Intent Classification: A Comparative Analysis

Thesis by

Niranjan Dalai

Class Roll No.: 001910504009

University Registration No:149844 OF 2019-20

Examination Roll No: M6TCT22011

Under the guidance and
supervision of

Dr. Sudip Kumar Naskar
Associate Professor

Department of Computer Science and Engineering Faculty of Engineering and Technology

In Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

JADAVPUR UNIVERSITY

Kolkata, West Bengal, India

2022

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

To Whom It May Concern

I hereby forward the thesis entitled "Intent Classification: A Comparative Analysis" prepared by NIRANJAN DALAI (University Registration No: 149844 OF 2019-2020, Examination Roll No: M6TCT22011) under my guidance and supervision. It is a bona-fide piece of work that may be accepted in partial fulfillment of the requirement for awarding the degree of Master of Computer Technology in the Faculty of Engineering and Technology, Jadavpur University, Kolkata

Countersigned

……
Dr. Sudip Kumar Naskar
(Thesis Supervisor)
Associate Professor
Department of Computer Science and Engineering
Jadavpur University, Kolkata-32

…….
Prof. Anupam Sinha
Head, Department of Computer Science and Engineering,
Jadavpur University, Kolkata-32.

…….
Prof. Chandan Mazumdar
Dean, Faculty of Engineering and Technology,
Jadavpur University, Kolkata – 32.

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Certificate of Approval*

This is to certify that the thesis entitled "Intent Classification: A Comparative analysis" prepared by NIRANJAN DALAI is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that, by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only for the purpose for which it has been submitted.

……..
Signature of Examiner 1:
Date:

Final Examination for evaluation of the thesis.

……..
Signature of Examiner 2:
Date:

***Only in case the thesis is approved**

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis entitled "**Intent Classification: A Comparative analysis"** contains literature survey and original research work by undersigned candidate, as part of her Master of Computer Technology studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

NAME : NIRANJAN DALAI

ROLL NUMBER : 001910504009

EXAMINATION ROLL NO : M6TCT22011

REGISTRATION NUMBER : 149844 OF 2019-2020

THESIS TITLE : INTENT CLASSIFICATION : A COMPARATIVE ANALYSIS

SIGNATURE WITH DATE :

Abstract

Understanding the user's purpose is a critical stage in any human-computer interaction in dialogue systems. Typically, different classifiers are used to classify natural language sentences into specified intent groups. Intent Classification is the task of comprehending and classifying the intents based on the text presented. The goal of this thesis is to classify the intentions based on the provided texts. We initially trained the texts in the datasets using a variety of methods, including Classical Text Classification models like Linear Support Vector Machine, Linear Regression, Random Forest, and Multinomial Naïve Bayes and deep learning methods like Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and the state-of-the-art model, BERT (Bidirectional Encoder Representations from Transformers) and compared the results.

In this thesis, we have used three different datasets which are SNIPS Natural Language Understanding Benchmark (SNIPS), Banking77, and ATIS Airline Travel Information System. We have individually trained and tested all three of the datasets on the above-given models and we have got results based on it.

Contents

Chapter 1: Introduction

1.1 Background

Languages are the most fundamental and fundamental feature of practically everything in the cosmos. Without language, humans, animals, birds, and insects cannot communicate with one another. Everything in our cosmos need the ability to communicate its needs and feelings to other beings. Researchers have been wondering how machines and humans would communicate since the start of computer science.

This is demonstrated famously in Alan Turing's Artificial Intelligence (AI) test, sometimes known as the Turing test. The Turing test involves a conversation between a computer and a human, with the human being unaware that they are conversing with a machine, then we can deem the computer a thinking machine (Turing, 1950).Like Turing, many researches have pondered the question. But the bigger question is "Will the machine on the other end be able to interpret the question asked by the human and give a correct answer".

Natural Language Processing (NLP), an area of artificial intelligence, includes Intent Classification. Natural language processing and analysis (NLP) is concerned with computers processing and analysing natural language, that is, any language that has evolved naturally rather than artificially, such as computer coding languages. The automatic categorization of text data based on client goals is known as intent classification.

In essence, an intent classifier analyses texts and sorts them into categories such as HR, Finance, R&D, and so on. This is useful for deciphering client intentions, automating procedures, and gaining valuable insights. Every customer connection has a goal, or intention, at the end of the day. To boost client retention, loyalty, and happiness, it should respond fast whether they wish to make a purchase, request more information, or unsubscribe. It's not enough to figure out what the customer wants using effective intent classification. Bottom line, the ability to interpret what a customer wants regardless of how they phrase their request (including typos and poor grammar) is critical to efficiently resolving customer issues.

1.2 Motivation

Everyone wants to communicate their thoughts and feelings to others, but can computers understand what we're trying to say? Human language is the most complicated, with thousands of ways to express one's emotions and feelings, including typos, poor grammar, and, most importantly, sarcasm, which has taken overthe world of social media. Chatbots are amazing creations in the area of NLP since they allow users to talk about their concerns in real time.

Instead of waiting for a live human agent to address the same problem as the chatbot, a chatbot or chatterbot is a software application that conducts an on-line chat conversation using text or text-to-speech. However, the chatbot's key issue is to first

comprehend the language presented and then classify the text based on the intents, which is the essence of intent classification.

The major goal of this thesis is to determine the most effective technique to operate on any form of multi-label dataset and correctly classify text based on intents.

1.3 Dataset Overview

Intent classification is similar to a simple text classification but it has remarkable differences. This thesis is basically a comparison between different classification techniques and models. Let's see the overview of the dataset.

1.3.1 ATIS Airline Travel Information System

The ATIS (Airline Travel Information Systems) dataset contains audio recordings and hand transcripts of humans asking automated airline travel enquiry systems for flight information. There are 22 distinct intent types in the data. In the train and test sets as depicted in Table 1.3.1.1, there are 4834, 500, and 800 intent-labelled reference utterances, respectively. We have obtained the dataset the website of Kaggle[1]. The First few values of the dataset is provided in Table 1.3.1.2

Dataset	Total Values
Training Set	4834
Test Set	800

Table 1.3.1.1 The table depicts how dataset is divided into Training and Testing dataset.

Text	Intent
i want to fly from boston at 838 am and arrive in denver at 1110 in the morning	atis_flight
what flights are available from pittsburgh to baltimore on thursday morning	atis_flight
cheapest airfare from tacoma to orlando	atis_airfare

Table 1.3.1.2: The table depicts first 3 values of the dataset

1. https://www.kaggle.com/hassanamin/atis-airlinetravelinformationsystem/

1.3.2 BANKING77 Dataset

The BANKING77 dataset contains a very fine-grained set of banking intents. Itcontains 13,083 customer service inquiries with 77 different intentions. It focuses on single-domain intent detection at a finer level. The text basically corresponds to a string feature. Intent corresponds to the classification labels (0-76) corresponds to unique intents. The First few values of the dataset is provided in Table 1.3.2.1

Dataset	Total Values
Train Values	10003
Test Values	3080

Table 1.3.2.1 The table depicts how dataset is divided into Training and Testing datasets

Let's look at how the datasets look like in table 1.3.2.1,

Text	Category
I am still waiting on my card?	card_arrival
Can you change my currency to EUR?	fiat_currency_support
I tried to get cash out of the ATM but it is taking too long	pending_cash_withdrawal

Table 1.3.2.2 The table depicts first 3 values of the dataset

1.3.3 SNIPS Natural Language Understanding Benchmark (SNIPS)

Snips' built-in-intents dataset was first used to compare different voice assistants before being provided as a public dataset under the folder 2016-12-built-in-intents at Github[1].The SNIPS dataset has 7 intents. The training set contains 13784 sentences and the test set contains 700 sentences. The First few values of the dataset is provided in Table 1.3.3.2

Dataset	Total Values
Train Values	13784
Test Values	700

Table 1.3.3.1 The table depicts how dataset is divided into Training and Testing datasets

Text	Intent
listen to westbam alumb allergic on google music	PlayMusic
add step to me to the 50 clásicos playlist	AddToPlaylist
i give this current textbook a rating value of 1 and a best rating of 6	RateBook

Table 1.3.3.2 The table depicts first 3 values of the dataset

Website Link: https://github.com/sonos/nlu-benchmark

Chapter 2: Literature Overview

2.1 Natural Language Processing

Natural Language Processing, or NLP, is a branch of Artificial Intelligence that allows robots to read, understand, and interpret human languages. It's a field that focuses on the interface of data science with human language, and it's growing in popularity across a variety of industries. Today, NLP is thriving as a result of massive advancements in data access and computational capacity, allowing practitioners to obtain substantial outcomes in fields such as healthcare, media, finance, and human resources, among others.

In simple words, NLP refers to the automatic processing of natural human language such as speech or text, and while the notion is intriguing, the actual value of this technology is found in its applications. NLP can assist us with a wide range of activities, and the fields of application seem to be expanding on a daily basis. Based on electronic health records and the patient's own voice, NLP allows for disease recognition and prognosis. This skill is being investigated in a variety of medical problems, including cardiovascular disease, depression, and possibly schizophrenia. Amazon Comprehend Medical, for example, is a service that extracts disease conditions, drugs, and treatment outcomes from patient notes, clinical trial reports, and other electronic health information using natural language processing (NLP). By discovering and extracting information from sources such as social media, businesses can learn what customers are saying about a service or product. This sentimentresearch can reveal a lot about a customer's preferences and decision-making processes. An IBM researcher created a cognitive assistant that functions like apersonalised search engine, learning everything about you and then reminding you of names, songs, and other things you can't remember when you need it.

Companies like Yahoo and Google use NLP to filter and classify your emails by analyzing emails as they pass through their systems, preventing spam from reaching your inbox. The MIT NLP Group developed a new algorithm to determine whether a source is accurate or politically biased, whether a news source can be trusted or not, to aid in the detection of fake news. Amazon's Alexa and Apple's Siri are two examples of intelligent voice-driven interfaces that employ NLP to respond to vocal commands and perform things like locating a specific store, locating weather forecasts, recommending the best route to work, and turning on the lights at home.

Financial traders can benefit greatly by knowing what is going on and what people are talking about. NLP is being used to track news, reports, and comments regarding potential company mergers, all of which can subsequently be put into a trading algorithm to create large gains. Remember the old adage: "Buy the rumour, sell the news." NLP also being utilised in the search and selection phases of talent recruiting, recognising potential employees' skills and spotting candidates before they go on the job market. Powered by IBM Watson NLP technology, LegalMation developed a

platform to automate routine litigation tasks and help legal teams save time, drive down costs and shift strategic focus.

