$$

Final equation is-

$$P(X) = \frac{1}{e^{-(\beta_0+\beta_1 X)} + 1} \tag{3.28}$$

3.2.7 BERT

Introduction

BERT or Bidirectional Encoder Representations from Transformers is a machine learning technique which is transformer-based. It is a natural language processing pre-training model designed and published by Jacob Devlin and his colleagues from Google in 2018. Language model pre-training has proved to be very effective to improve the Natural Language processing tasks. Training parts of a neural network algorithm at different times is what pre-training refers to. Pre-trained representations absolves requirement of heavily-engineered task specific architectures. Feature-based approach and fine-tuning approach are the two existing strategies for pre-training language models. There are many state-of-the-art models which involve pre-training, such as ELMo (Embeddings from Language Models) and OpenAI GPT (OpenAI Generative Pre-trained Transformer). ELMo, which is a feature-based approach, uses the pre-trained representations as additional features in training, as it primarily uses a task-specific architecture. OpenAI GPT on the other hand, does not focus on task-specific parameters. It is trained

on all the downstream tasks by fine-tuning all the pertained parameters. The limitations of these lie in the fact that the standard language models are unidirectional, and hence the choice of the architectures which can be used during pre-training, are limited. For instance, in the OpenAI GPT model, left-to-right architecture is used, where each token can only attend to previous tokens in the self-attention layers of the Transformer. BERT is designed to pre-train deep bidirectional representations text that is unlabeled by simultaneously training on both the left and the right context in all layers, therefore it is 'bi-directional'. It achieves this using a MLM (masked language model) pre-training objective.

It is the first deeply bidirectional language representation. That is, it worked on the words after a given word, not just the preceding words. This gives it a comparatively superior performance.BERT is the first fine-tuning-based representation model that outperforms many task-specific architectures and achieves state-of-the-art performance on Language tasks. Additionally, its architecture is transformer-based, which weighs the context of a word in the sentence. It was, therefore, not a surprise that it achieved State-of-the-art (SOTA) results in multiple NLP tasks like Sentiment Analysis, Neural Machine Translation, Text Summarization etc. The masked language model (MLM) is for randomly masking some of the tokens from the input. The objective is to make a context-based prediction of the original vocabulary ID. The MLM pre-training allows the representation to fuse the left and the right context. This enables us to pre-train a deep bidirectional Transformer. This is the primary difference in comparison to the left-to-right language model pre-training. To jointly pre-train text-pair representations, a "next sentence prediction" task is also added to the masked language model. Pre-training and Fine-tuning are the two steps detailing the framework of BERT. The model is trained on unlabeled data during pretraining and during fine-tuning, the BERT model is initialized with the pre-trained parameters. All of the parameters are then fine-tuned

using labeled data from the downstream tasks and all downstream tasks are initialized with the same pre-trained parameters, but each task has discrete fine-tuned models.

Architecture

BERT is a multi-layer bidirectional Transformer encoder. Its official paper produces 2 model sizes for BERT:

1. BERT Base: L=12, H=768, A=12

2. BERT Large: L=24, H=1024, A=16,

where L is the number of stacked encoders, H is the number of hidden layers, and A is the number of heads in the multi-headed Attention layers.

The Transformer - Encoder Stack

The Transformer uses Attention to boost the training speed of neural machine translation models. This was brought to light in the paper titled 'Attention Is All You Need.' It outperforms other models as it utilizes the concept of parallelization. The Transformer consists of two main components - an encoder and a decoder.

For the purpose of BERT for Intent classification, only the Encoder part of the Transformer will be used.

The encoding component consists of a stack of encoders, which are identical in structure, placed on top of each other. Similarly, the decoding component consists of an equal number of decoders. For instance, the transformer used for the task of machine translation is illustrated below figure: 3.9:

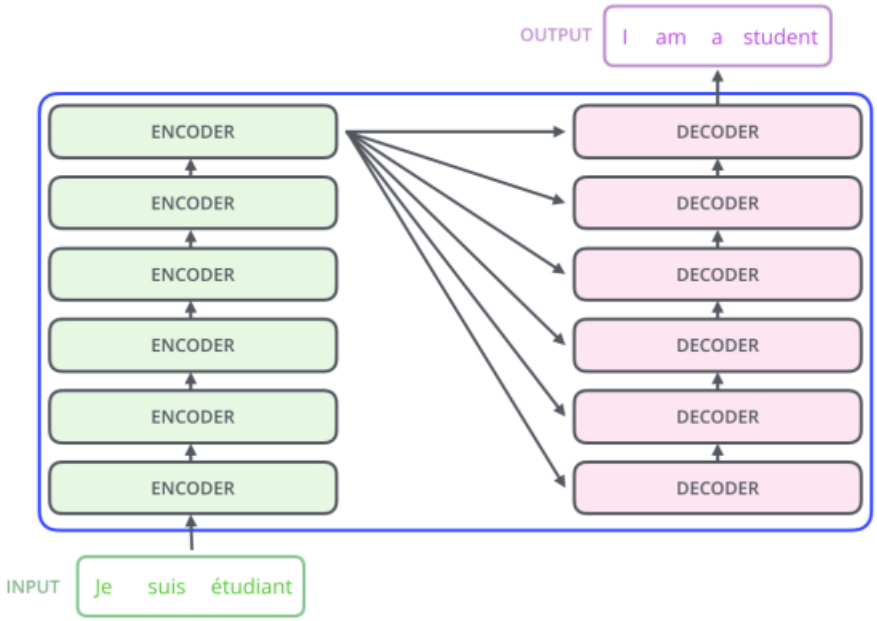


Figure 3.9: General Architecture of The Transformer

The encoder, in turn, is broken down into a Self-Attention layer and a Feed Forward layer as shown below in figure 3.10.

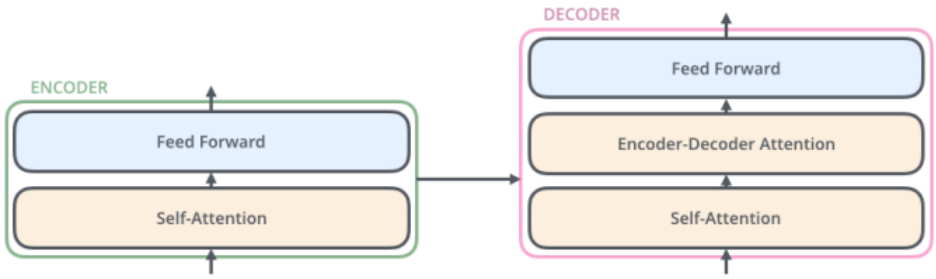


Figure 3.10: The High-level Architecture of Encoder and Decoder in The Transformer

Feed Forward Network

At each position, the Feed Forward Neural Network is applied independently as this process does not require interdependence between multiple word embeddings. Therefore this stage can be easily parallelized for speed.

Positional Encoding

The final step required by the transformer encoder accounts for the order of input words: To do this, a vector is added to each embedding in the transformer. It helps in determining the position of each word by forming an abstraction of distance between different words in the sequence. Incorporation of these values to embeddings makes the distance between embeddings meaningful for attention calculation.

