Fine-Tuning Bert Model for Intent Classification task using GAN

MSc Research Project
Masters in Data Analytics

Sayali Kule
Student ID: 21101205

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sayali Kule
Student ID:	21101205
Programme:	Masters in Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Abubakr Siddig
Submission Due Date:	15/08/2022
Project Title:	Fine-Tuning Bert Model for Intent Classification task using GAN
Word Count:	9087
Page Count:	30

Signature:	Sayali Kule
Date:	18th September 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Fine-Tuning Bert Model for Intent Classification task using GAN

Sayali Kule
21101205

Abstract

The intelligent chatbot systems are the result of natural language processing and neural network algorithm. A chatbot is nothing but a virtual human-like assistant, that helps in completing a task or provides required service in no time through a conversation. The task-oriented dialog systems need to classify the query intent correctly to avoid incorrect responses. Even though the issue of a low data regime is common, this study proposes the use of the pre-trained transformer model BERT which is fine-tuned under adversarial settings for the intent classification task. This unique method makes use of a few shots of supervised data and the GAN model allows the adaptation of unsupervised data which is available easily. With the experimental comparison and validation, the performance of the model was found to improve when fewer examples were trained for longer epochs. The size of annotated examples can be reduced up to 60, still obtaining an accuracy of 94.8% and an F1 score of 0.95. This approach successfully allowed us to achieve high performance even with low resources.

1 Introduction

1.1 Background

The world is blessed by the existence of deep learning and artificial intelligence. A widely used example of this is a chatbot. A chatbot is nothing but a virtual human-like assistant, that helps in completing a task or provides required service in no time through a conversation. The commutation happens through an interface or voice commands. It had made a lot of day-to-day tasks easier. Natural language understanding has helped chatbots understand human queries and automate the responses. It is a low-cost engagement and it helps in resource optimization.

A chatbot is taking a language and then traversing it through a decision tree in order to take an action. The very first chatbot was developed in 1966 and named ‘Eliza’. Eliza was a very simple example, think of it as a bunch of ‘If-then’ Statements. That is what was called a ‘Rule-Based’ Chatbot. In 2000, and the name of the chatbot ‘ALICE’, which stands for Artificial Language Internet Computer Entity, Alice’s sole purpose was to use pattern Recognition, she was just designed to communicate with humans, She was the foundation for most of modern-day Chatbots. ALICE however, didn’t pass the Turing Test. In 2010, Apple’s Siri, was kind of the first entry to using a chatbot as a personal assistant. That also led to the development of Home-assistants, in 2010, Think about a smart speaker(Current day Alexa or google assistant). This led to the introduction of chatbots in 2016.

Every domain significantly makes use of a conversational system irrespective of the service it provides. A few highly popular examples are ‘Siri’, ‘Alexa’, an educational chatbot, and a virtual assistant for vaccination appointment booking. A lot of service-based websites makes use of chatbot to exploit their capabilities and provide continuous service. The conversational system is available all day long without any waiting time. It reduces manual intervention. It allows parallelism to provide responses to all the users.

Traditional chatbots were question-answer or rule-based. The chatbots were designed to answer a limited number of frequently asked questions in no due time. This used the pre-defined set of question-answer. It did not help much because the corpora of the question increased over due course of time. Updating the database every time the new query comes in wasn’t feasible. This led to upgrading the chatbot, which allowed user’s to feed in the queries. This led to a new era of the digital assistance system.

1.2 Motivation

Intent classification plays a vital role in the architecture of digital assistance. It is important to know the intent behind the user’s request. If not classified correctly, a chatbot may fail to serve the purpose. A known challenge is, that human language is difficult to understand. The meaning of a sentence is flipped when a word’s position changes. The text-based queries may have jargon, incorrectly spelled words, and abbreviations. To achieve better performance, high-quality large labeled datasets are required which helps to comprehend the semantic similarity of queries. The mentioned limitations might cause difficulty in understanding the intent and responding back incorrectly. The study focuses on making the dialogue system more comprehensive.

Deep learning techniques have given high performance in all Natural Language Processing tasks. Devlin et al. (2019) exposed the capabilities of the transformer-based BERT model and proved that it has achieved state-of-the-art performance. The pre-trained models are easy to implement and can be fine-tuned for the targeted task. The experiments were performed on a large amount of supervised data to get fine results. When Croce et al. (2020) tried applying the same model to the dataset with less than 200 records, it did not yield significant results.