2.2 Related Work

One branch of NLU is intent classification. Intent classification is the process of mapping user utterances to predefined meanings known as intents, such as a "Hello" greeting or a "Will it rain today?" weather query. The resource specified by the affective, cognitive, or situational goal expressed in a Web search engine interaction is known as user intent. According to Belkin's [20] states of a searching episode, intent is similar to goal, and expression is similar to method of interaction. In contrast to goals, intent is concerned with how the goal is expressed because the expression determines the type of resource the user wants to address his or her overall goal. User intent is a broad concept that has been studied in a variety of domains. We will begin by reviewing previous research on user intent in various domains. In most applications, user intent is task-specific and may facilitate further information mining in the back-end system, such as named entity extraction. "Play Arijit Singh's song Humnawa," for example. This user input could have the intent "PlaySong," and there are several task-specific slots, such as "AddToPlaylist" and "RemovePlaylist." Hollerit et al. [21] investigated commercially motivated tweets. They begin with a definition of a tweet with commercial intent as A tweet contains at least one verb and describes the user's intention to engage in a recognisable commercial activity. They also take note of another aspect of intent, namely whether a tweet's intent is explicit orimplicit. The tweet "Facing Repossession, Let us buy your house for cash now" expresses the explicit intention to buy a house. In contrast, the tweet "Debating on buying a pair of 80s cop shades..." contains some commercial intent, but it is implied rather than stated explicitly as a possibility in the future. They also recognise that implicit commercial intent has commercial value.

On Twitter data, the paper [22] proposed a semi-supervised user intent classification task. They begin by defining "intent tweet," "intent-indicator," and "intent-keyword." They define an intent tweet as one that contains at least one verb and expresses the user's intent to perform an activity in a recognisable manner. They also define the terms "intent-indicator" and "intent-keyword." Intent-Indicator: It is a collection of terms that users use to express their intentions. It is a verb or infinitive phrase that follows a subject word, such as "I." In the tweet "I want to reach New Delhi," for example, "want to" is an intent-indicator, indicating that the tweet is likely to be an intent tweet. Intent-Keyword: It is a noun, verb, multi-word verb, or compound noun (consisting of several nouns) contained in a verb or noun phrase that immediately follows an intent-indictor, for example, "reach" and "New Delhi" are intent-keywords in the phrase "reach New Delhi."

Characterizing and identifying the intent of user queries in the domain of user intent mining in information retrieval is one of the most important challenges of modern information retrieval systems [13]. Broder [11] proposed the taxonomy of user intent as navigational, informational, and transactional, which is the first classification found in the literature. He claimed that the "need behind the query" on the web is frequently

not informational. The goal of "Navigational queries" is to get to a specific site that you already know about. The goal of "Information queries" is to find information that is assumed to be on the webpage. And "transactional queries" are searches for resources to be used in subsequent actions, such as downloading or streaming. Broder classified queries using a user survey and manual classification of a query log. This work was later taken up by Rose and Levinson [22], who developed a framework for manual classification of search goals by extending the classes proposed by Broder. The paper [23] extracts a dataset from Yahoo! search logs by randomly sampling 2,492 user sessions. This data set contains 3,658 queries and 18,296 documents. A professional editor examined the user activity in each user session and judged the userintents. Their work centred on shopping-related questions. This work considers five shopping intents: buying guide, reviews, support, official product page, and shopping site/purchase. In their scope, the consumer first consults buying guides to learn how toshop for the desired product, then consults reviews and visits the official product homepage. Finally, the consumer makes a purchase through a shopping site andconsults support pages for post-purchase information. In Paper [8], we performed query log entity mining using user intent understanding. They generate a training dataset by automatically aligning some selected freebase entity types to the queries in order to evaluate the model. And their testset is created by meticulous human labelling. However, such a simple model is frequently insufficient for capturing the complexities of real-world information seeking, and search engines require a better understanding of user intent and its multi-dimensional nature. As a result, facets with the most representative abilities are chosen in order to provide a more comprehensive understanding of the user and his/her intents. The following aspects were investigated: genre, objective, specificity, scope, topic, task, authority sensitivity, spatial sensitivity, and time sensitivity [17].

Recently, there has been a focus on various types of neural networks (NN), both for feature extraction and classification. Word embeddings are a more recent feature representation technique that has proven successful for many text classification tasks [24]. However, it has only been used sparingly in intent classification. Enhanced GloVe word embeddings are used as input to a bidirectional long short-term memory network (BiLSTM) [25] and FastText word embeddings are used as input to a convolutional neural network (CNN) [26]. In [27], both word2vec and GloVe word embeddings were used in conjunction with three different CNN architectures for intent classification and other text classification tasks, with both embedding types producing comparable results. Paper [28] compared multiple types of NNs on two different binary intent classification tasks: flight vs. all other intents for the ATIS dataset, and web search vs. all other intents for a much larger in-house Microsoft Cortana dataset. They tried both word embeddings and word hashing and found that LSTM and networks with gated recurrent units produced the best overall results (GRU).

In several papers, intent classification is combined with slot filling and, in some cases, domain detection. Although less flexible and does not allow for domain- and intent-specific features, this has the advantage of only requiring one model for all domains and being able to benefit from common domain and intent-specific features

[29]. CNN [32], recursive neural networks (RecNN) [30], bidirectional recurrentneural network (RNN) with LSTM [29], RNN[1] with gated recurrent units (GRU) [12], attention-based RNN [33], and slot-gated attention-based RNN [32] have all been used in joint modelling. The authors of [7] employed the pre-trained language representation model BERT. On the Snips and ATIS datasets, their model outperformed the models in [33], [32], and [34] in both slot-filling and intent classification.

Posting short messages via social network services (e.g., Facebook, Twitter) has become an essential part of many users' daily lives. Online social networks have evolved into major platforms for users to discuss their needs and desires through online activities such as chatting with friends and posting short status updates. Severalprevious works have concentrated on extracting and inferring commercial intents from Twitter users. Hollerit and Kol[21] use a binary commercial intent classification to identify tweets with explicit and implicit commercial intent. They discover a number of discriminative patterns that users may employ when expressing purchasingor selling intent. Other papers[35] have focused on extracting users' explicit purchase intent from tweets in order to facilitate product recommendation for e-commerce. They begin by generating a list of candidate tweets using seed words such as "buy," "purchase," and "on sale." The tweets are then classified using textual features from the tweets as well as user demographic information.

2.3 Text Processing

Text pre-processing is a technique for cleaning text data and preparing it for use in a model. Text data comprises noise in the form of emotions, punctuation, and text in a different case, among other things. When it comes to Human Language, there are many different ways to communicate the same thing, and this is only the beginning of the issue. Machines cannot understand words; they want numbers; thus, we must convert text to numbers efficiently. Pre-processing our text can be done in a variety ofways.

2.3.1 Lowercasing

Although often overlooked, one of the simplest and most effective forms of text preparation is lowering the case of our text data. It works for most NLP problems and can aid in circumstances when our dataset considerably improves predicted output consistency. Lowercasing the terms in all of our datasets was really helpful in achieving uniformity across all of our approaches.

2.3.2 Stemming

The process of eliminating inflection in words (for example, disturbed, difficulties) to their root form is known as stemming (e.g., trouble). In this situation, the root could be a canonical variant of the original word rather than a true root word.

Stemming employs a rudimentary heuristic that chops off the ends of words in the hopes of appropriately changing them into their base form. As a result, the terms problem, troubled, and troubles could be spelled troubl instead of trouble.

For stemming, there are a variety of algorithms. Porter's Algorithm is the most often used algorithm, and it is also recognised to be experimentally effective for English.

2.3.3 Lemmatization

On the surface, lemmatization resembles stemming, in which the goal is to remove inflections and map a word to its root form. The only difference is that lemmatization makes an effort to do it correctly. It doesn't simply cut things off; it also converts words to their root form. The term "better," for example, would be mapped to "good." For mappings, it might employ a dictionary like WordNet or other unique rule-based techniques.

2.3.4 Stopword Removal

Stop words are a group of words that are frequently employed in a language. Stop words in English include "a," "the," "is," "are," and others. The idea behind stop words is that by deleting low-information terms from a document, we can concentrate on the crucial words.

If your search question is "What is the meaning of intent classification?" in the context of a search system, we want the search system to prioritise surfacing content that discuss intent classification over publications that discuss what is. This can be accomplished by blocking the analysis of all terms from our stop word list.

In all our datasets, which is basically consists customer complaints and asks about their respective matters there are different lot of words which are inconsequential suchas “is”, ”the”, “of” which does not makes ultimate importance in the text had to be removed so that when it's used in sequence-to-sequence modelling, it is easy for them to handle and often times when encoding has to be done these words are inconsequential as they are ultimately tokenized and they do not create an impact on the result.

2.3.5 Tokenization

Tokenization is the process of breaking down a text into a list of tokens, which can include words, sentences, characters, numbers, punctuation, and more. Tokenization has two key advantages: the first is that it significantly reduces search time, and the second is that it makes efficient use of storage space.

The process of mapping sentences from characters to strings and strings to words is the first step in any NLP problem since we need to comprehend the meaning of any text or document by analysing the words/sentences existing in the text to understand it.

2.3.6 Bag of Words

To pre-process text and extract all of the features from a text document for use in deep learning, we employ the bag of words technique. It's also a representation of any text that explains or elaborates on the occurrence of words in a corpus (document). It's also known as "Bag" because of its mechanism, which simply cares if known words are in the document, not where they appear.

Let's take an example to understand bag-of-words in more detail. Like below, we are taking 2 text documents:

"Sunil was happy for Ganesh and he was angry on Paresh."

"Paresh love animals."

Above you see two corpora as documents, we treat both documents as a different entity and make a list of all the words present in both documents except punctuations as here,

"Sunil", "was", "happy", "for", "Ganesh", "and", "he", "Paresh", "love", "animals"

Then we create these documents into vectors (or we can say, creating a text into numbers is called vectorization in ML) for further modeling.

Presentation of "Sunil was happy for Ganesh and he was angry on Paresh" into vector form as [1,1,1,1,1,1,1,0,0] , and the same as in, "Paresh love animals" having vector form as [1,0,0,0,0,0,0,0,1,1]. So, the bag-of-words technique is mainly used for featuring generation from text data.

Each document is represented as a word-count vector in the bag of words model. These counts can be binary (a word may be in the text or not) or absolute (a word mayappear in the text or not). The amount of elements in the vocabulary determines the size of the vector. The bag of words will be a sparse matrix if the majority of the components are zero.

Because we will be working with a large amount of training data in deep learning, we will have a sparse matrix. Sparse representations are more difficult to model for both computational and informational reasons.

2.4 Term Frequency-Inverse Document Frequency

The text vectorizer Term frequency-inverse document frequency converts the text into an useable vector. Term Frequency (TF) and Document Frequency (DF) are combined in this idea (DF). The term frequency refers to the number of times a term appears in a document. The frequency of a term in a document reflects how essential it is. Term frequency depicts each text in the data as a matrix with the number of documents in the rows and the number of distinct terms in the columns.

Document frequency is the number of documents containing a specific term. Document frequency indicates how common the term is.

Inverse document frequency (IDF) is the weight of a term, it aims to reduce the weight of a term if the term's occurrences are scattered throughout all the documents. IDF can be calculated as follow:

$$idf_i = \log(n/df_i)$$

Where idf_i is the IDF score for term i, df_i is the number of documents containing term i, and n is the total number of documents. The higher the DF of a term, the lower the IDF for the term. When the number of DF is equal to n which means that the term appears in all documents, the IDF will be zero, since log(1) is zero, when in doubt just put this term in the stopword list because it doesn't provide much information.