The Transformer was used in OpenAI GPT in which model is pre-trained and this was later applied and tested on a variety of language tasks. For ensuring bi-directionality in BERT, rather than pre-training the model on a language model, the model is pre-trained into the ”masked language model” and ”next sentence prediction”. BERT only uses the encoder part of the Transformer architecture as described in the paper ‘Attention Is All You Need’. The figure 3.11 illustrates the same.

Working Of BERT

The precedent topics were prerequisites to understanding the working of BERT. As already seen before, the BERT comes in two types. The first one is BERT base and the other is BERT large. The base version consists of 12 encoder layers, 768 hidden units and 12 attention heads while the larger version consists of 24 encoder layers, 1024 hidden units and 16 attention heads.

Handling Bi-directionality

Models/Architectures prior to BERT, namely OpenAI GPT, and Transformer handled Language Modelling tasks differently in comparison to BERT. Let’s take an example input sentence - ’I love to work on NLP’. The ’love’ token would just exhibit a self-

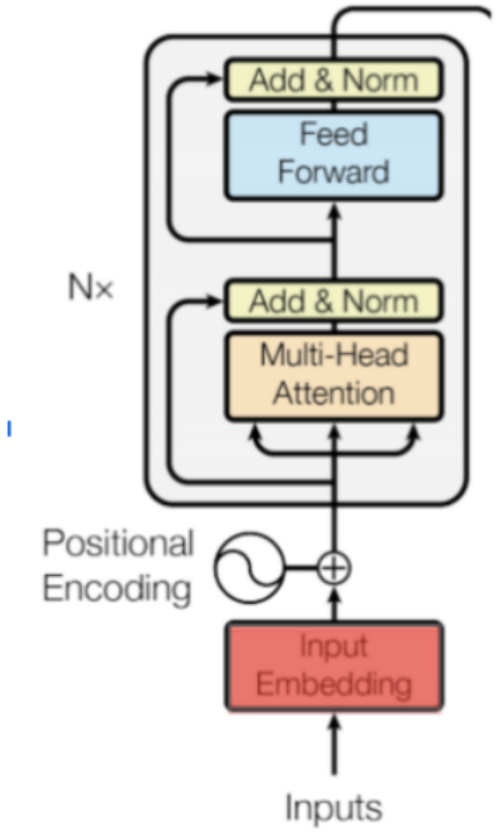


Figure 3.11: Detailed Working of the Encoder

attention relationship with the 'I' token and itself. This relationship exists only backwards in the case of OpenAI GPT. But in the case of BERT, the same token would exhibit a self-attention relationship with the rest of the tokens present in the sentence.

This is how BERT handled Bi-directionality, which wasn't done by previous models that were pre-training on Language Models:

In place of performing pre-training on language models, BERT pre-trains it using "masked language model" and "next sentence prediction".

Input Embeddings for BERT

BERT differs from the Transformer in how it handles input sequences by increasing the number of tasks of the working model. Figure 3.12 illustrates how inputs are handled in BERT.

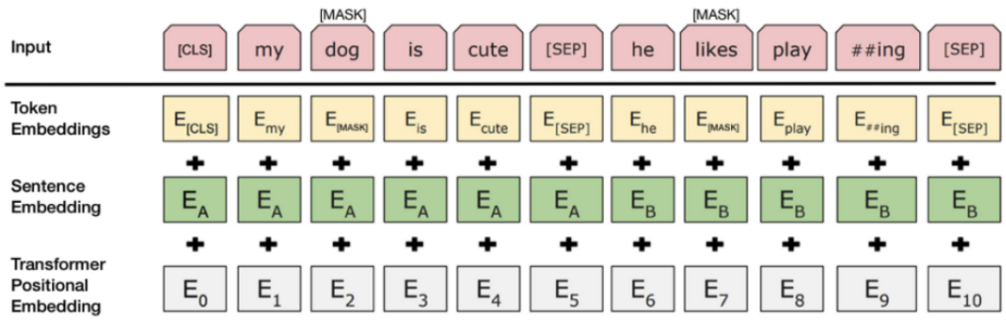


Figure 3.12: Input Embeddings of BERT

As shown above, the components of the Transformer i.e, positional embeddings and token embeddings are still there. But the difference lies in BERT using the learned positional embeddings and also making use of pre trained token embeddings.

- In order to find a solution for the two sentence problems i.e. next sentence prediction and question answering, a [SEP] token was added to mark the end of a sentence and added a sentence embedding. This sentence embedding is constant for each sentence but different across two sentences. This allows the model to give different meanings to different sentences by knowing where one ends another begins.
- Also there is an addition of the [CLS] token at the start of the input sequence for classification.

3.3 Chat-bot Development

3.3.1 Introduction

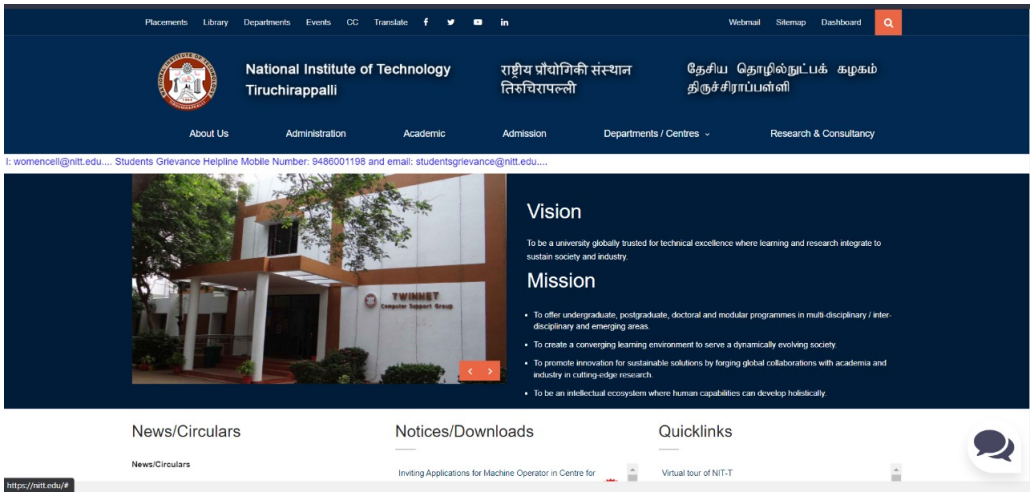


Figure 3.13: NITT chat-bot

The chat-bot is built to make the students' experience smooth. From greeting the students, enlightening their mood with subtle jokes to redirecting them to the page that they want to visit or the information they need, this chat-bot does it all.

There were two deployment options:

First, to use standard Jinja-2 template of chat-bot where directly python files will be used.

Second, to code separate Front-end HTML and CSS files. It can be run completely separate from the Flask App.

The second option has been used. Currently, for Back-end, Flask App has been used and for Front-end - JS, CSS, HTML has been used.

3.3.2 Step 1: Front-end

For front-end part, a clone of official institute website's HTML, CSS and JS pages was made. Then, a separate class for the chat-bot was defined. Inside that chat-bot class, js and css files were referenced which made the bot dynamic.