So far the researchers have taken care of supervised data in the majority of cases. The research proved that the results were better only when the model was trained on large data. The lack of labeled data might under-fit the model and creating the labeled data involves a lot of manual effort. It is costly and time-consuming. This paper focused on using the easily available unsupervised data. This has motivated to explore the deep learning techniques that could make use of unlabeled data and still understand the intents and classify it correctly.

This research study proposed a pre-trained BERT-based model which uses the discriminator from the GAN(Generative adversarial network) to classify the intents. This study makes use of a combination of labeled and unlabelled data. Croce et al. (2020) proposed this GAN-BERT model for text classification to conclude that BERT can be fine-tuned over fewer examples and still achieve high performance. The generic idea is to explore the capabilities of BERT, it converts the examples into a high-quality representation and also adapts the unlabelled data. The generator keeps creating fake data similar to real examples and it trains the discriminator to classify the intents. The argument or hypothesis here is the BERT model can be fine-tuned using the GAN settings.

1.3 Research Question

This paper studies the extent to which a pre-trained BERT model under a semi-supervised generative adversarial setting be fine-tuned for classifying intent and understanding the optimal amount of data required to achieve high performance.

1.4 Objective

1. To build a model using pre-trained BERT and GAN
2. Perform different experiments on the chosen dataset
3. Evaluate the model performance based on new data

1.5 Plan of the paper

This research paper is further sectioned as follows. Section 2 describes the previous work in the same field. Section 3 describes the research methodology used in this research work. Section 4 describes the proposed approach in detail. Section 5 defines the steps involved in the implementation stage. Section 6 discusses the experiments performed and results obtained that support the research question. Section 7 provides the summary about the study conducted and makes some suggestion about future work that may be useful.

2 Related Work

With emerging advancements seen in machine learning, a lot of learning has been implemented for conversational systems or digital assistance. It has played a vital role in discarding the manual intervention needed for answering human queries. The legacy method used for intent classification which is a part of a dialogue management system were SVM, logistic regression, or Naïve Bayes. So far following deep learning techniques have been applied for intent classification task: CNN, RNN, BLSTM, and the self-attention model. The self-attention-based transformer model has given state-of-the-art results in many NLP-related tasks. Schmidt and Wiegand (2017)(Kresnakova et al.; 2020) . This section discusses and compares the previous research explored for the intent classification task. The research papers are grouped together in different subsections.

The digital assistance systems are continuously being improved to have a human-level understanding of the queries presented and the intelligence. The study compared by Abedin et al. (2021) chose to apply Bi-LSTM and Bi-GRU models on online banking query-based dataset. The bi-directional approach helped to understand the context of the words in a sentence. Bi-GRU outperformed the Bi-LSTM because of its easy modification, faster to-train feature and it works without the memory units.

2.1 Universal Intent Recognition System

As digital assistance is being widely accepted in all domains for different purposes. An example of this could be booking a flight, booking an appointment, information retrieval system. In any dialogue management system, a few sentences like salutation/greeting, farewell messages, and transfer to the agent are common. A study proposed by Vasquez Correa et al. (2021) based on a word-embedding and deep learning classification model was used to discriminate the universal intents. These experiments were performed on the Spanish and English language datasets. The study compared the performance achieved by applying word2vec and BERT word-embedding. The classifier used was CNN and RNN. The evaluation matrix described, for the Spanish language word2vec embedding ranked better accuracy and similar accuracy was observed for the English language with a small set of words. The context-independent word2vec embedding outperformed this experiment because the common intents have a small set vocabulary and understanding of the context was not significant. The accuracy achieved was 80.4%. The same developed model was able to detect the intents for short and long text. The Convolutional neural network yielded the highest accuracy. This study addressed an important error, misclassification error for intents having similar intents.

The above-mentioned experiment dealt with a small set of data to identify the intents where the method applied was context-independent. In contrast to this, a study based on the Arabic language was proposed by Mezzi et al. (2022) to check the mental health of the patients from the Military Hospital of Tunis, Tunisia. The answers received from the patients in an interview were given as input to the transformer-based BERT model. This model diagnosed the problem and classified it as depression, social phobia, adjustment disorder, panic disorder, and suicidality. The evaluation criteria used were accuracy, recall value, precision value, false positive rate(FPR), F1 score, false-negative rate (FNR), and specificity as well as a subjective test where a psychiatrist evaluated the diagnosis. The accuracy achieved was above 92% and the F1 score was above 94%.