The TF-IDF score as the name suggests is just a multiplication of the term frequency matrix with its IDF, it can be calculated as follow:

$$W_{i,j}=tf_{i,j} \times idf_i$$

Where w_i is TF-IDF score for term i in document j, tf_i is term frequency for term i in document j, and idf_i is IDF score for term i.

2.5 Text Encoding

Text encoding is a method of converting meaningful text into numbers or vectors while preserving the context and relationships between words and sentences, such that a machine can recognise the pattern associated with any text and grasp the context of sentences.

Converting categorical values to numerical values can be done in a variety of ways. Each strategy has its own set of trade-offs and consequences on the feature set. I'd want to concentrate on two key strategies here: Label-Encoder and One-Hot-EncodingBoth of these encoders are part of the SciKit-learn toolkit, and they are used to translate text or categorical data into numerical data, which the model expects and with which it performs better.

We have used the Label encoder on all of the datasets when we were performing the classification by CNN. In label encoding in Python, we replace the categorical value with a numeric value between 0 and the number of classes minus 1. In the SNIPS dataset there are categorical variable values containing 7 distinct classes, we use (0, 1,2, 3,4, 5 and 6).

This approach is very simple and it involves converting each value in a column to a number. We have a column named intent-types having values.

That's all there is to label encoding. Label encoding, on the other hand, can introduce a new challenge depending on the data values and type of data because it uses number sequencing. The issue with employing numbers is that they introduce

comparison/relationship between them. Although there appears to be no correlation between the various purpose categories, one can mistakenly believe that one type has a higher priority than another based on the numbers. The algorithm could be misled into believing that the data has some sort of hierarchy or structure.

2.6 Word Embeddings

A numeric vector input that represents a word in a lower-dimensional space is known as Word Embedding[2] or Word Vector. It permits words with comparable meanings to be represented in the same way. They can also make educated guesses about meaning. A 50-valued word vector can express 50 distinct features.

Word embeddings[2] are a technique for extracting features from text and putting them into a machine learning model that works with text data. They make an effort to keep syntactic and semantic information intact. The word count in a sentence is used by methods like Bag of Words(BOW), CountVectorizer, and TFIDF, but no syntactic or semantic information is saved. The size of the vector in these algorithms is equal tothe number of entries in the vocabulary. If the majority of the elements are 0, we can get a sparse matrix. Large input vectors will result in a large number of weights, requiring a lot of computation during training. Word Embeddings[2] provide ananswer to these issues.

If embedding finds a good relationship between works like for an example

King-Man+Women= Queen

Chapter 3: Classification Models

In this section we are going to discuss theory behind different models that were used in this project.

3.1 Text Classification Models

Text classification is a machine learning technique that assigns a set of predefined categories to texts, in our case text utterances. Text classifiers can be used to organize, structure, and categorize pretty much any kind of text – from documents, medical studies and files, and all over the web.

There are a lot of different classifiers and feature representation techniques that have been successfully used for text classification and that could be of interest for intent classification. This project focuses on comparing different classifiers on all the three datasets, in this section we are going to discuss the theory behind different methods.

3.1.1 Support Vector Machine

A supervised machine learning algorithm is the support vector machine (SVM)[36]. The goal of the SVM algorithm is to find a hyperplane in an N-dimensional space that categorises data points clearly. The hyperplane's size is determined by the number of features. If there are only two input characteristics, the hyperplane is just a line, as given in Fig.3.1.1.1. When the number of input features reaches three, the hyperplane transforms into a two-dimensional plane. It locates the hyperplane that maximises the margin between the two classes, which is defined as two times the perpendicular distance between the hyperplane and the nearest data points. The dividing hyperplane is the solid line, and the margin is the distance between the two dashed lines in this simple two-dimensional example. As given in below Fig 3.1.1.1, The three highlighted dots represent the data points closest to the separating line, which are known as support vectors.

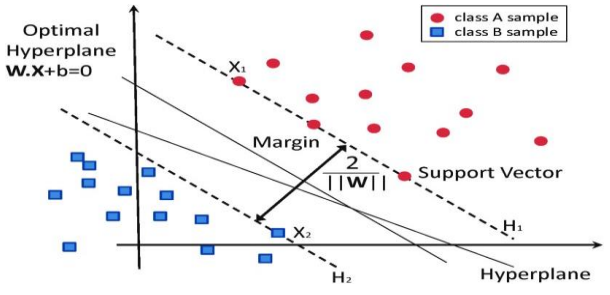

Fig3.1.1.1: The maximum margin hyperplane (the solid line) and support vectors (the three highlighted points) for a 2-dimensional classification task.

(Source:https://www.researchgate.net/figure/Classification-of-data-by-support-vector-machine-SVM_fig8_304611323)

So, for this sort of data, SVM finds the greatest margin, as it has for previous data sets, and then adds a penalty each time a point crosses the margin. In these instances, the margins are referred to as soft margins. When the data set contains a soft margin. A common penalty is the loss of a hinge. There will be no hinge loss if there are no infractions. If a violation is proportionate to the distance of the violation, the loss is proportional to the distance of the violation.

SVM can be used with any type of vector that encodes any type of data. This means that in order to take advantage of SVM text classification's capability, texts must be converted into vectors. Vectors are numerical lists that represent a set of coordinatesin a given space.

As a result, when SVM calculates the decision boundary we described earlier, it decides where to draw the best "line" (or best hyperplane) that divides the space into two subspaces: one for vectors that belong to the specified category and one for vectors that do not.

Although the SVM was designed to be a binary classifier, there are numerous methods for expanding it to multi-class classification. One-vs-one and one-vs-rest are two popular strategies. A data point is classified as belonging to the class that receivesthe most votes from these SVMs, with K being the number of classes. We train one SVM for each class in one-vs-rest, with the negative class having all the data points from the other K 1 classes. After that, a data point is categorized as belonging to the class with the highest SVM output value. The one-vs-rest technique is preferable when there are a lot of classes. However, this approach has the problems of using unbalanced training datasets for each SVM, and that the output values might not be onthe same scale and therefore not completely comparable.

3.1.2 Random Forest

Random Forest is a frequently used supervised learning technique for categorization tasks. It constructs decision trees from data samples, then extracts predictions from each of them, and ultimately votes on the best answer. It's an ensemble method that's superior than a single decision tree because it averages the results to reduce over-fitting.

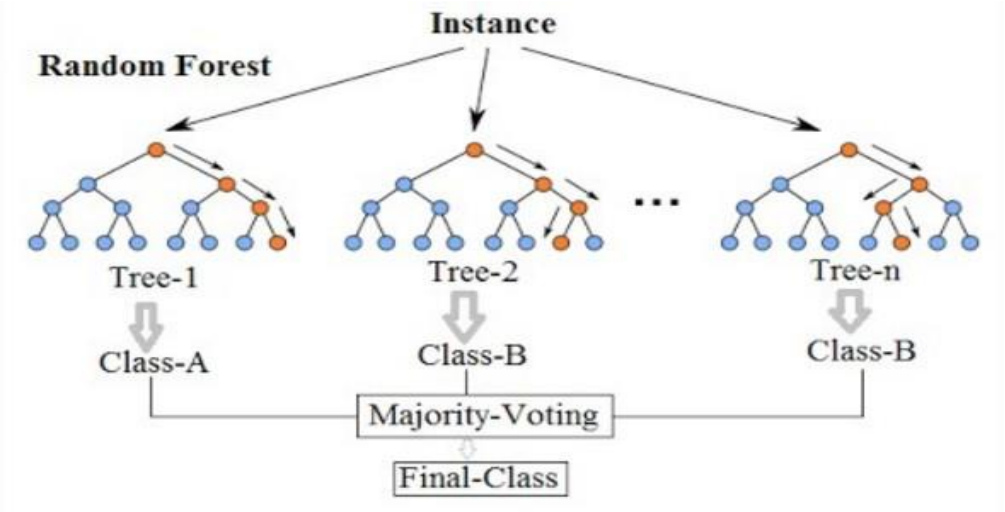

Fig 3.1.2.1 The figure depicts how random forest classifies the data using multiple decision trees and finding the class most suitable for the data.

(Source:https://towardsdatascience.com/machine-learning-basics-random-forest-classification-499279bac51e)

As given in Fig: 3.1.2.1, when the Random Forest classifier is given a dataset. Each decision tree receives a portion of the dataset. Random forest selects n number of records at random from a data collection of k records. For each sample, an individual decision tree is built. There will be an output from each decision tree. Majority Voting is used to evaluate the final result.

3.1.3 Logistic Regression

By default, logistic regression is limited to two-class classification tasks. Some extensions, such as one-vs-rest, can be used to solve multi-class classification issues with logistic regression, but they require the classification problem to be split into many binary classification problems first.

To predict the outcome, logistic regression employs a sigmoid function. The sigmoid function returns a number between 0 and 1. In general, we use a threshold like 0.5. If the sigmoid function returns a value greater than or equal to 0.5, we consider it to be 1, and if it returns a value less than 0.5, we consider it to be 0.

By default, logistic regression cannot be utilised for multi-class classification tasks, which have more than two class labels. Instead, it needs to be tweaked to accommodate multi-class categorization difficulties.

Splitting the multi-class classification problem into numerous binary classification problems and fitting a standard logistic regression model to each sub problem is a typical strategy for adapting logistic regression to multi-class classification problems. One-vs-rest and one-vs-one wrapper models are examples of this type of technique.

3.1.3 Multinomial Naïve Bayes

The nave multinomial Bayes is a popular method for categorizing documents based on statistical analysis of their contents as portrayed in Fig 3.1.3.1. The classification seeks to allocate text fragments to classes by calculating the likelihood that a text belongs to the same class as other texts with the same subject. This algorithm is based on the Bayes theorem, and it calculates the probability of an event occurring based on prior knowledge of event conditions. It's based on the formula below :

$$P(A|B) = P(A) * P(B|A)/P(B)$$

When predictor B is already available, we calculate the likelihood of class A.
P(B), where B is a class, is the prior probability of B.
P(A) is the prior probability of class A, where A denotes the characteristics.
P(B|A) = probability of predictor B occurring given class A.

Each text contains a number of words (i.e. terms) that help the reader grasp the material. A class is a grouping of one or more texts that all refer to the same subject.

The statistical analysis is used to classify texts with one of the existing classes, evaluating the hypothesis that the terms in a text have already appeared in other texts from that class. This raises the likelihood that a text belongs to the same class, as given in Fig 3.1.3.1.