3.3.3 Step 2: Back-end

Initially, Python files for the previously mentioned algorithms were made and trained on the dataset. The same files are now used after training. A separate json file is made for intents. This file will contain all the responses for the possible queries from user. Now, the model is re-trained with this json file to obtain a data.pth file.

3.3.4 Step 3: Front-end and Back-end Integration

A separate python file for deploying the website is made.The data.pth file obtained from the previous step is now used here to instantly give responses. Anytime, one can deploy the website locally using this file and can interact with the bot. The same set of files can be used to deploy on the server also.

CHAPTER 4
DATASET, RESULTS AND ANALYSIS

4.1 Running Environment

The programs run do not require the use of any particular operating system, nor does it have any OS specific dependencies. However, the running of the program requires a list of other dependencies, all of which are Python based and which have been listed below.

For the purposes of cross compatibility and collaborative experimentation, we have chosen to perform our experiments on Google Collaboratory, an open-source tool that allows for the running of Python notebooks. The hardware configuration listed below is of the Google Colaboratory runtime and not of our local systems.

4.1.1 Software Configuration

Operating System: Windows 11

Programming Languages: Python 3, HTML, CSS, JS, Flask

Tools Used: Google Colaboratory

4.1.2 Hardware Configuration

CPU Count: 4

Processor: Intel(R) Core i5 8th Gen(R) CPU @ 1.6 Ghz. Turbo Upto 3.4 Ghz

GPU: NVIDIA GeForce MX130

RAM: 16 GB

4.2 Dataset

4.2.1 ATIS Dataset

The ATIS dataset [1] is a standard benchmark dataset which is widely used as an intent classification. ATIS refers to Airline Travel Information System. The dataset consists of audio recordings and corresponding manual transcripts about humans asking for flight information on automated airline travel inquiry systems.

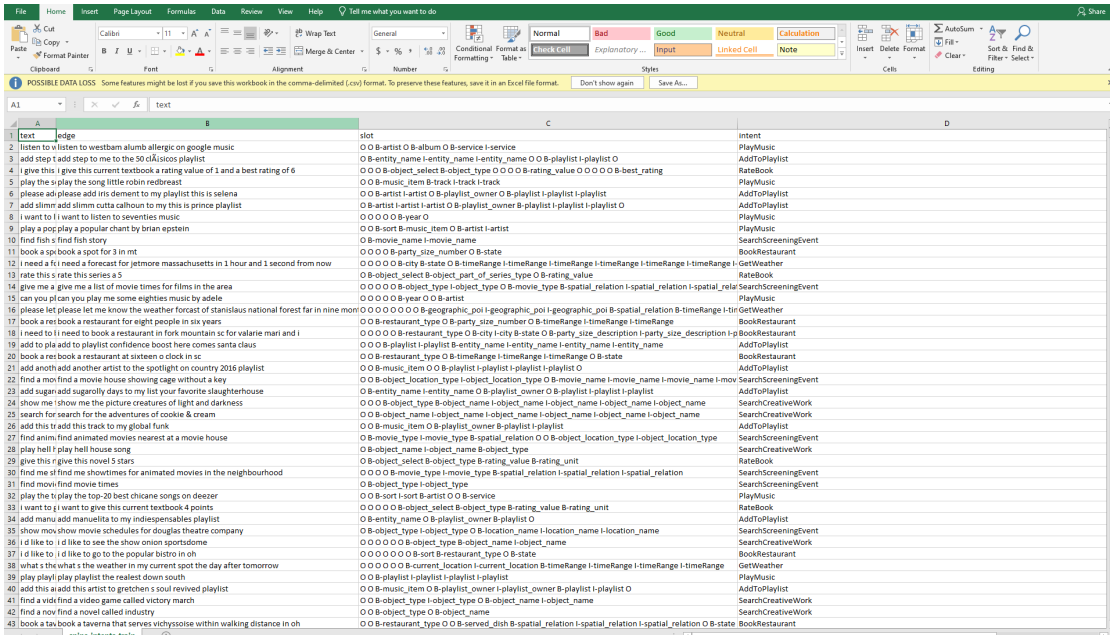
The data comprises 17 unique intent categories. ATIS dataset provides large number of messages and their associated intents that can be used in training a classifier. The speech has many of the characteristics of spontaneous spoken language (e.g., disfluencies, false starts, and colloquial pronunciations) which makes it more effective.

atis_flight	tokens
atis_flight	what flights are available from pittsburgh to baltimore on thursday morning
atis_flight_	what is the arrival time in san francisco for the 755 am flight leaving washington
atis_airfar	cheapest airfare from tacoma to orlando
atis_airfar	round trip fares from pittsburgh to philadelphia under 1000 dollars
atis_flight	i need a flight tomorrow from columbus to minneapolis
atis_aircra	what kind of aircraft is used on a flight from cleveland to dallas
atis_flight	show me the flights from pittsburgh to los angeles on thursday
atis_flight	all flights from boston to washington
atis_groun	what kind of ground transportation is available in denver
atis_flight	show me the flights from dallas to san francisco
atis_flight	show me the flights from san diego to newark by way of houston
atis_flight	what is the cheapest flight from boston to bwi
atis_flight	all flights to baltimore after 6 pm
atis_airfar	show me the first class fares from boston to denver
atis_groun	show me the ground transportation in denver
atis_flight	all flights from denver to pittsburgh leaving after 6 pm and before 7 pm
atis_flight	i need information on flights for tuesday leaving baltimore for dallas dallas to boston and boston to baltimore
atis_flight	please give me the flights from boston to pittsburgh on thursday of next week
atis_flight	i would like to fly from denver to pittsburgh on united airlines
atis_flight	show me the flights from san diego to newark
atis_flight	please list all first class flights on united from denver to baltimore
atis_aircra	what kinds of planes are used by american airlines
atis_airfar	i'd like to have some information on a ticket from denver to pittsburgh and atlanta
atis_flight	i'd like to book a flight from atlanta to denver
atis_airline	which airline serves denver pittsburgh and atlanta
atis_flight	show me all flights from boston to pittsburgh on wednesday of next week which leave boston after 2 o'clock pm
atis_groun	atlanta ground transportation

Figure 4.1: ATIS Dataset

4.2.2 SNIPS Dataset

The SNIPS[7] Natural Language Understanding benchmark is a popularly used dataset, containing 16,000 crowd sourced queries that is distributed among 7 user intents of various complexity: SearchCreativeWork (e.g. Find me the I, Robot television show), GetWeather (e.g. Is it windy in Boston, MA right now?), BookRestaurant (e.g. I want to book a highly rated restaurant in Paris tomorrow night), PlayMusic (e.g. Play the last track from Beyoncé off Spotify), AddToPlaylist (e.g. Add Diamonds to my roadtrip playlist), RateBook (e.g. Give 6 stars to Of Mice and Men), SearchScreeningEvent (e.g. Check the showtimes for Wonder Woman in Paris).