Though the pre-trained model has proven great performance in NLP, they were checked only against the English language. Researchers Martin (2020) explored the capacity of the pre-trained model in the French language using a version of BERT, CamemBERT. This has shown effective findings in text classification, Named entity Recognition and Part of speech tag. Though the size of the model is large and requires high computational time and a large amount of data, the performance achieved by the pre-trained word vectors on monolingual approaches was significantly high.

2.2 Bilingual Intent recognition

So far the review described approaches chosen for monolingual language, but a survey proved there are more multilingual speakers. In daily life, nowadays every person happens to speak a mixture of languages. This led to exploring the research that happened for intent classification tasks when the input text contains a mix of languages. A model was proposed by Hu et al. (2021) to handle the Chinese and English languages. The information retrieval from a mixed sentence is difficult hence the proposed method used wordent and word2vec where the difference between the languages can be identified using synset id. Synsets are cognitive synonyms of a word in a lexical database. The challenge of text information was solved by introducing word intention features, transformer-based context-dependent features, and word frequency features. This model was applied to online logistic data, results explained improved accuracy because of semantic features added by the transformer model.

2.3 Self-Attention mechanism

The legacy technique to process natural language was the Seq2seq encoder-decoder which was later evolved with bidirectional models. The Seq2seq processes all the words one by one from left to right and passed them to the recurrent neural network-based encoder. The encoder outputs a semantic representation of the sequence provided which was later classified by the decoder. The decoder decodes this representation to create an output sequence. This model is easy to implement and works in a low-resource environment. The disadvantage of this model is it failed to understand the context of the words. To understand the natural language when presented in a long text, the order is important to comprehend the information, to achieve this memory-based bi-directional RNN, Gated recurrent unit or long short-term memory models are preferred as it allows to store the memory of each processed element before accessing next element.

A lot has been changed during the pandemic time including the education system. The importance of virtual teaching assistants has gained a lot of popularity because of their ease of convenience. A Xatkit framework-based Edu-chatbot was developed by BAHA et al. (2022) in the French language. It used the pre-trained model trained on a French dataset named Camem-BERT. The architecture uses a multi-headed self-attention mechanism. The bidirectional transformer-based encoder helps understand the context relations. The decoder has the same architecture where each head aims at different projections and creates a query, key, and value vector to classify the intent. In addition to this study, a 3D avatar was produced to give out the generated responses.

For the Vietnamese University, Le-Tien et al. (2022) created a chatbot to ease the admission process, course selection etc. The intents were classified as business and random to reduce the complexity and increase the performance of the targeted task. The Levenshtein algorithm ensures all the spelling mistakes were ignored and it reduced the noise related to the keyword which has different intentions. Haldar et al. (2011) This algorithm calculates the distance from each sample to the word. The classification problem was addressed by adding a Bi-directional LSTM layer on the embedding layer with an attention mechanism. 0.89 of F1 score was achieved by this model.

2.4 Transfer Learning Models

Knowledge distillation also known as transfer learning is highly popular in deep learning. The models are trained on large corpora. Keeping aside the popularity, they are difficult to support because of high QPS computational performance. The concept of transfer learning on NLP was explained by Wang et al. (2020). A teacher-student distillation model was introduced to improve the performance of a small-scale model. The concept of this model was, that the teacher model has strong learning ability and will be rigorously trained on a complex network to learn the features. This knowledge will be later transferred to a student model with weak learning skills. The learnings captured by the teacher model helped the small-scale model to perform better.

FastText , a text classifier similar to continuous bag of words has proven to have faster training and evaluation capacity. Joulin et al. (2016). The FastText has the capacity to train ¿ 1 billion words in a few minutes using a standard CPU. It can also discriminate half a million sentences in lass a minute into 312 different classes. Though it has great learning capacity, when tested for text classification, it did not yield significant results. In comparison to traditional models, a complex ERNIE ("Enhanced Representation through knowledge Integration) model has outperformed other models for the intent classification tasks. The drawback of ERNIE was its complex structure and high computational cost. This ERNIE model was used as pre-trained teacher model by Guo et al. (2022) on massive online unlabelled data to predict the intent and the FastText model was treated as a student model to raise the performance and applied to the e-commerce datatset. The uniqueness of this study was the use of unlabelled data as labeling is expensive. The result comparison shows the accuracy of FastText was improved by 0.94% to original model.