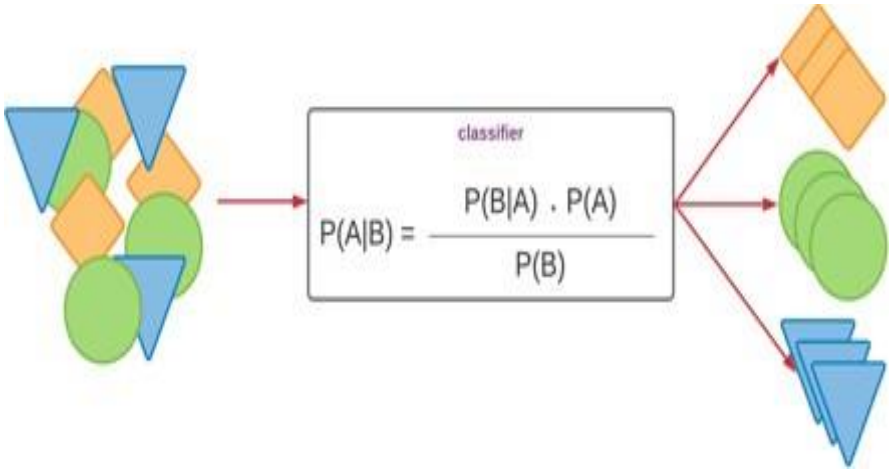

Fig 3.1.3.1: The above figure depicts the calculation of likelihood of each sentence and using the above formula, it classifies the data into respective classes.

(Source:https://hands-on.cloud/implementing-naive-bayes-classification-using-python/)

3.2. Deep Learning models

3.2.1 Recurrent Neural Network

A recurrent neural network (RNN)[37] is a form of artificial neural network that is designed to operate with time series or sequence data. Ordinary feed forward neural networks are only designed to handle data items that are unrelated to one another. We need to alter the neural network to incorporate the dependencies between these data points because we have data in a sequence where one data point depends on the preceding data. RNNs feature a concept of memory,' which allows them to store the states or information of prior inputs in order to construct the sequence's next output.

We must encode the words into vectors because plain text cannot be utilised in a neural network. In most cases, the vocabulary includes all English words. It is for this reason that word embeddings are required.One of the allure of RNNs is the possibilityof connecting earlier knowledge to the current activity.

When the current step has some form of relationship with the prior steps, RNNs are designed to employ sequential data. This makes them appropriate for natural language processing and applications with a time component (audio, time-series data). RNNs work effectively in applications where sequential information is critical, because without it, the meaning could be misconstrued or the grammar could be erroneous.

Sometimes all we need to do is glance at recent data to complete the task at hand. Consider a language model that tries to anticipate the next word based on the ones that came before it. We don't need much more information to guess the last word in "the clouds are in the sky" — it's very evident that the following word is going to be sky. RNNs can learn to use past knowledge in situations where the distance between relevant information and the location where it's needed is narrow.

Unfortunately, as the gap widens, RNNs lose their ability to learn to connect the dots. RNNs, in theory, are perfectly capable of dealing with such "long-term dependencies." To tackle toy issues of this type, a human could carefully selectsettings. Regrettably, RNNs do not appear to be capable of learning them in practise.

A straightforward technique for modelling sequence is to use one RNN to map the input sequence to a fixed-sized vector, then send the vector to a softmax layer for classification or other tasks [37]. Unfortunately, components of the gradient vector might expand or decay exponentially over lengthy sequences when using RNNs with transition functions of this type during training [11]. The RNN model struggles to learn long distance correlations in a sequence because of the difficulty with burstingor vanishing gradients. [11] proposed the long short-term memory network (LSTM) to explicitly address the problem of learning long-term dependencies.

3.2.2 LSTM Networks

Long Short Term Memory[16] networks – commonly referred to as "LSTMs" — are a type of RNN that can learn long-term dependencies. Hochreiter & Schmidhuber(1997) introduced them, and numerous individuals developed and popularised them in subsequent work. They are currently frequently utilised and function exceptionally effectively on a wide range of situations. The long-term reliance problem is explicitly avoided by LSTMs[16]. They don't have to work hard to remember knowledge for lengthy periods of time; it's nearly second nature to them. All recurrent neural networks are made up of a series of repeated neural network modules. This repeating module in ordinary RNNs will have a relatively simple structure, such as a single tanhlayer. A single layer is present in the repeating module of a conventional RNN. LSTMs have a chain-like structure as well, but the repeating module is different. Instead of a single neural network layer, there are four, each of which interacts in a unique way. An LSTM's repeating module is made up of four layers that interact with one another.

The Embedding layer is the first layer in the model definition portion. It uses a dense vector format to represent words. When a word is employed, its position in the vector space is determined by the words that surround it. The size of the vocabulary is provided. An LSTM layer is the next layer. The sentences are intertwined, and each one is the same length. The activation function is a widely used rectified linear activation function. With return sequence set to True, any other appropriate activation function can be used; this is a crucial parameter when utilising several LSTM layers since it allows the output of the previous LSTM layer to be utilised as an input to the next LSTM layer. If it is not set to true, the input will not be passed to the next LSTM layer. A dropout layer is used to regulate the network and maintain it as free of bias asfeasible. The output layer, the Dense layer, includes n cells that reflect the n various categories in this scenario. Depending on the number of categories, the number can beadjusted. Adam optimizer and sparse categorical cross entropy were utilised. The Adam optimizer is the best optimizer for sparse gradients and noisy issues right now. When the classes are mutually exclusive, i.e. when each sample belongs to exactly one class, the sparse categorical cross entropy is commonly utilised. Now that the model is complete, it's time to train the data by separating it into dependent and independent datasets.

The LSTM model is then trained with the training data and validated with the test data. As a result, the LSTM model performed similarly for all three datasets.

3.2.3 Convolutional Neural Networks

In a neural network, neurons are supplied inputs, and the weighted total of those inputs is considered by the neurons, who then pass it through an activation function and provide the output to the next neuron as given in Fig 3.2.3.1.

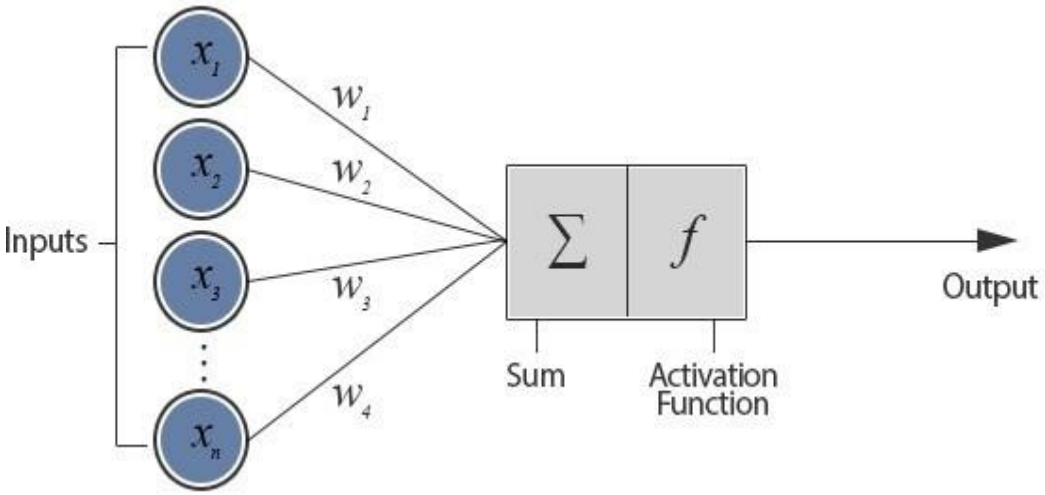

Fig 3.2.3.1 The figure depicts how neurons are fed as inputs which calculates the weighted sum and pass it by an activation function and passes out the output to next neuron.

(Source: https://miro.medium.com/max/589/0*7BDHwjsLAkg-IxEg.png)

Because it functions over a volume of inputs, a convolutional neural network[18] differs from a neural network. Each layer searches the data for a pattern or helpful information. An image with pixels as the input vector and RGB as the three input channels representing the channel is an example of multi-channel input.

The architecture of a CNN[19] is given in Fig 3.2.3.3. Before we begin our objective of intent classification, we must first learn some concepts in the construction of convolutional neural networks. Convolution is a mathematical process that combines two relationships to create a third. Combines two sets of data.

Convolution over input: We use convolution to extract features from input data by applying a filter/kernel (both can be used interchangeably). This is crucial while extracting features. There are some settings for the sliding filter, such as how much input to take at once and how much input should be overlapped.

Stride: The step filter's size changes with each instance of time.

Filter count: Number of filters we want to use.

Once the filter has been applied to the input and numerous feature maps have been formed, an activation function is applied to the output to provide a non-linear connection for our output. As shown in Fig 3.2.3.2, we usually add padding around input to prevent the feature map from shrinking. If we don't include padding, feature maps that are larger than the number of input items will begin to shrink, and relevant information beyond the boundaries will be lost.

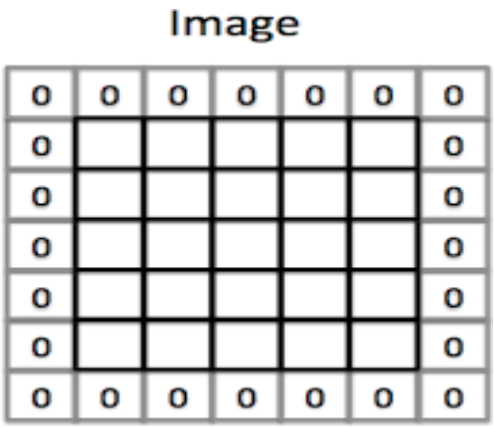

Fig 3.2.3.2 The figure depicts how padding is done on features to get a specified input length

(Source:https://miro.medium.com/max/216/0*2nJaNoqf38GtNPgW.png)

It also enhances performance by ensuring that the filter size and stride are appropriate for the input. We're not finished yet. We need something to help us reduce the amount of computation in the CNN while also avoiding overfitting the data. Overfittingcauses the model to memorise rather than learn from the training data.

We utilise a pooling layer in between the convolutional layers to reduce dimensional complexity while retaining the convolutions' significant information.

The max pooling layer is an example. As seen in Fig 3.2.3.3, it identifies the pool's maximum and passes it to the next layer.