	A	B	C	D
1	text	edge	slot	intent
2	listen to vi	listen to westbam alumb allergic on google music	O O B-artist O B-album O B-service I-service	PlayMusic
3	add step t	add step to me to the 50 clÃ¡sicos playlist	O B-entity_name I-entity_name I-entity_name O O B-playlist I-playlist O	AddToPlaylist
4	i give this	i give this current textbook a rating value of 1 and a best rating of 6	O O O B-object_select B-object_type O O O O B-rating_value O O O O O B-best_rating	RateBook
5	play the si	play the song little robin redbreast	O O B-music_item B-track I-track I-track	PlayMusic
6	please ad	please add iris dement to my playlist this is selena	O O B-artist I-artist O B-playlist_owner O B-playlist I-playlist I-playlist	AddToPlaylist
7	add slimm	add slimm cutta calhoun to my this is prince playlist	O B-artist I-artist I-artist O B-playlist_owner B-playlist I-playlist I-playlist O	AddToPlaylist
8	i want to l	i want to listen to seventies music	O O O O O B-year O	PlayMusic
9	play a pop	play a popular chant by brian epstein	O O B-sort B-music_item O B-artist I-artist	PlayMusic
10	find fish s	find fish story	O B-movie_name I-movie_name	SearchScreeningEvent
11	book a spo	book a spot for 3 in mt	O O O O B-party_size_number O B-state	BookRestaurant
12	i need a fo	i need a forecast for jetmore massachusetts in 1 hour and 1 second from now	O O O O O B-city B-state O B-timeRange I-timeRange I-timeRange I-timeRange I-timeRange I-timeRange I-	GetWeather
13	rate this s	rate this series a 5	O B-object_select B-object_part_of_series_type O B-rating_value	RateBook
14	give me a	give me a list of movie times for films in the area	O O O O O B-object_type I-object_type O B-movie_type B-spatial_relation I-spatial_relation I-spatial_rela	SearchScreeningEvent
15	can you pl	can you play me some eighties music by adele	O O O O O B-year O O B-artist	PlayMusic
16	please let	please let me know the weather forcast of stanislaus national forest far in nine mon	O O O O O O O O B-geographic_poi I-geographic_poi I-geographic_poi B-spatial_relation B-timeRange I-tir	GetWeather
17	book a res	book a restaurant for eight people in six years	O O B-restaurant_type O B-party_size_number O B-timeRange I-timeRange I-timeRange	BookRestaurant
18	i need to l	i need to book a restaurant in fork mountain sc for valarie mari and i	O O O O O B-restaurant_type O B-city I-city B-state O B-party_size_description I-party_size_description I-p	BookRestaurant
19	add to pla	add to playlist confidence boost here comes santa claus	O O O B-playlist I-playlist B-entity_name I-entity_name I-entity_name I-entity_name	AddToPlaylist
20	book a res	book a restaurant at sixteen o clock in sc	O O B-restaurant_type O B-timeRange I-timeRange I-timeRange O B-state	BookRestaurant
21	add anoth	add another artist to the spotlight on country 2016 playlist	O O B-music_item O O B-playlist I-playlist I-playlist I-playlist O	AddToPlaylist
22	find a mo	find a movie house showing cage without a key	O O B-object_location_type I-object_location_type O B-movie_name I-movie_name I-movie_name I-mov	SearchScreeningEvent
23	add sugar	add sugarolly days to my list your favorite slaughterhouse	O B-entity_name I-entity_name O B-playlist_owner O B-playlist I-playlist I-playlist	AddToPlaylist
24	show me	show me the picture creatures of light and darkness	O O O B-object_type B-object_name I-object_name I-object_name I-object_name I-object_name	SearchCreativeWork
25	search for	search for the adventures of cookie & cream	O O B-object_name I-object_name I-object_name I-object_name I-object_name I-object_name	SearchCreativeWork
26	add this tr	add this track to my global funk	O O B-music_item O B-playlist_owner B-playlist I-playlist	AddToPlaylist
27	find anim	find animated movies nearest at a movie house	O B-movie_type I-movie_type B-spatial_relation O O B-object_location_type I-object_location_type	SearchScreeningEvent
28	play hell	play hell house song	O B-object_name I-object_name B-object_type	SearchCreativeWork
29	give this r	give this novel 5 stars	O B-object_select B-object_type B-rating_value B-rating_unit	RateBook
30	find me sh	find me showtimes for animated movies in the neighbourhood	O O O O B-movie_type I-movie_type B-spatial_relation I-spatial_relation I-spatial_relation	SearchScreeningEvent
31	find movi	find movie times	O B-object_type I-object_type	SearchScreeningEvent
32	play the t	play the top-20 best chicane songs on deezer	O O B-sort I-sort B-artist O O B-service	PlayMusic
33	i want to g	i want to give this current textbook 4 points	O O O O O B-object_select B-object_type B-rating_value B-rating_unit	RateBook
34	add manu	add manuelita to my indiespensables playlist	O B-entity_name O B-playlist_owner B-playlist O	AddToPlaylist
35	show mov	show movie schedules for douglas theatre company	O B-object_type I-object_type O B-location_name I-location_name I-location_name	SearchScreeningEvent
36	i d like to	i d like to see the show onion sportsdome	O O O O O O B-object_type B-object_name I-object_name	SearchCreativeWork
37	i d like to	i d like to go to the popular bistro in oh	O O O O O O O B-sort B-restaurant_type O B-state	BookRestaurant
38	what s the	what s the weather in my current spot the day after tomorrow	O O O O O O B-current_location I-current_location B-timeRange I-timeRange I-timeRange I-timeRange	GetWeather
39	play playl	play playlist the realest down south	O O B-playlist I-playlist I-playlist I-playlist	PlayMusic
40	add this ar	add this artist to gretchen s soul revived playlist	O O B-music_item O B-playlist_owner I-playlist_owner B-playlist I-playlist O	AddToPlaylist
41	find a vide	find a video game called victory march	O O B-object_type I-object_type O B-object_name I-object_name	SearchCreativeWork
42	find a nov	find a novel called industry	O O B-object_type O B-object_name	SearchCreativeWork
43	book a tav	book a taverna that serves vichyssoise within walking distance in oh	O O B-restaurant_type O O B-served_dish B-spatial_relation I-spatial_relation I-spatial_relation O B-state	BookRestaurant

Figure 4.2: SNIPS Dataset

The training set comprises of 13,084 utterances. Then there's the validation set and the test set contain 700 utterances each, with 100 queries per intent.

4.3 Evaluation Metrics

The measures that have been used to evaluate the performance of the models are accuracy and F1 score.

Accuracy is refers to the ratio of correct predictions to the total number of instances present in the dataset. The accuracy is given by:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \tag{4.29}$$

F1 score is defined as a weighted average of precision and recall. The formula is:

$$F1 = \frac{2.precision.recall}{precision + recall} \tag{4.30}$$

Where Precision is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{4.31}$$

The recall is given by the formula:

$$Recall = \frac{TP}{TP + FN} \tag{4.32}$$

Where in the above equations, TP stands for True Positive, FP stands for False Positive and FN stands for False Negative.