2.5 Unlabelled Data

The earlier mentioned research paper dealt with enough amount of data to achieve that high accuracy. But it was difficult to find the correct data to classify the intent for each domain. This raised awareness about working with unlabelled data which is available generally. The pre-trained models like ULMFiT, GPT3, BERT, and ELMO were developed to handle unsupervised data. This section talks about how the unlabelled data has been handled so far by different models and their results. The previous section described ERNIE by Guo et al. (2022) which was applied to unlabelled data for the Chinese language for an e-commerce dataset.

Recently to adapt unsupervised domain learning Lew et al. (2021) proposed a Conversation clustering Adaptation model. The architecture of this was a combination of different methods, a clustering approach, and an encoder domain adaptation applied to cluster classification, fine-tuning the model over labeled data. This experimented against the Polish banking dataset and open web app data. The assumption considered was if the two queries have the same answers, then it has the same intent. The implementation used the BERT model to train and get the domain knowledge. The embedding of vector representation was clustered using the k- means algorithm and ranked. To which cluster the query belongs to was predicted by the model. The unlabelled data was used to complement the labeled data to fine-tune the model for intent detection. The accuracy achieved by this was 96.6% on web app and 92.5% for banking data.

The intents defined for an assembly process are different at each stage, this leads to a complex classification process. To resolve this the same question was answered differently at each stage though the intent was the same. The experiment carried out with the CNN-LSTM model used in combination with the text and step feature failed to identify the intents. The reason for failure was the model created a noise factor when the intents were stage-independent. Chiu et al. (2021) provided a resolution by proposing a multi-task learning model. The main task of this was intent classification and the secondary task was step-independent dialogue act classification. The result comparison shows that the multi-task model outperformed the single-task model.

2.6 Minimal Data Modelling

2.6.1 Few shot Learning

As seen in previous sections, gathering the required data was challenging and expensive as well. To resolve this, Wang et al. (2020) introduced a few shot learning models which bridge the gap between artificial intelligence and human learning. The study described the model architecture of few-shot learning in NLP. It was based on distant supervision, knowledge distillation, and meta-learning. A study focused on the following objective: How much minimum training data was required for intent classification? How does the number of samples affect the performance? was it compatible with the models which were easy to implement and customize as per need? To evaluate the performance, the model was trained on a few training samples. Fine-tuning was applied by checking the dependency on syntactic and semantic features. When dealing with a low amount of data, the memory-based models were preferred as the network learns from the events. To support this Luo et al. (2018) explained a way to deal with minimal training data combined with regular expression. It used a bi-directional LSTM model to improve the performance. The only drawback of this proposal was, that an expert was needed to write a regular expression.

The main idea to train the model with minimum data from a typical training sample was similar kinds of data can be generated using models like GAN. It reduces the model building cost as well. Huggins et al. (2021) took the challenge to train the model with less data for intent recognition. It compared BERT and Bi-LSTM models. The results proved the entire dataset achieved 98% of accuracy whereas only 25 training samples per intent scored 94% of accuracy. The common observation was, that the performance increased significantly after a small increase in training examples. The quality of the data equally impacts the training. This study successfully proved a few shot learning can achieve better performance.

2.6.2 Zero shot Learning

The previous section proved that a few shot learning models can identify the intents correctly. This has motivated to check the behavior and performance of zero-shot learning. The models were trained on a dataset with no label utterances. Xia et al. (2018) proposed a correlation-based model known as INTENTCAPSNET- ZSL. It benefitted from the advantages of the capsule model of text modeling. This model can derive intent representation for new intents. Semantic features were learned via the self-attention method. Those were aggregated to identify the intent at the utterance level. This model made use of the knowledge transfer technique, it used learning from existing intents to classify the new intents when there's no utterance. The comparison proved model performed better when compared with the BOW classifier using TF-IDF and other models.

2.6.3 Half shot Learning

The new intents keep coming in over the period of time. These new intents were handled in a model, by following the method. The system was supposed to collect the new intents into training data. It is said that natural language understanding should not be dependent on learning from pre-trained models and sample efficiency. This removes the need of building a new dataset of sample sentences. A hot-shot learning NLU framework AidME (Adaptive Intent detection in Multi-Domain Environments) was given by Lair et al. (2020). The training was performed on gathered sentences and adapted new intents. It had a capacity to learn new patterns after the interaction. The proposed framework was not domain-specific, it can be reused to develop digital assistance without hampering performance. The results presented by this study were interesting, 90% time AidMe successfully identified intents even with 10 samples collected. The comparison shows, that it outperformed one-shot learning by applying zero-shot learning on 43% of the data.