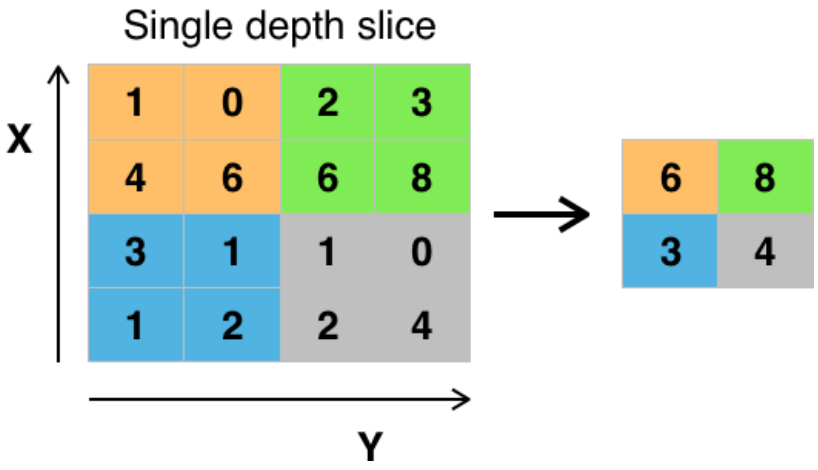

Fig:3.2.3.3 The figure depicts the max pooling layer. It identifies the pool's maximum and passes it to the next layer

(Source : https://miro.medium.com/max/570/0*iGJ2k5a-jS_bH9G6.png)

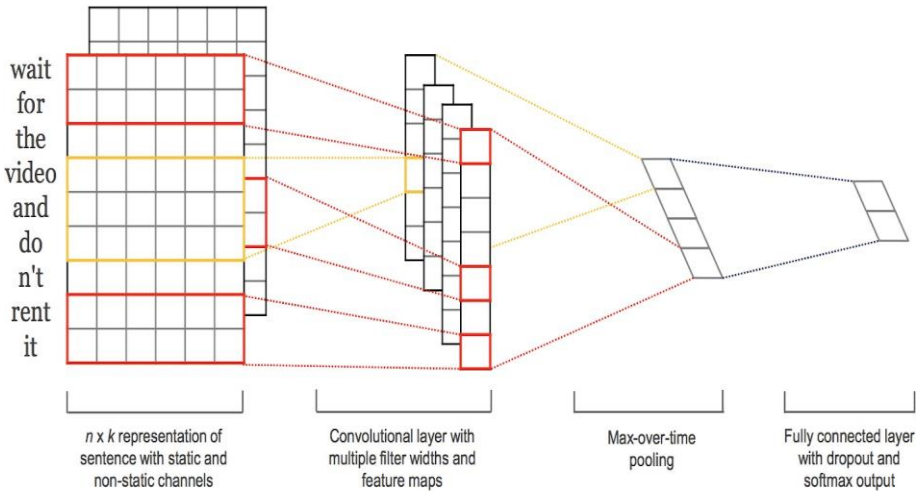

Fig 3.2.3.4 CNN architecture for classifying texts

(Source : https://miro.medium.com/max/616/0*zGBFInybPXCUDsfp.png)

We used to encode our text as a vector and run the algorithm on that vector when using Naive Bayes, but now we need to consider the similarity of terms in different reviews because this will help us look at the review as a whole rather than focusing onthe influence of each individual word.

We employ a library-provided pre-defined word embedding. If the data isn't embedded, there are a variety of embeddings to choose from.

Even though vectors representing text are of the same class, their dot product may return zero, but if you do a dot product of those embedded word vectors to identify similarity between them, you will be able to find the interrelation. The Max Pooling layer then slides the filter/ kernel across these embeddings to discover convolutions, which are then further dimensionally reduced to reduce complexity and processing.

Finally, the completely connected layers and the activation function on the outputs will provide values for each class.

Each word is converted into a fixed-length dense vector with the help of a dimension embedding layer. The vocabulary size is used as the input dimension. As a result, each word in the input will be represented by a vector of the embedding size. A total of two convolutional layers (Conv1D) were used. MaxPooling1D is a max-pooling layer. Max Pooling is a CNN operation that selects the most significant element from the input region covered by the filter/kernel. Pooling minimises the output's dimensions while preserving the most relevant information.

The ultimate probability of belonging to each of the classes are determined by the dense layer and softmax activation. Softmax activation is employed here since it works best with categorical cross-entropy loss, which is what we'll use to train the model using.

Categorical cross-entropy loss and the rmsprop optimizer are used to build the model. Categorical cross-entropy is a loss function that is frequently used in multi-class classification problems. The rmsprop optimizer is a gradient-based optimization technique that uses a moving average of squared gradients to normalize the gradient. This helps to overcome the vanishing gradients problem.

3.2.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a new publication by Google researchers. It has created a stir in the Machine Learning community by presenting cutting-edge results in a wide range of NLP tasks.

The application of the bidirectional training of Transformer, a popular attention model, to language modelling is BERT's[9] fundamental technical novelty. In contrastto earlier research, which looked at a text sequence from left to right or a combinationof left-to-right and right-to-left training, this study looked at a text series from left to right. The findings of the paper suggest that bidirectionally trained language models can have a better sense of language context and flow than single-direction language models.

BERT[10] makes use of Transformer, an attention mechanism that learns contextual relations between words in a text. Transformer includes two separate mechanism — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

The Transformer encoder reads the complete sequence of words at once, unlike directional models that read the text input sequentially (left-to-right or right-to-left). As a result, it is classified as bidirectional, however it is more correct to describe it as non-directional. This property enables the model to deduce the context of a word fromits surrounds (left and right of the word).

The input consists of a series of tokens that are embedded into vectors before being processed by the neural network. The result is an H-dimensional series of vectors, each of which corresponds to an input token with the same index.

The task of defining a prediction target when training language models is difficult. A directive strategy that limits context learning by definition. BERT employs two training methodologies to meet this challenge:

Masked LM (MLM)

15 percent of the words in each word sequence are substituted with a [MASK] token before being fed into BERT[11], in Fig 3.2.4.1. Based on the context provided by the other, non-masked words in the sequence, the model then attempts to predict the original value of the masked words. In technical terms, output word prediction necessitates:

On top of the encoder output, a classification layer is added.

Transforming the output vectors into the vocabulary dimension by multiplying them by the embedding matrix.

Softmax is used to calculate the likelihood of each word in the vocabulary.

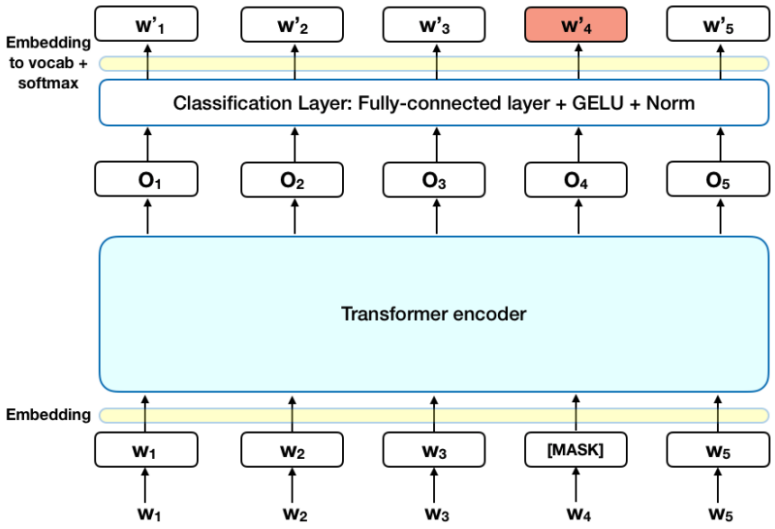

Fig 3.2.4.1 The figure depicts the working of Masked LM

(Source" https://miro.medium.com/max/1400/0*ViwaI3Vvbnd-CJSQ.png)

Only the prediction of masked values is taken into account by the BERT loss function, which ignores the prediction of non-masked words. As a result, the model converges more slowly than directional models, however this is countered by its higher context awareness. Note that the BERT implementation is slightly more involved in practise, and it does not replace all of the 15% masked words.

Next Sentence Prediction (NSP)

The BERT training approach involves feeding the model pairs of sentences andlearning to predict whether the second sentence in the pair is the next sentence in the original document. During training, 50 percent of the inputs are a pair in which the second sentence is the following sentence in the original text, while the other 50 percent are a random sentence from the corpus. The random sentence will, it is assumed, be detached from the first sentence.

Before entering the model, the input is processed in the following fashion to assist the model distinguish between the two sentences in training:

A [CLS] token is placed at the start of the first sentence, and a [SEP] token is placed at the end. Each token has a sentence embedding that indicates whether it is Sentence A or Sentence B. Sentence embeddings are similar to token embeddings in concept, but with a vocabulary of two as seen in Fig. 3.2.4.2

Each token is given a positional embedding to denote its place in the sequence. The Transformer paper[11] explains the concept and implementation of positional embedding as given in Fig 3.2.4.2

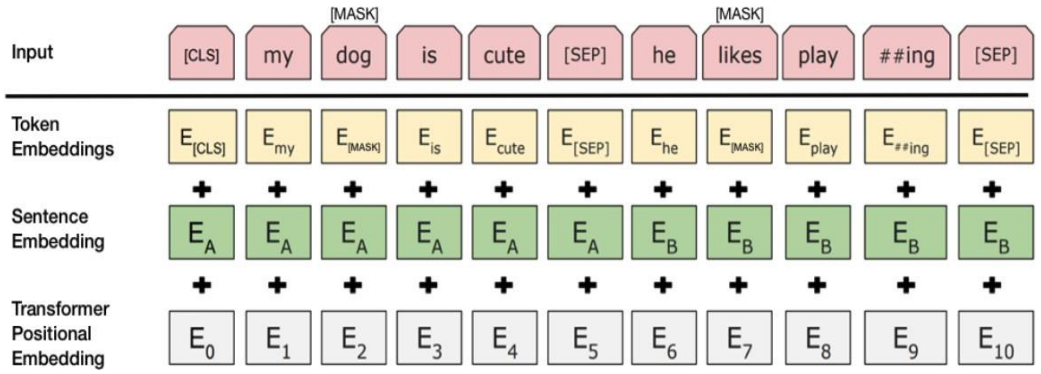

Fig 3.2.4.2 The figure depicts token embeddings, sentence embeddings and positional embeddings

(Source:https://miro.medium.com/max/1400/0*m_kXt3uqZH9e7H4w.png)

Using a simple classification layer, the [CLS] token's output is transformed into a 21 shaped vector (learned matrices of weights and biases).

Softmax is used to calculate the likelihood of IsNextSequence. Masked LM and Next Sentence Prediction are coupled for training the BERT model, with the goal of minimising the combined loss function of the two techniques.

Chapter 4: EXPERIMENTAL RESULTS

4.1 Evaluation Metrics

Predicting class labels from input data is what classification is all about. There are just two possible output classifications in binary categorization (i.e., Dichotomy). Multiclass categorization allows for the presence of more than two possible classes, which is what our thesis is all about.

Let's look at the parameters we'll use to judge our performance before moving on to the experimental results.

There are four types of outcomes that could arise while making categorization predictions:

True Positive (TP): When you forecast that an observation belongs to a particular class and it actually does.

True Negative (TN): When you forecast that an observation does not belong to a class, and it actually does not belong to that class.

False Positive (FP): When you forecast an observation but it doesn't happen.

False Negative (FN): When you predict an observation does not belong to a class and it actually does belong to that class.

Predicted value / Actual Value	**Positive(1)**	**Negative(0)**
Positive(1)	**TP**	**FN**
Negative(0)	**FP**	**TN**

Table 4.1.1 The above table depicts the confusion matrix for the different outcomes of the experimental results

Accuracy

The most basic categorization metric is accuracy. It's really simple to comprehend. And it's well-suited to both binary and multiclass classification problems.

$$\text{Accuracy} = (TP+TN)/(TP+FP+FN+TN)$$

The proportion of true results among the total number of cases studied is known as accuracy.