Macro-F1 is the harmonic mean between precision and recall. Here, the average is calculated per label and it's then averaged across all the labels. If p_j and r_j are the precision and recall for all $\lambda_j \epsilon$ h(x_j) from $\lambda_j \epsilon\ y_i$ the macro- f1 is,

$$Macro\ F1 = \frac{1}{Q}\sum_{j=1}^{Q}\frac{2.p_j.r_j}{p_j + r_j} \tag{4.33}$$

Micro-F1 Score is used for assessing the quality of multi-label binary problems. It measures the F1-score of the aggregated contributions of all classes. Because od perfect micro-precision and micro-recall, Micro f1-score is considered as the best value, and the worst value will be 0. Suppose, C is the Number of classes and K ϵ C

$$Precision_{micro} = \frac{\sum_{K\epsilon C} TP_k}{\sum_{K\epsilon C} TP_k + \sum_{K\epsilon C} FP_k} \tag{4.34}$$

$$Recall_{micro} = \frac{\sum_{K\epsilon C} TP_k}{\sum_{K\epsilon C} TP_k + \sum_{K\epsilon C} FN_k} \tag{4.35}$$

Micro F1-Score is defined as the harmonic mean of the precision and recall:

$$Micro\ F1 - Score = 2 * \frac{Precision_{micro} * Recall_{micro}}{Precision_{micro} + Recall_{micro}} \tag{4.36}$$

Weighted-F1 score is measured by taking the mean of all per-class F1 scores and each class's support is considered.

4.4 Performance Analysis

First, we analyze the results of the baseline models and look at how the word embeddings affect the results of those models. The performance of each model on four different word embedding models such as Simple Averaging of Word embeddings, TF-IDF Weighted Averaging of word embeddings, PV-DM Doc2Vec Training and TF-IDF and Doc2Vec Combined Feature is examined.

4.4.1 Results on ATIS dataset

The results Tables 4.1 represent Accuracy and F1 Scores of the five models that are considered for ATIS dataset.

Evaluation Metrics				
Model	Accuracy	Macro F1	Micro F1	Weighted F1
Simple RNN	0.77	0.109	0.769	0.67
GRU	0.78	0.11	0.78	0.683
BiLSTM	0.925	0.456	0.925	0.91
Logistic Regression	0.98	0.98	0.97	0.97
Naive Bayes	0.97	0.92	0.97	0.97
Linear SVM	0.978	0.93	0.978	0.978
Poly SVM	0.921	0.689	0.991	0.913
Rbf SVM	0.959	0.799	0.959	0.954
Sigmoid SVM	0.915	0.713	0.915	0.91

Table 4.1: Evaluation Metrics - ATIS

Comparing Models

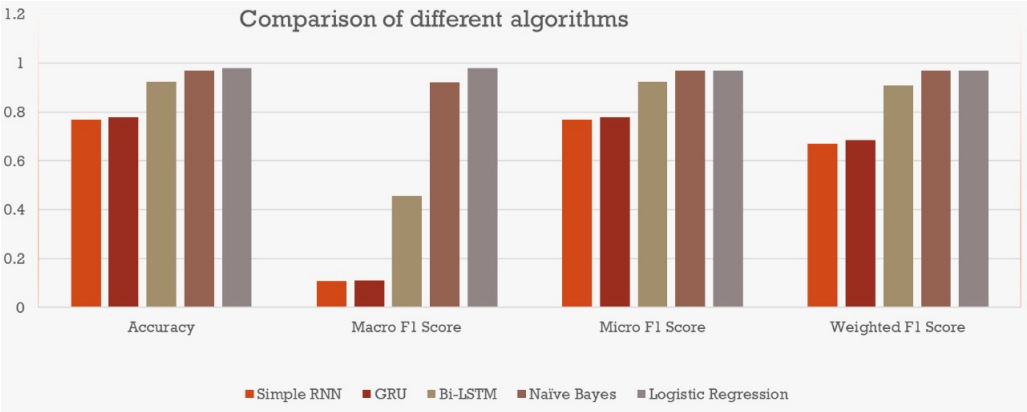


Figure 4.3: Comparing Models for ATIS Dataset

After reviewing other existing work, We saw SVM worked the best for most of them.

But For us Logistic Regression Model Worked the best and The accuracy is High.

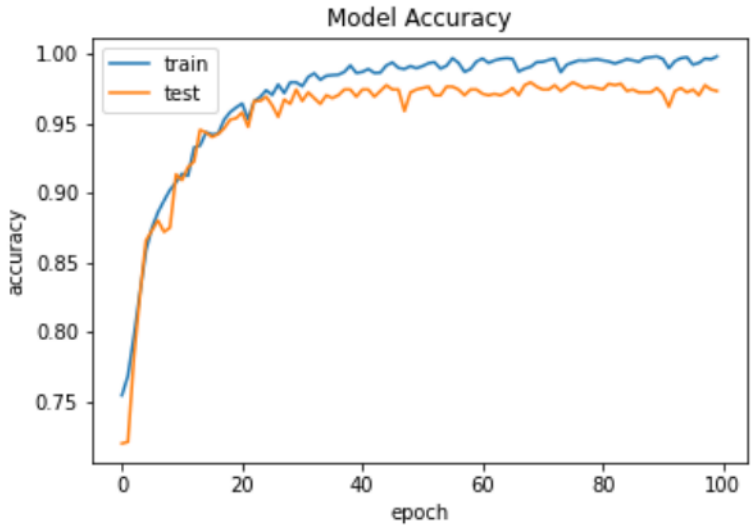


Figure 4.4: Logistic Regression Accuracy - ATIS

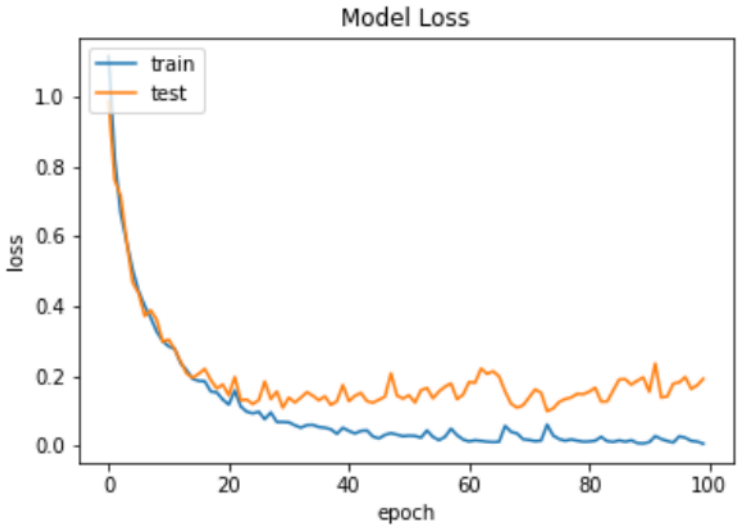


Figure 4.5: Logistic Regression Loss - ATIS

4.4.2 Results on SNIPS dataset

Table 4.2 represent Accuracy and F1 Scores of the five models that are considered for SNIPS dataset. Among all the baseline models that have been trained the results shows

Evaluation Metrics				
Model	Accuracy	Macro F1	Micro F1	Weighted F1
Simple RNN	0.97	0.97	0.97	0.96
GRU	0.14	0.035	0.14	0.04
BiLSTM	0.98	0.98	0.98	0.98
Logistic Regression	0.99	0.98	0.99	0.98

Table 4.2: Evaluation Metrics - SNIPS

Logistic Regression and Bilstm perform very good results but we will take Logistics because of its results and also it is easier to deal with when it comes to multiple classes of classification.