2.7 Domain Specific Model

The domain knowledge was adapted by training word2vec on the input text and the output vectors were aligned with wordpiece, achieved through a pre-trained language model applied to the general domain. The traditional fine-tuning methods have failed for domain-related vocabulary. Poor results were obtained when the pre-trained model was extended to fine-tune. The reason for failure could be the input query might have words from the original and new domains. This motivated to create a method that could handle a mix of the vocabulary that will allow a PTM to adapt new domain. A study conducted on Covid19 domain by **?** resulted that the performance obtained by BERT and BETO variant was significant when compared to traditional recurrent networks. The term frequency decided which words should be added to the vocabulary. The drawback of this method was the exact and optimal number of new tokens to be added was unknown. Another finding from the study was text normalization techniques resulted in better performance over data embedding techniques.

For a domain-centric digital assistant, it is equally important to identify the intents belonging to the domain and within the scope, which can be answered. Classifying the out-of-scope intents would help the system to avoid giving out wrong responses. Larson et al. (2019) dealt with the classification of in-scope and out-of-scope intents by applying different deep learning models. The result proved that BERT has outperformed all the other techniques. The same performance was achieved with the imbalanced dataset.

2.8 Literature Gap

The literature review has described the different work done so far in overcoming the challenges of the intent classification task. The challenges addressed were low data, domain knowledge, multilingual language, out of scope intents, out of vocab words. In all the cases the self-attention-based transformer and pre-trained model have given significant results. It is observed that Generative Adversarial Network are not much explored in the field of NLP for intent recognition task specifically. It would be interesting to explore the ability of high performance models like GAN.

3 Methodology

This section describes the data selection, and collection through the methodology. This study followed the KDD approach carried out in an iterative manner to enhance evaluation measures.

3.1 Data selection and Data Understanding

This research study deals with commands or queries given to any conversational system. There are very few datasets available for the study. This dataset was chosen from previous work. This dataset was recently made available publicly via Kaggle, it is also available at [1] This project runs a few experiments on the mentioned dataset and draws interesting inferences.

1. CLINC150 Dataset

[1]Dataset URL: `https://aclanthology.org/D19-1131/`

This dataset contains queries or commands used by users in the English language. This dataset is in JSON format and contains labeled and unlabelled data. The labeled data has a total of 15000 records describing 150 intents. Each intent is described in 100 different ways for the training dataset which was crowdsourced. The test dataset contains 4500 records and another 15000 unlabelled records are available. This dataset covers queries from 10 different domains namely banking, home, travel, kitchen, work etc. The dataset is balanced as the distribution of queries is equal.

3.2 Data Pre-processing

The data pre-processing step is followed by the data analysis steps. The data analysis steps included understanding the dataset structure, analyzing using plotting graphs, and understanding the available intents. The unique labels were derived from the dataset and used later for classification purposes.

The dataset is text-based, the pre-processing steps involved applying the text processing steps to covert the queries into lower representation. The text was processed by cleaning the data, removing the punctuations, lowering the text, and removing the stop words. Since this study used a pre-trained transformer model, the queries were tokenized using the corresponding tokenizer. The data was prepared to be used as an input for the modeling technique. This study explored how the semi-supervised can be implemented and how the model behaves differently when the record sizes are adjusted. The dataset is already divided into train and test, making it easier to have balanced data.

3.3 Model Building

The general idea of semi-supervised intent classification is given as: The input data was a collection of labeled and unlabelled queries. This data was converted into vector representation using a pre-trained BERT model. A generator and a discriminator were defined and initialized. The generator created fake examples similar to real data. The tensors of real data and fake data were provided to the discriminator for classification. In the proposed semi-supervised model, the labeled data was used to train the discriminator while the unlabelled data and fake data improved its inner representation. The discriminator was used to classify the intents. This baseline approach was implemented using BERT and GAN, addressing the issue of scarcity of annotated data and giving higher accuracy in detecting the intent.

3.4 Evaluation

This study deals with the classification problem. To understand the model behavior, the accuracy and the loss were noted at each step. Since the model involved GAN, the generator's loss and the discriminator's loss was also observed. The evaluation of the proposed model was given by a confusion matrix. Since there are 150 different intents, instead of storing the confusion matrix, each step calculated the total number of correctly and incorrectly identified intents. The results obtained from different experiments will be discussed in the section 6

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

Where:

TP – True positive, model correctly predicted the positive class
TN – True negative, model correctly predicted the negative class
FP – False positive, model incorrectly predicted the positive class
FN- False Negative, model incorrectly predicted the negative class

4 Design Specification

The proposed methodology experiments with adding extra layers of GAN to the existing BERT model to fine-tune it for the intent classification task. The two additional layers of the architecture are intent identification layers and to adapt to semi-supervised learning add GAN layers.