For classification problems that are well balanced and not skewed or have no class imbalance, accuracy is a suitable choice of evaluation.

Precision

Precision, which answers the following question: What percentage of the expected Positives actually turn out to be Positive?

$$\text{Precision} = (TP+FP)/(TP+FP)$$

When we want to be absolutely certain of our prediction, precision is a good statistic to use. For example, if we're developing a system to forecast whether or not we should reduce the credit limit on a specific account, we need to be absolutely certain about our prediction, otherwise we risk causing consumer discontent.

Recall

Another relevant metric is recall, which answers a different question: what percentage of actual Positives is identified correctly?

$$\text{Recall} = (TP)/(TP+FN)$$

Recall is a valid choice of evaluation metric when we want to capture as many positives as possible. For example: If we are building a system to predict if a person has cancer or not, we want to capture the disease even if we are not very sure.

F1 Score

The F1 score is the harmonic mean of precision and recall and is a value between 0 and 1.The harmonic mean is utilised because, unlike ordinary averages, it is not sensitive to extremely large numbers. If we have a model with a precision of 1 and a recall of 0, the simple average will be 0.5 and the F1 score will be 0. When one of the factors is low, the F1 score is no longer affected by the second. Classifiers with equivalent precision and recall are favoured by the F1 score. If you want to strike a compromise between precision and recall, the F1 score is a better metric to utilise.

$$F1 = 2*(\text{precision}*\text{recall})/(\text{precision}+\text{recall})$$

4.2 Discussion and Results

Now let us see the experimental data that we have obtained.

4.2.1 LSTM

The first layer of the lstm is the Embedding layer which represents words using a dense vector representation. The next layer is an LSTM layer. The sentences are embedded and every sentence is made of the same length. The activation function we have used is a rectified linear unit, which is widely used. A dropout layer is used for regulating the network and keeping it as away as possible from any bias.

The final Dense layer is the output layer which has n cells representing the n different categories in this case. We have used Adam optimizer and sparse categorical crossentropy. Adam optimizer is the current best optimizer for handling sparse gradients and noisy problems. The sparse categorical crossentropy is mostly used when the classes are mutually exclusive, ie, when each sample belongs to exactly one class.

In this model, we see that all the models especially BANKING77 performed really well given its number of classes, as the BANKING77 dataset has highest number of classes the precision comes out to be pretty well. Apart from this dataset, SNIPS and ATIS dataset has performed really well for LSTM model but as for the ATIS dataset especially the results we got are very good but being a highly imbalanced data we cannot predict its actual results.

	Accuracy	Precision	Recall	F1-Score
SNIPS	98.31	98.67	98.08	98.37
ATIS	95.47	97.42	93.72	95.52
BANKING	86.57	89.39	81.71	85.33

Table 4.2.1.1 The table depicts the result obtained for all three datasets using LSTM

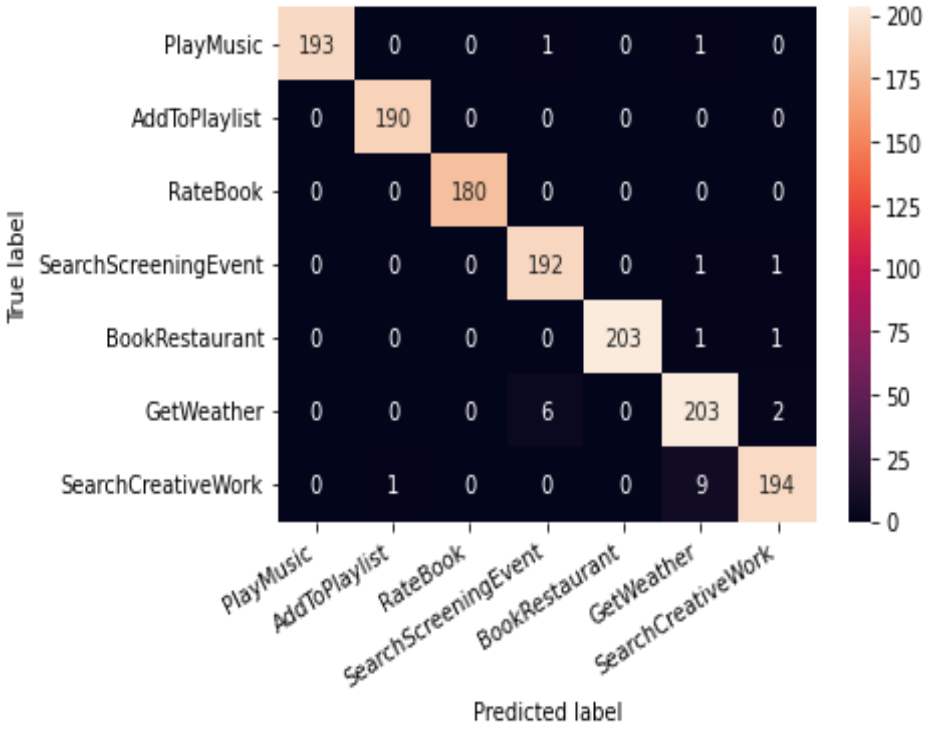

Fig 4.2.1.1: Confusion Matrix for SNIPS Dataset for LSTM Model

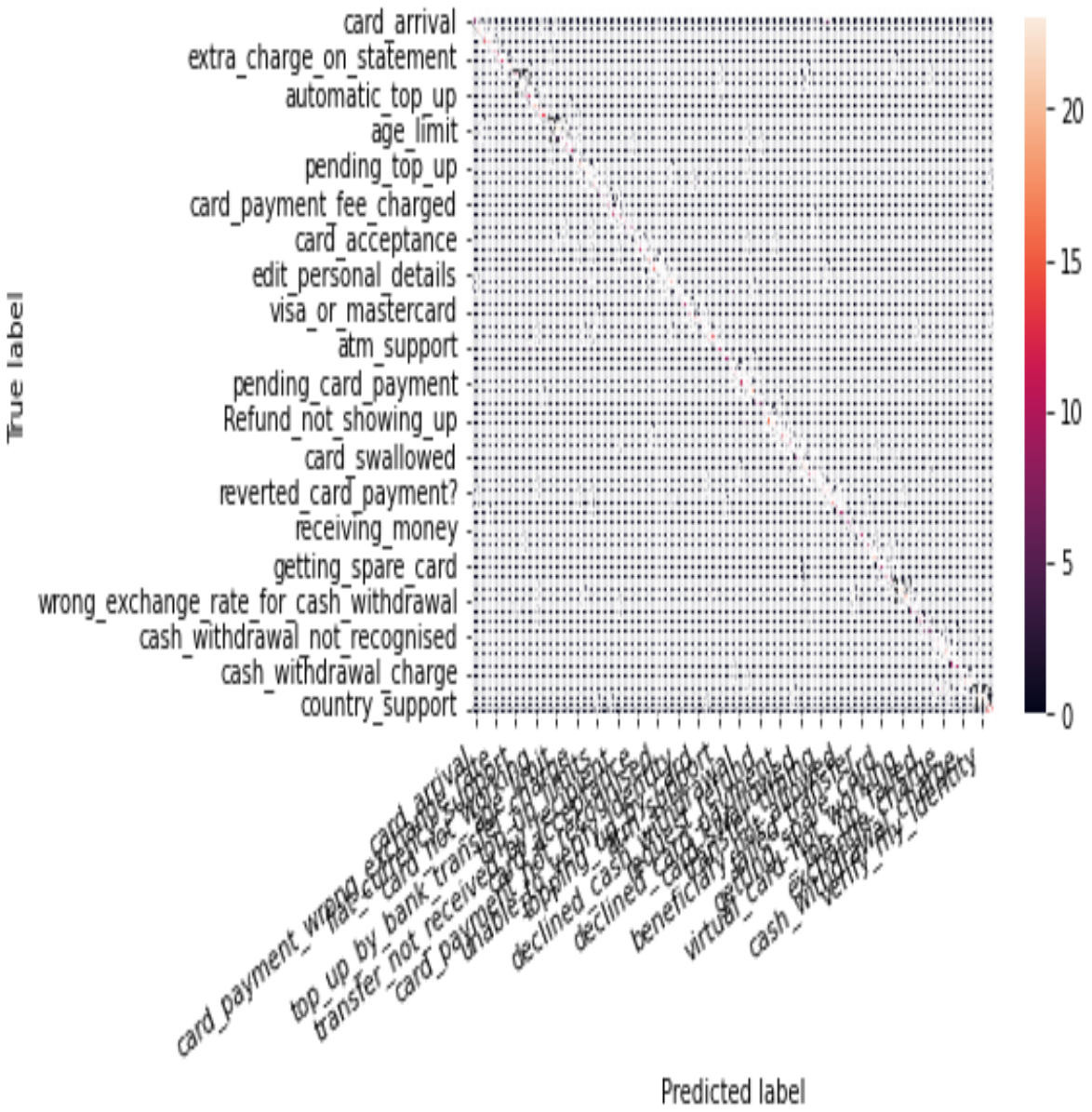

Fig 4.2.1.2: Confusion Matrix for Banking Dataset for LSTM Model

4.2.2 CNN

We have used an embedding layer which helps to convert each word into a fixed-length dense vector . The input dimension is set as the size of the vocabularyand the output dimension . Each word in the input will hence get represented by a vector of fixed size.Two convolutional layers (Conv1D) are used with respective filters each, a kernel size , and relu activation.A max-pooling layer(MaxPooling1D) which is an operation that selects the maximum element from the region of the input which is covered by the filter/kernel. Pooling reduces the dimensions of the output,but it retains the most important information.A flatten layer to flatten the inputwithout affecting batch size. A dense (fully connected) layer of 100 units and relu

activation.A dense layer of n units(where n is the number of classes) and softmax activation that outputs the final probabilities of belonging to each of the n classes. Softmax activation is used here since it goes best with categorical cross-entropy loss, which is the loss we are going to be using to train the model.

As we can see in the results that for SNIPS dataset, it has produced very good accuracy, precision, recall and F1-score. But coming towards ATIS dataset which performs exceptionally for this model, the reason behind it is that the ATIS dataset is highly imbalanced that is why all the texts are being classified into few of the intents that is why it is performing pretty well. For the Banking dataset, it produced good results as because it is balanced but it has highest number of intents for that reason its accuracy is less.

	Accuracy	Precision	Recall	F1-Score
SNIPS	97.53	97.57	97.57	97.57
ATIS	98.98	99.00	99.00	99.00
BANKING	77.70	77.08	77.60	77.84

Table 4.2.2.1 The table depicts the result obtained for all three datasets using CNN

4.2.3 Classification models

To train supervised classifiers, we first transformed the data into a vector of numbers using TFIDF, and evaluate how important a particular word is in the collection of words. For this we need to remove punctuations and do lower casing, then the word importance is determined in terms of frequency. After this perform linear SVC,Random Forest, Multinomial Naïve Bayes and Logistic regression. We have seen that Linear Support Vector Machine outperforms all the other classification algorithms. Sothen we train the model on Linear SVC and saw how it works on unseen data and we have applied all three datasets on this model. As we can see that SNIPS dataset work best as because it has less number of classes and for ATIS dataset we can see in Fig 4.2.3.2 that for ‘atis_flight’,’atis_airfare’,’atis_flight_time’,’atis_aircraft’,’atis_airline’ and ‘atis_abbreviation’ has all the values and rest of them does not have any values and that is why rest of them are left 0 and at last Banking dataset which has thehighest number of classes present in it and it produced the results which are good.