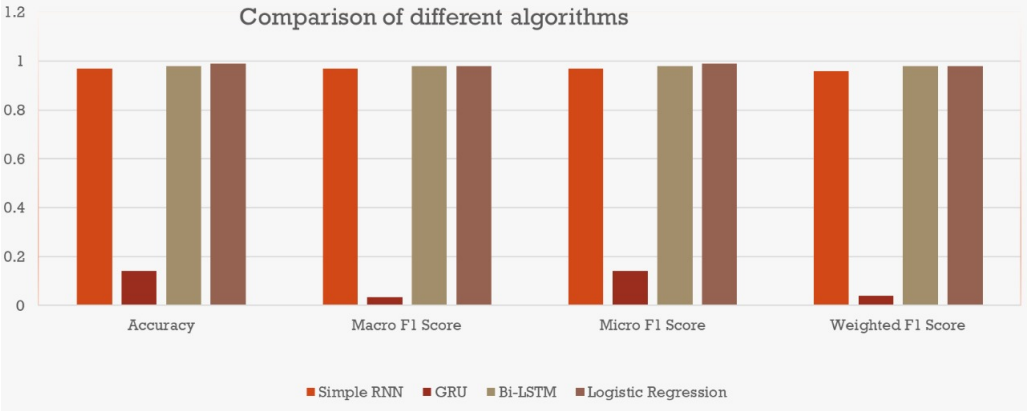


Figure 4.6: Comparing Models for SNIPS Dataset

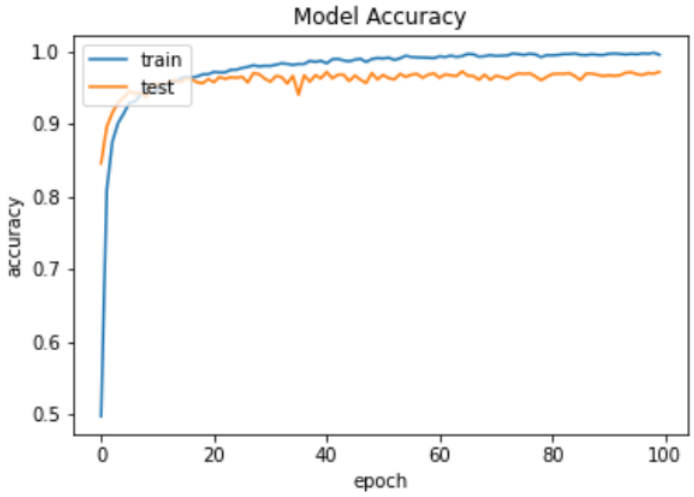


Figure 4.7: Logistic Regression Accuracy - SNIPS

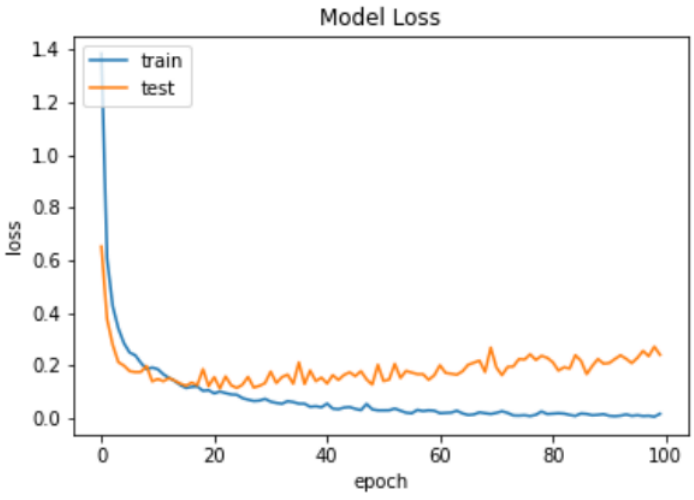


Figure 4.8: Logistic Regression Loss - SNIPS

4.5 ChatBot Result

In the figures below, the demonstration of NITT chatbot is given.