The below figure Figure 1 illustrates the semi-supervised GAN model for the intent classification system. The architecture of the model involves a transformer model and a generative adversarial model. There are three main components in this architecture. The vector representation of training data is given as an input to the BERT model. Bert provides a representation of the given input as a result of the pre-training stage. The Multi-Layer Perceptron(MLP) Generator module of the GAN model creates the vector representation of fake examples using a 100-dimensional noise vector drawn from Gaussian Distribution. The other MLP, the discriminator receives input from both which includes real and fake data. If there were k intents to be recognized, the last layer of the discriminator is a softmax-activated layer, and this results in k+1 vector of logits.

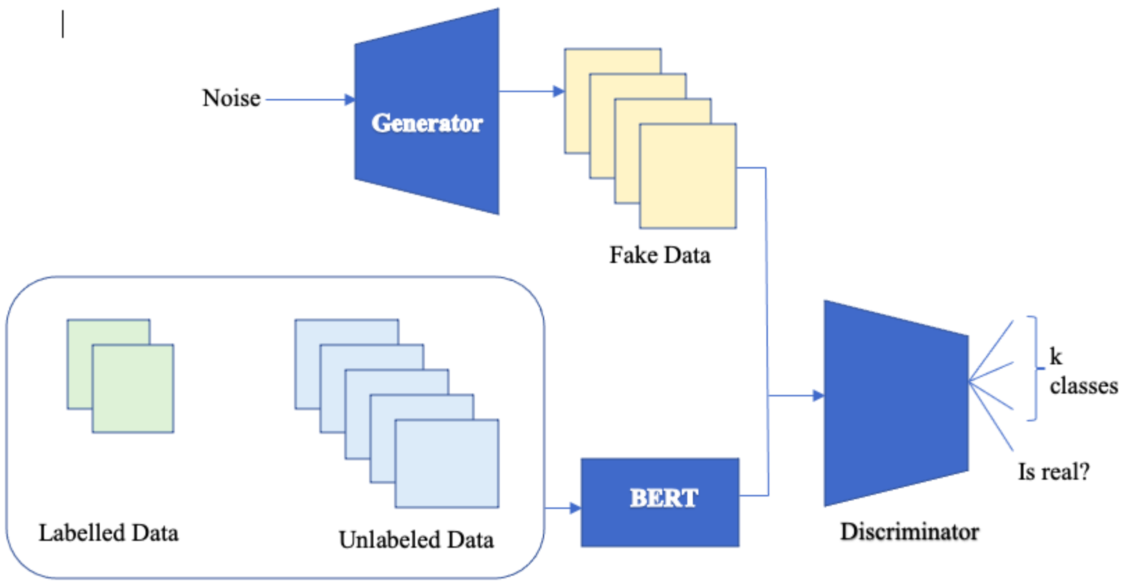


Figure 1: Architecture of proposed model

The discriminator identifies real samples as real data and assigns them a label from k categories. Similarly, the fake examples are categorized into k+1 category. The labeled data in the training dataset is used to train the discriminator. The training process tries to find the equilibrium between two competing losses i.e the loss from the generator and the discriminator. During back propagation, the unlabelled data contributes to the unsupervised loss of a discriminator if the examples are erroneously categorized into k + 1 classes. The discriminator loss is the summation of supervised and unsupervised loss. The supervised loss refers to incorrectly classifying the real examples among k categories. While unsupervised loss refers to identifying a real unlabelled example as fake and not identifying a fake example. The generator loss is given as a summation of feature matching and unsupervised loss. The goal of feature matching loss is to generate an example whose vector representation matches with a real example and as discussed unsupervised loss refers to the error that occurred by fake samples when correctly identified by the discriminator. For a given text, BERT would produce a vector representation of n+ 2 tokens that includes CLS and SEP. As per the study explained by Devlin et al. (2019) for sentence embedding CLS 11representation would be used. The weights of the BERT are changed in order to fine-tune its inner representation.