	Precision	Recall	F1-Score
SNIPS	97.00	97.00	97.00
ATIS	94.00	91.00	91.00
BANKING	82.00	81.00	81.00

Table 4.2.3.1 The table depicts the result obtained for all three datasets using Classical Classification models

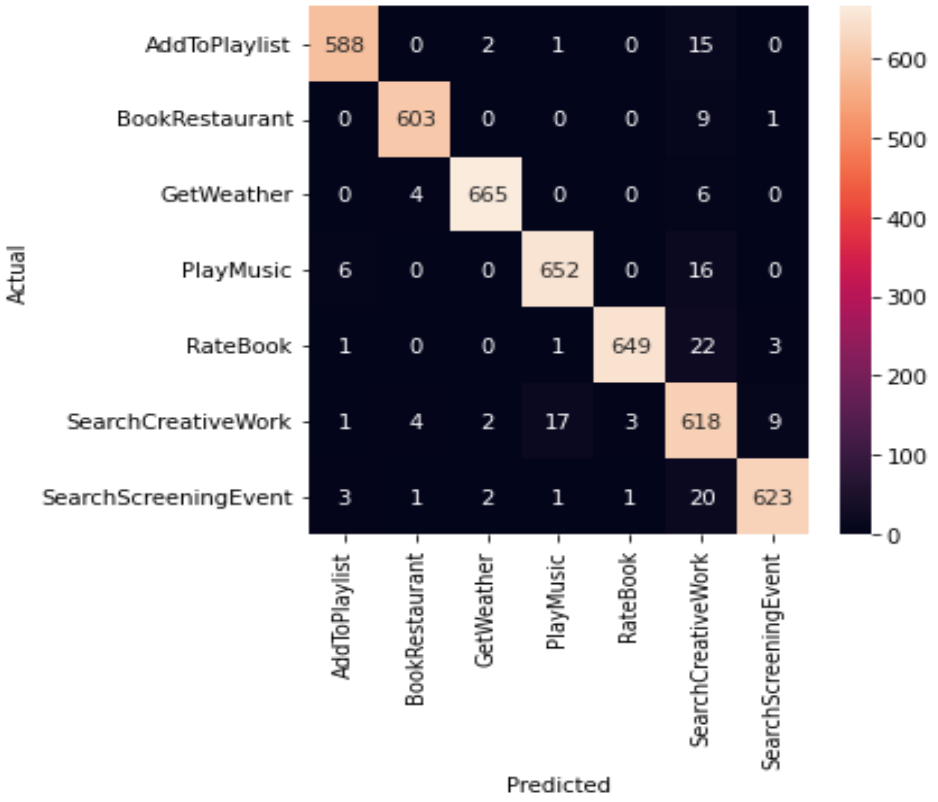

Fig 4.2.3.1: Confusion Matrix for SNIPS dataset for Classification Model

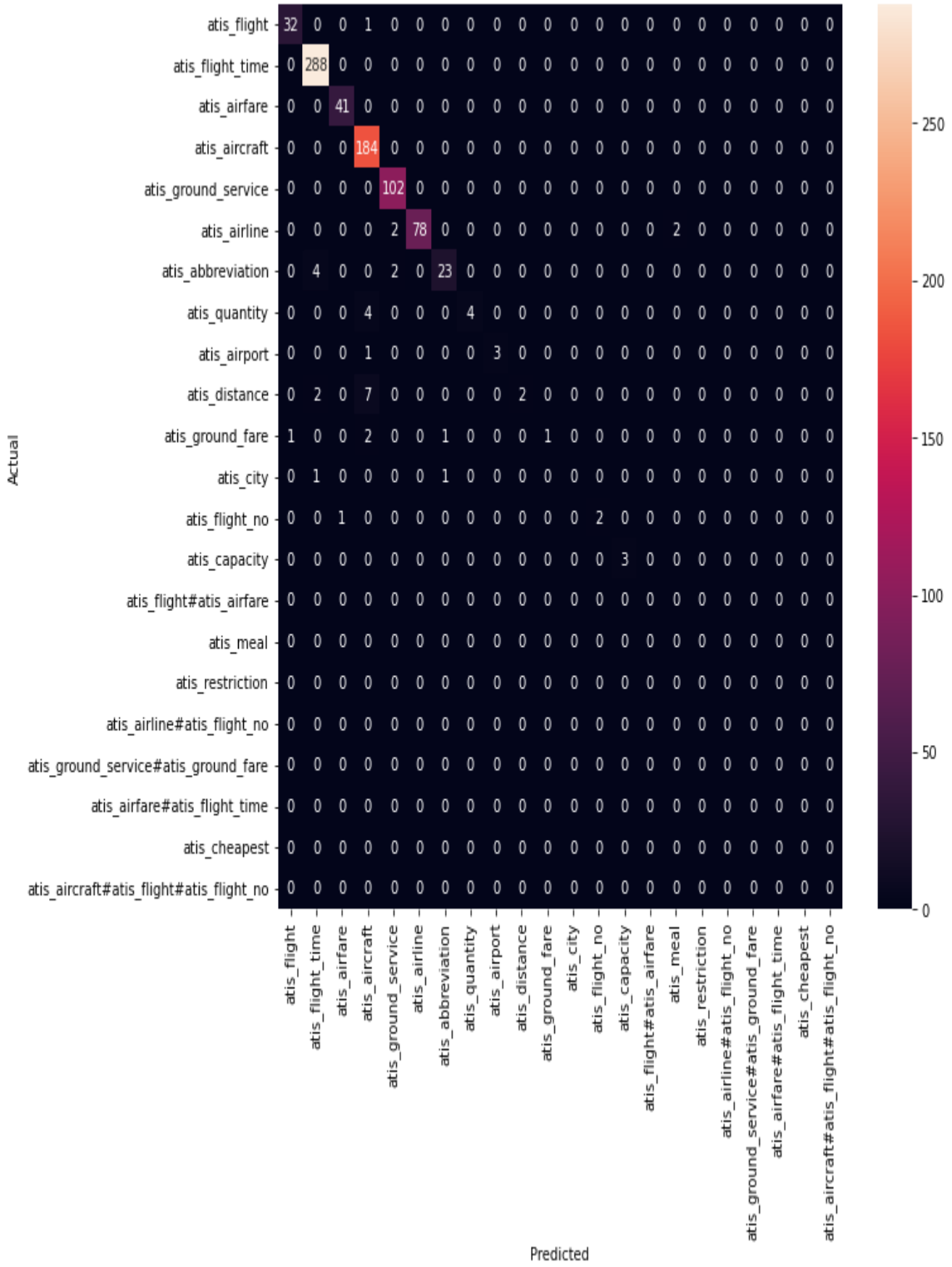

Fig 4.2.3.2: Confusion Matrix for ATIS dataset for Classification Model

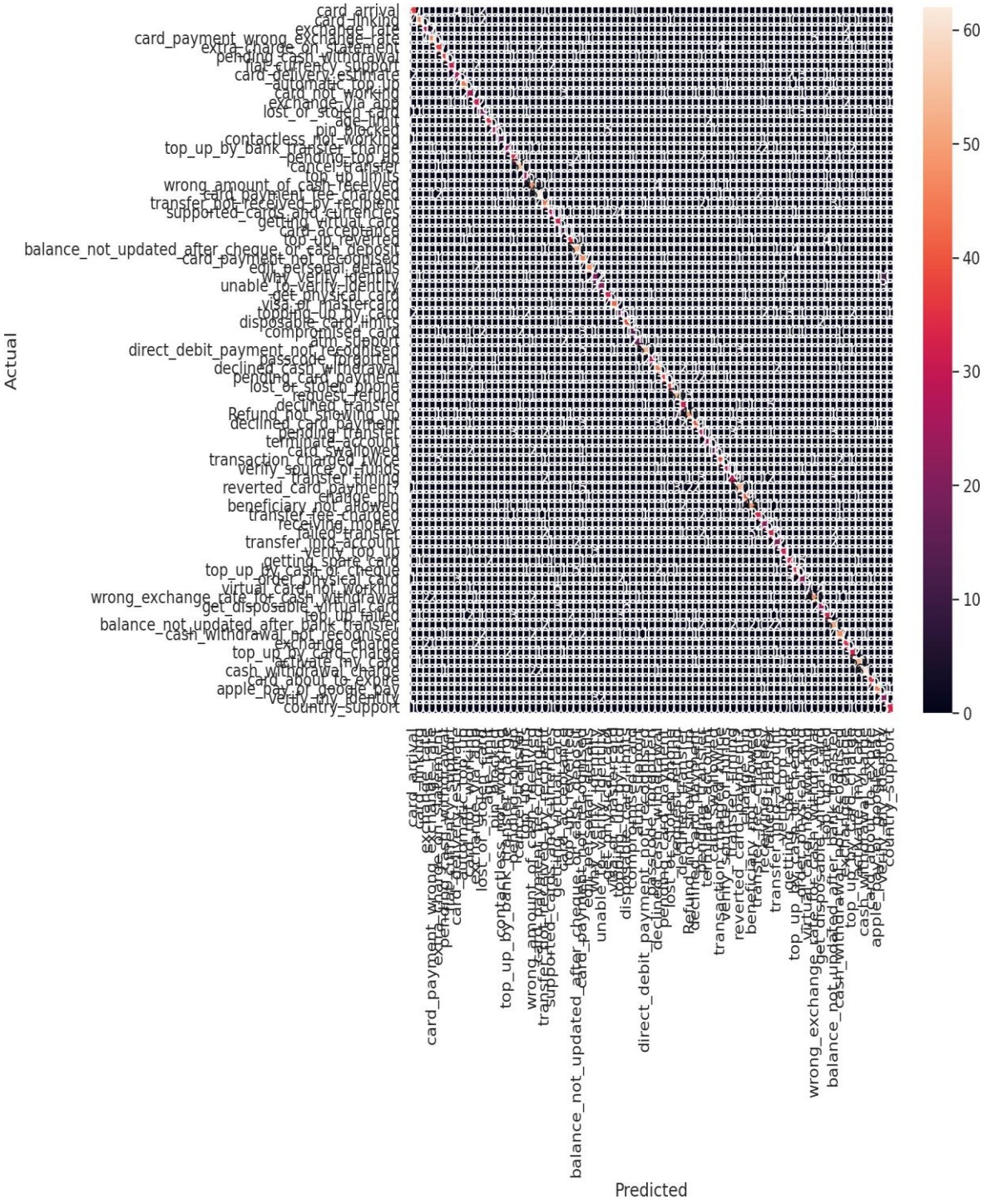

Fig 4.2.3.3: Confusion Matrix for Banking Dataset for Classification Model

4.2.4 BERT

We have used the pretrained BERT model ‘uncased_L-12_H-768_A-12’ to train.We first pad the text for getting the padding length by taking the minimum between the longest text and the max sequence length parameter. We also surround the tokens for each text with two special tokens: start with [CLS] and end with [SEP].We’re fine-tuning the pre-trained BERT model using our inputs (text and intent). We also flatten the output and add Dropout with two Fully-Connected layers. The last layer has a softmax activation function. The number of outputs is equal to the number of intents. We have used Adam with sparse categorical crossentropy.