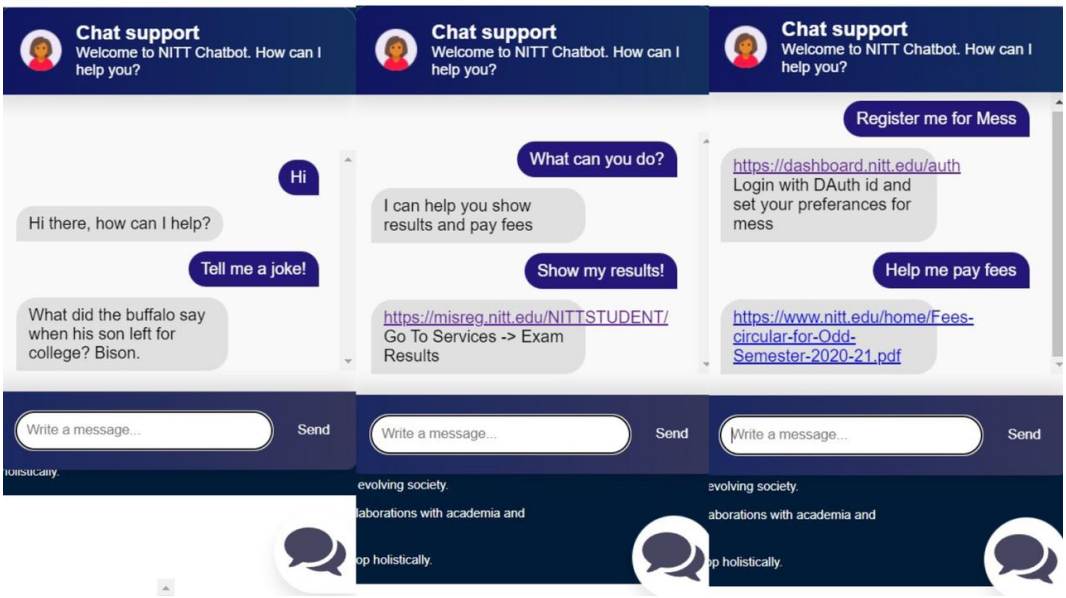


Figure 4.9: NITT ChatBot Demonstration 1

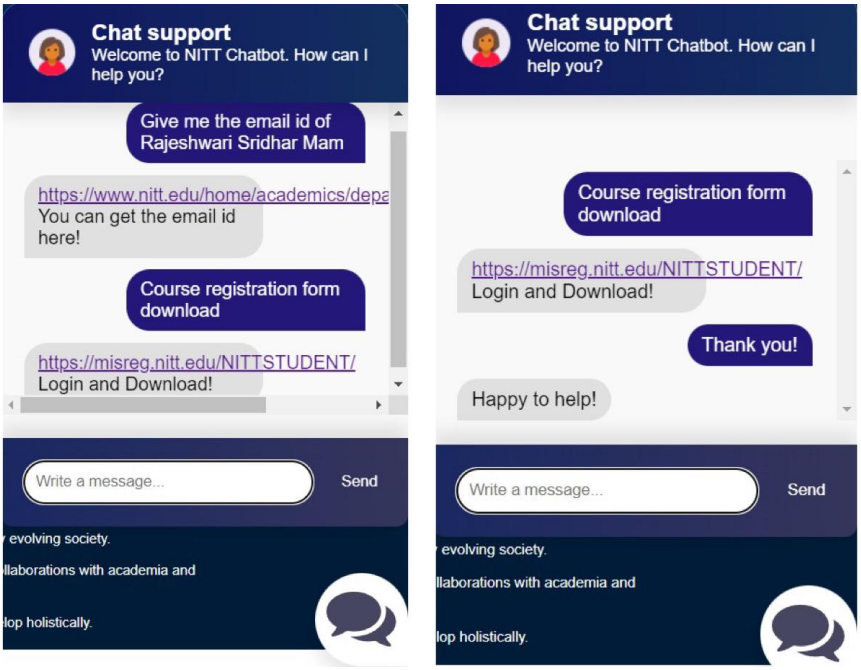


Figure 4.10: NITT ChatBot Demonstration 2

CHAPTER 5
CONCLUSION AND FUTURE WORKS

5.1 Summary

Through this project, we have investigated the efficacy of various intent classification models in conjunction with word embedding techniques to decide the best performing algorithm which is then used in the development of a chatbot. All models are examined based on their performance with two different datasets to ensure that there is no bias in the result. The Accuracy, micro F1 score, macro F1 score, and weighted F1 score are used as parameters for evaluation. We have used two approach. In one approach, We have only used Baseline Models and in another we have used BERT model. For the first approach, we implemented Naive Bayes classifiers in which the Bag-of-words approach is used to convert sentences to word embeddings. Then we used the combination of Doc2Vec method to generate word embeddings and implemented the Support Vector Machine model as well as Neural Network based classifiers such as Bidirectional Long Short Term Memory networks, Gated Recurrent Units, Recurrent Neural Network,and binary classification method Logistic Regression. For the second approach, after pre-processing of dataset, vectorization has been done, and then state-of-the-art Model-BERT-NER has been implemented. The chatbot to aid college students for their queries was developed using flask app as back-end and JS, CSS, HTML as front-end.

5.2 Conclusion

All approaches were successfully implemented and we discovered that the Logistic Regression delivers the best result for across all the datasets. It must be noted that the other approaches are not ineffective but for the purpose of single query classification, the neural network based models consume too much computing power and time and yet are marginally outperformed by logistic regression. However, as the query/text length increases, there is no doubt that the capability of the LSTM would come to the fore and the trade-off in terms of time and complexity will still justify the results. The NITT Chatbot was developed using the Flask App's Back-end and HTML, CSS, JS in front-end. It uses the BERT to classify user intent. It will re-direct the user to the required pages after taking their queries. To conclude, this Chatbot is user-oriented and beneficial in guiding students with most accurate, up to date sources of information. It is advantageous for international applicants for queries like fee payment and academic matters. Students can clear their doubts and get answers of their queries anytime at their fingertips rather than visiting college office. Thus, It improves efficiency by taking over tasks for which human assistance is not necessary.

5.3 Future Scope

In this work we have analyzed various intent classification algorithms to develop a college chat-bot. In order to deal with longer queries that require information retention, powerful neural networks are used but they exact a heavy toll on computing capabilities. Research into increasing the accuracy of classification while reducing the resource requirement for complicated questions can be done. Another possible avenue of work is increasing the usability and intents of the chat-bot. A channel for student-teacher communication can be implemented; that will make the student-teacher communication smooth and protected. Student feedback option can be added. According to the feedback's, more training data will be inserted. Some of the new feauters that can be added are- 1) Speech recognition by which students can use their voice to get answers, 2) Language Setting- Students can change the language according to their mother tonhue and get a smooth experience without any language barrier, 3) Direct integration with services like Course Enrollment, fees payment and so on.

APPENDIX

1 Baseline Models

```
import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
import re
import sklearn
from sklearn import svm
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score
nltk.download("stopwords")
nltk.download("punkt")



import pandas as pd
import numpy as np
df = pd.read_csv("atis_intents_train.csv")
df.head()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =
↪  0.20, random_state = 99)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
model.score(X_test,y_test)

train_X, val_X, train_Y, val_Y = train_test_split(sentences, intent,
↪  shuffle = True, test_size = 0.1)
```

2 State of the Art Models

```
!pip install simpletransformers

import pandas as pd
data = pd.read_csv('/content/ner_dataset.csv',encoding="latin1" )

data =data.fillna(method ="ffill")

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data["Sentence #"] = LabelEncoder().fit_transform(data["Sentence #"]
 ↪  )

data.rename(columns={"Sentence
 ↪  #":"sentence_id","Word":"words","Tag":"labels"}, inplace =True)

data["labels"] = data["labels"].str.upper()

X= data[["sentence_id","words"]]
Y =data["labels"]

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size
 ↪  =0.2)

#building up train data and test data
train_data =
 ↪  pd.DataFrame({"sentence_id":x_train["sentence_id"],"words":x_trai["words"],"
test_data =
 ↪  pd.DataFrame({"sentence_id":x_test["sentence_id"],"words":x_test["words"],"la

train_data

from simpletransformers.ner import NERModel,NERArgs

label = data["labels"].unique().tolist()
label

args = NERArgs()
args.num_train_epochs = 1
args.learning_rate = 1e-4
args.overwrite_output_dir =True
args.train_batch_size = 32
```

```
args.eval_batch_size = 32
#args.use_cuda=False

# import torch
# print(torch.cuda.is_available())

model = NERModel('bert', 'bert-base-cased',labels=label,args =args)

model.train_model(train_data,eval_data =
    test_data,acc=accuracy_score)

result, model_outputs, preds_list = model.eval_model(test_data)

prediction, model_output = model.predict(["What is the new name of
    Bangalore"])

prediction
```

PACKAGES AND LIBRARIES USED

Pandas

Pandas is used for data manipulation, providing high-performance, easy-to-use data structures and data analysis tools. It is often used as it contains data structures and operations for performing operations on time series and numerical tables.

NumPy

NumPy is used to work with large multi-dimensional arrays and matrices and provides and extensive set of mathematical functions to work with these two constructs. It is one of the key libraries for scientific computing in Python.

NLTK

NLTK or Natural Language Toolkit, as the name suggests, is a set of libraries and programs to work human language processing in Python. It comes with the useful functionality of generating graphs for results and is widely used.

OS

Functions for interacting with the operating system are provided by the OS module. It allows you to interface with the underlying operating system that Python is running on, thus, providing a way of using operating system dependent functionality.