4.1 Dataset

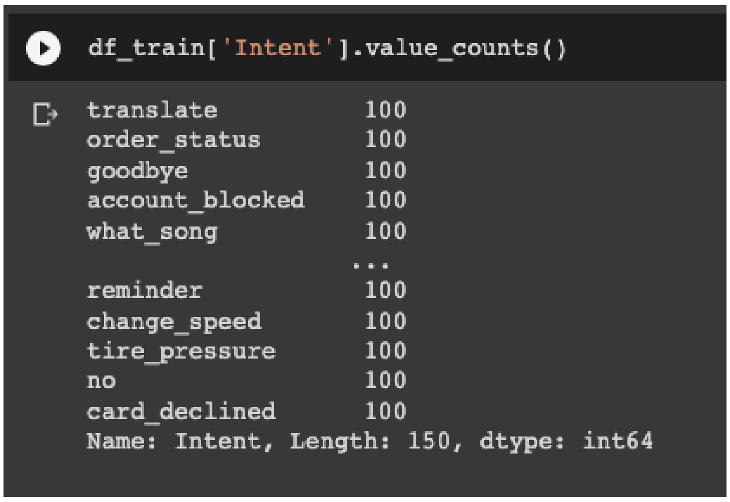
Intent detection is considered a crucial task in the development of task-oriented conversation systems yet very less data or very few datasets are built and available publicly. The datasets which have been studied earlier the Chatbot Corpus Braun et al. (2017) or SNIPS Coucke et al. (2018) has less number of classes (¡ 10) and the task of classification is oversimplified. This does not imitate the actual commercial system environment to have a conversation. The CLINC150 dataset by Larson et al. (2019) is available publicly. It comprises 150 different intents covering 10 domains and has 23700 records.

Having a single domain query dataset might train the algorithm easily but it would not correctly identify the queries which are from different domains. Looking at the conversational system that is getting popularity is not specific to any domain. It can answer queries from any domain. Considering that in mind the CLINC150 dataset is appropriate to carry out further study. It includes queries from 10 different domains which are listed in the below table. Each domain has 15 different intents corresponding to that domain. Each intent has 100 curated queries that were crowdsourced. The dataset was generated by asking crowd workers to either paraphrase “seed” phrases or respond to given scenarios e.g pretend you need to book a flight, what would you say? The dataset is balanced since every intent has an equal number of queries available. All the queries are written in the English language. All the queries in this dataset are single-intent samples. Figure Figure 2 describes the available intents.

The dataset also includes unlabeled records where only text queries are available. To have this file in a similar format as that of labeled records where all the queries are marked with a distinct intent, a new intent column was added to the unlabeled records file and was set to 'UNK_UNK' i.e intent unknown.

The test dataset has a total of 4500 records and covers the data for all the 150 intents. Each intent has 30 dedicated queries in the test dataset. Below figure Figure 3 provides a list of domains and corresponding intents.

Understanding textual ambiguity in natural language understanding is essential. To eliminate textual ambiguity syntactical context can be helpful. Bigrams are two words when used together create a distinct meaning. Similarly, trigrams are three words that create a distinct meaning when used together. Understanding the existence of bigrams and trigrams is important for a machine in order to understand the language the way humans do. This allows understanding of nuances of words and how the meaning changes



```
df_train['Intent'].value_counts()

translate          100
order_status       100
goodbye            100
account_blocked    100
what_song          100
                  ...
reminder           100
change_speed       100
tire_pressure      100
no                 100
card_declined      100
Name: Intent, Length: 150, dtype: int64
```

Figure 2: Number of available intent

when used in conjunction with other words. The N-grams are calculated here with the function CountVectorizer which converts a collection of text to a matrix of token counts. The below two figures Figure 4 and Figure 5 shows the bigrams and trigrams for the train dataset.