As can be seen in the Table 4.2.4.1, it provided very good accuracy, precision, recall, and F1-score for the SNIPS dataset. However, when it comes to the ATIS dataset, which performs incredibly well for this model, the reason for this is that the ATIS dataset is highly imbalanced, which is why all of the texts are classified into a small number of intents, which is why it works so well.

	Accuracy	Precision	Recall	F1-Score
SNIPS	99.56	98.00	97.00	97.00
ATIS	81.16	62.00	79.00	70.00

Table 4.2.4.1 The table depicts the result obtained for two datasets (SNIPS and ATIS) datasets using BERT

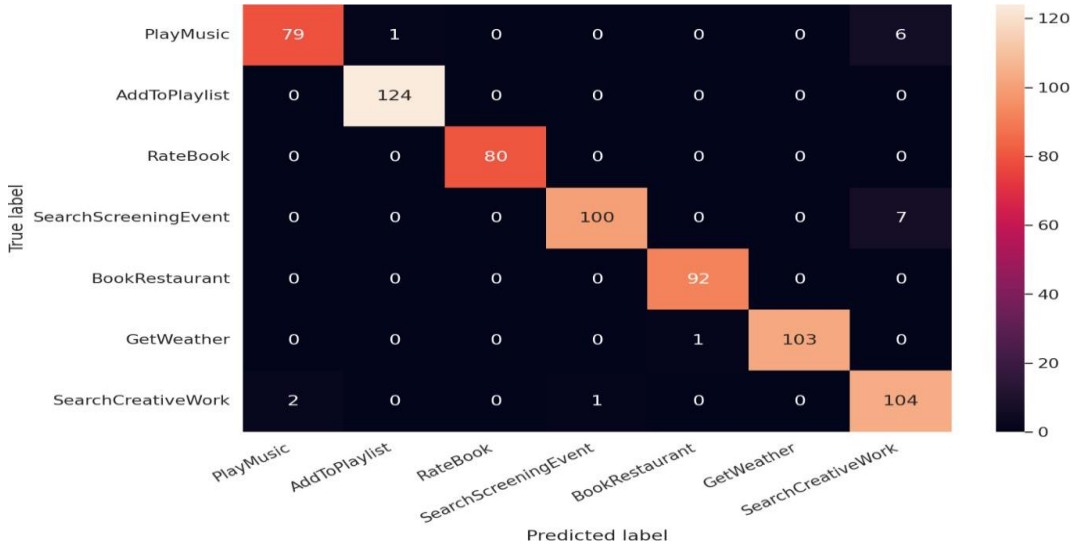

Fig4.2.4.1: Confusion Matrix for SNIPS dataset

Conclusion

In this thesis, we have tried to compare three datasets on different models through which we have learnt to see how intent classification can be done. The SNIPS dataset is the smallest dataset used in this thesis which had only seven classes. We firstly implemented this dataset to the text classification models in which we compared basically four algorithms which are linear support vector machine, logistic regression, random forest and multinomial naïve bayes. We saw that linear SVC is performing well on this model, so we train it and then we get very good accuracy. Then we implemented this dataset to deep learning models like LSTM (Long Short Term Memory), CNN(Convolutional Neural Network) and BERT. We saw that for this dataset, LSTM performed very well with highest accuracy among all the models. For the ATIS dataset, the total number of classes present is 22, but the main problem with this dataset was that it was highly imbalanced, that is, maximum texts that is presentin this dataset are mapped to a few number of classes. As for the performance of ATISdataset on all the models, it performs the best in CNN model. For the BANKING77 dataset, this dataset has the highest number of classes which is 77, we see that the precision we get from all the models is the highest and as for the accuracy all the models produce the optimal results given the number of classes present in the dataset.

At last we can say after implementing all the datasets on various models, for the text classification models the Support vector machine did best for all the datasets and as for the deep leaning models, BERT produced the best results.

References

[1]. Joo-Kyung Kim et al. “Intent detection using semantically enriched word embeddings”. In: 2016 IEEE Spoken Language Technology Workshop (SLT). IEEE. 2016, pp. 414–419

[2]. J. Pennington, R. Socher, and C. D. Manning, “Glove: Global Vectors for Word Representation,” in Proc. EMNLP, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[3]. K. Shridhar, A. Dash, A. Sahu, G. G. Pihlgren, P. Alonso, V. Pondenkandath, G. Kov´acs, F. Simistira, and M. Liwicki, “Subword Semantic Hashing for Intent Classification on Small Datasets,” in International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–6.

[4]. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[5]. Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. Understanding user’s query intent with wikipedia. In Proceedings of the 18th international conference on World wide web, pages 471–480. ACM, 2009.

[6]. Liliana Calder´on-Benavides. Unsupervised Identification of the User’s Query Intent in Web Search. Universitat Pompeu Fabra, 2011.

[7]. Qian Chen, Zhu Zhuo, Wen Wang . BERT for Joint Intent Classification and Slot Filling, Speech Lab, DAMO Academy, Alibaba Group,2019

[8]. Patrick Pantel, Thomas Lin, and Michael Gamon. Mining entity types from query logs via user intent modeling. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, pages 563–571. Association for Computational Linguistics, 2012.

[9]. Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, Jason Mars ·Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling,2016

[10]. Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, Jian Sun ·Induction Networks for Few-Shot Text Classification, arXiv:1902.10482v2, 2019

[11]. Sepp Hochreiter , Jürgen Schmidhuber, Long Short-Term Memory,1997, pp 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735

[12]. Andrei Broder. A taxonomy of web search. In ACM Sigir forum, volume 36, pages 3–10. ACM, 2002.

[13]. Xiaodong Zhang and Houfeng Wang. “A joint model of intent determination and slot filling for spoken language understanding.” In: IJCAI. Vol. 16. 2016, pp. 2993–2999.

[14]. Liliana Calder´on-Benavides. Unsupervised Identification of the User’s Query Intent in Web Search. Universitat Pompeu Fabra, 2011.

[15]. Giovanni Di Gennaro, Amedeo Buonanno, Antonio Di Girolamo, Armando Ospedale, Francesco A.N. Palmieri. Intent Classification in Question-Answering Using LSTM Architectures, arXiv:2001.09330, 2020

[16]. Alex Sherstinsky, Fundamentals of Recurrent Neural Network (RNN) andLong Short-Term Memory (LSTM) Network, arXiv:1808.03314, 2018
[17]. Yuandong Luan; Shaofu Lin. Research on Text Classification Based on CNN and LSTM pages 1-40, ACM, 2022
[18]. Mengyuan Gao, Tinghui Li.Text Classification Research Based on Improved Word2vec and CNN, Service-Oriented Computing – ICSOC 2018 Workshops
[19]. Liliana Calder´on-Benavides, Cristina Gonz´alez-Caro, and Ricardo Baeza-Yates. Towards a deeper understanding of the users query intent. In SIGIR 2010 Workshop on Query Representation and Understanding, pages 21–24, 2010.
[20]. Gokhan Tur and Renato De Mori. Spoken language understanding: Systems for extracting semantic information from speech. John Wiley & Sons, 2011.
[21]. Nicholas J Belkin et al. Interaction with texts: Information retrieval as information seeking behavior. Information retrieval, 93:55–66, 1993.
[22]. Bernd Hollerit, Mark Kr¨oll, and Markus Strohmaier. Towards linking buyers and sellers: detecting commercial intent on twitter. In Proceedings of the 22nd International Conference on World Wide Web, pages 629–632. ACM, 2013
[23]. Jinpeng Wang, Gao Cong, Wayne Xin Zhao, and Xiaoming Li. Mining user intents in twitter: A semi-supervised approach to inferring intent categories for tweets. In AAAI, pages 318–324, 2015.
[24]. Daniel E Rose and Danny Levinson. Understanding user goals in web search. In Proceedings of the 13th international conference on World Wide Web, pages 13–19. ACM, 2004.
[25]. Olivier Chapelle, Shihao Ji, Ciya Liao, Emre Velipasaoglu, Larry Lai, and Su-Lin Wu. Intentbased diversification of web search results: metrics and algorithms. Information Retrieval, 14(6):572–592, 2011.
[26]. Tom Young et al. “Recent trends in deep learning based natural language processing”. In: ieee Computational intelligenCe magazine 13.3 (2018), pp. 55–75.
[27]. Ryen W White, Paul N Bennett, and Susan T Dumais. Predicting short-term interests using activity-based search context. In Proceedings of the 19th ACM international conference on Information and knowledge management, pages 1009–1018. ACM, 2010.
[28]. Justin Cheng, Caroline Lo, and Jure Leskovec. Predicting intent using activity logs: How goal specificity and temporal range affect user behavior. In Proceedings of the 26th International Conference on World Wide Web Companion, pages 593–601. International World Wide Web Conferences Steering Committee, 2017.
[29]. Long Chen, Dell Zhang, and Levene Mark. Understanding user intent in community question answering. In Proceedings of the 21st International Conference on World Wide Web, pages 823–828. ACM, 2012.
[30]. Suman Ravuri and Andreas Stoicke. “A comparative study of neural network models for lexical intent classification”. In: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). IEEE. 2015, pp. 368–374.

[31]. Dilek Hakkani-Tür et al. “Multi-domain joint semantic frame parsing using bi-directional rnn-lstm.” In: Interspeech. 2016, pp. 715–719.
[32]. Daniel Guo et al. “Joint semantic utterance classification and slot filling with recursive neural networks”. In: 2014 IEEE Spoken Language Technology Workshop (SLT). IEEE. 2014, pp. 554–559.
[33]. Bing Liu and Ian Lane. “Attention-based recurrent neural network models for joint intent detection and slot filling”. In: arXiv preprint arXiv:1609. 01454 (2016)
[34]. Dilek Hakkani-Tür et al. “Multi-domain joint semantic frame parsing using bi-directional rnn-lstm.” In: Interspeech. 2016, pp. 715–719.
[35]. Chih-Wen Goo et al. “Slot-gated modeling for joint slot filling and intent prediction”. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). 2018, pp. 753– 757.
[36]. Xin Wayne Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu, and Xiaoming Li. We know what you want to buy: a demographic-based system for product recommendation on microblogs. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1935–1944. ACM, 2014.
[37]. Stuart J Russell and Peter Norvig. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, 2016, pp. 744–748.
[38]. Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),pp. 1724–1734