IO

The IO module deals with inputs and outputs. It provides Python interfaces to stream handling. It also manages the file-related input and output operations.

Matplotlib

Matplotlib is a comprehensive library containing visualization tools, for embedding plots into applications using general-purpose GUI toolkits. It is used to create static, animated, and interactive visualizations in Python.

spaCy

spaCy is a library that is used for natural language processing. It has a convolutional neural network backend for for part-of-speech tagging, dependency parsing, text categorization etc. It is primarily used to create software for production usage.

TensorFlow

TensorFlow is a relatively new library for machine learning that was developed by Google. It has a flexible architecture for easy deployment and focuses on training and inference of deep neural networks. It also provides APIs for easy execution.

Scikit-learn

Scikit-learn features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, kmeans and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It also features word embedding functions.

Genism

Gensim is a python library that was developed to extract semantic information from the given input. It is the state-of-the-art package used for working with the word vector models and consists of unsupervised machine learning algorithms where no human intervention is required. It can process raw unstructured data and create a semantic network out of it. The input is a corpus of plain texts and it can manage large inputs efficiently by not loading the whole file in the memory.

REFERENCES

[1] Charles T. Hemphill, John J. Godfrey, and George R. Doddington. "The ATIS Spoken Language Systems Pilot Corpus". In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*. 1990. URL: https://aclanthology.org/H90-1021.

[2] Jian Hu et al. "Understanding User's Query Intent with Wikipedia". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: Association for Computing Machinery, 2009, pp. 471–480. ISBN: 9781605584874. DOI: 10.1145/1526709.1526773. URL: https://doi.org/10.1145/1526709.1526773.

[3] Mark Kröll and Markus Strohmaier. "Analyzing Human Intentions in Natural Language Text". In: *Proceedings of the Fifth International Conference on Knowledge Capture*. K-CAP '09. Redondo Beach, California, USA: Association for Computing Machinery, 2009, pp. 197–198. ISBN: 9781605586588. DOI: 10.1145/1597735.1597780. URL: https://doi.org/10.1145/1597735.1597780.

[4] Edward Loper Steven Bird Ewan Klein. *Natural Language Processing with Python*. O'Reilly, 2009.

[5] Yongli Zhang. "Support Vector Machine Classification Algorithm and Its Application". In: *Information Computing and Applications*. Ed. by Chunfeng Liu, Leizhen Wang, and Aimin Yang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 179–186. ISBN: 978-3-642-34041-3.

[6] Jey Han Lau and Timothy Baldwin. "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation". In: (2016). DOI: 10.

48550/ARXIV.1607.05368. URL: https://arxiv.org/abs/1607.05368.

[7] Alice Coucke et al. *Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces*. 2018. DOI: 10.48550/ARXIV.1805.10190. URL: https://arxiv.org/abs/1805.10190.

[8] Lian Meng and Minlie Huang. "Dialogue Intent Classification with Long Short-Term Memory Networks". In: *Natural Language Processing and Chinese Computing*. Ed. by Xuanjing Huang et al. Cham: Springer International Publishing, 2018, pp. 42–50. ISBN: 978-3-319-73618-1.

[9] Shahzad Qaiser and Ramsha Ali. "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents". In: *International Journal of Computer Applications* 181 (July 2018). DOI: 10.5120/ijca2018917395.

[10] Shuo Xu. "Bayesian Naïve Bayes classifiers to text classification". In: *Journal of Information Science* 44.1 (2018), pp. 48–59. DOI: 10.1177/0165551516677946. eprint: https://doi.org/10.1177/0165551516677946. URL: https://doi.org/10.1177/0165551516677946.

[11] Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. "Joint Multiple Intent Detection and Slot Labeling for Goal-Oriented Dialog". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 564–569. DOI: 10.18653/v1/N19-1055. URL: https://aclanthology.org/N19-1055.

[12] Vraj Desai; Sidarth Wadhwa; Anurag A; Bhisham Bajaj. "Text-Based Intent Analysis using Deep Learning". In: *International Journal of Innovative Science and Research Technology* 5 (2020), pp. 267–274. DOI: 10.38124/IJISRT20JUL342.

[13] Jetze Schuurmans and Flavius Frasincar. "Intent Classification for Dialogue Utterances". In: *IEEE Intelligent Systems* 35.1 (2020), pp. 82–88. DOI: 10.1109/MIS.2019.2954966.

[14] Alberto Benayas et al. “Unified Transformer Multi-Task Learning for Intent Classification With Entity Recognition”. In: *IEEE Access* 9 (2021), pp. 147306–147314. DOI: 10.1109/ACCESS.2021.3124268.

[15] Geetanjali Bihani and Julia Taylor Rayz. “Fuzzy Classification of Multi-intent Utterances”. In: *NAFIPS*. 2021.

[16] Tzu-Yu Chen et al. “Multi-Modal Chatbot in Intelligent Manufacturing”. In: *IEEE Access* 9 (2021), pp. 82118–82129. DOI: 10.1109/ACCESS.2021.3083518.

[17] Zixian Feng et al. “An Evaluation of Chinese Human-Computer Dialogue Technology”. In: *Data Intelligence* 3.2 (June 2021), pp. 274–286. ISSN: 2641-435X. DOI: 10.1162/dint_a_00090. eprint: https://direct.mit.edu/dint/article-pdf/3/2/274/1963473/dint_a_00090.pdf. URL: https://doi.org/10.1162/dint%5C_a%5C_00090.

[18] Manoj Kumar et al. “ProtoDA: Efficient Transfer Learning for Few-Shot Intent Classification”. In: *CoRR* abs/2101.11753 (2021). arXiv: 2101.11753. URL: https://arxiv.org/abs/2101.11753.

[19] Tulika Saha et al. “A Unified Dialogue Management Strategy for Multi-Intent Dialogue Conversations in Multiple Languages”. In: *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 20.6 (Sept. 2021). ISSN: 2375-4699. DOI: 10.1145/3461763. URL: https://doi.org/10.1145/3461763.

[20] Anant Saraswat, Kumar Abhishek, and Sheshank Kumar. “Text Classification Using Multilingual Sentence Embeddings”. In: *Evolution in Computational Intelligence*. Ed. by Vikrant Bhateja et al. Singapore: Springer Singapore, 2021, pp. 527–536. ISBN: 978-981-15-5788-0.

Plagiarism Report

Below is the plagiarism report of our thesis. The report has been taken using Turnitin.

MULTI-INTENT CLASSIFICATION IN CHATBOTS

ORIGINALITY REPORT

17%	11%	13%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Soft Computing and Signal Processing", Springer Science and Business Media LLC, 2021 Publication	3%
2	"Evolution in Computational Intelligence", Springer Science and Business Media LLC, 2021 Publication	2%
3	www.arxiv-vanity.com Internet Source	2%
4	blog.vsoftconsulting.com Internet Source	2%
5	paperswithcode.com Internet Source	1%
6	aclanthology.org Internet Source	1%
7	regenerativetoday.com Internet Source	1%
8	www.coursehero.com Internet Source	1%

Figure 5.1: Plagiarism Report