4.2 Data Preprocessing

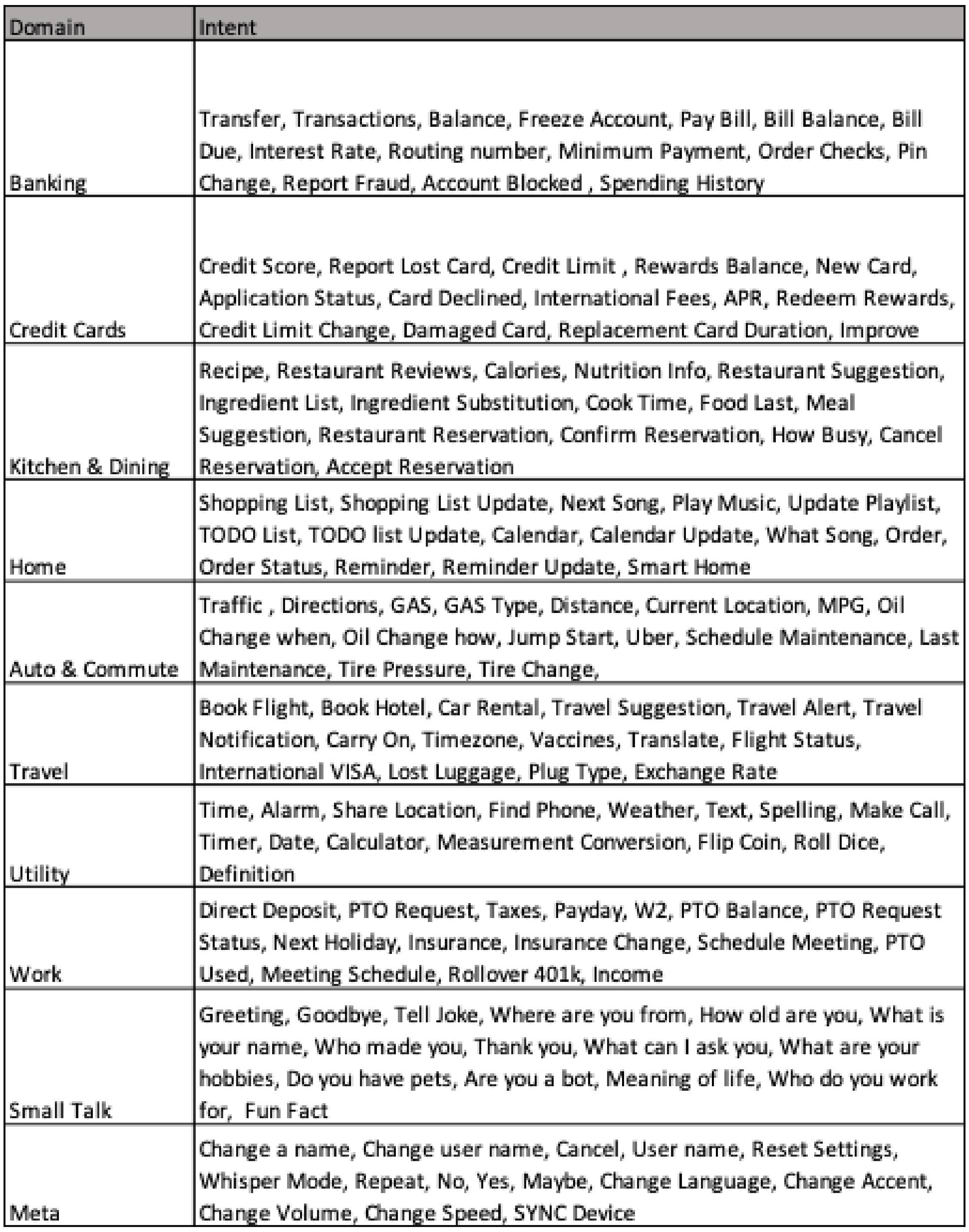
Data preprocessing is an important task to perform in natural language processing as well as for text analysis. It converts the human-readable language to a form which machine-readable and machine learning algorithms can be applied to that. A different preprocessing technique was applied to clean the data and the steps are listed below.

1. Lower Casing: The user query text for correctly classifying intent was first pre-processed by converting it in lower case.

2. Punctuation Removal: As observed in text analysis, one of the records contains a hashtag. It is necessary to get rid of the symbols from the text which also includes punctuations.

3. Stop Word Removal: The stop words are low information words e,g ‘a’, ’an’, ‘the’. These are commonly used words in a language. The removal of additional low information words allows focusing on important words. A predefined list of stopwords is available or it can also be customized.

4. Tokenization: It refers to breaking the sentence into a sequence of words. This process splits the text into tokens by a set of rules. The proposed methodology is based on a pre-trained model. It is required to use the associated pre-trained tokenizer. This is essential and makes sure that the given text is split in the same manner the way pretraining corpus is split, and it uses the same corresponding tokens-to-index while pretraining. The AutoTokenizer class loads the pre-trained tokenizer. It downloads vocab when the model is pre-trained. The tokenizer available by Hugginface for BERT pre-trained model was used. The output from the tokenizer creates a dictionary with three items



Domain	Intent
Banking	Transfer, Transactions, Balance, Freeze Account, Pay Bill, Bill Balance, Bill Due, Interest Rate, Routing number, Minimum Payment, Order Checks, Pin Change, Report Fraud, Account Blocked , Spending History
Credit Cards	Credit Score, Report Lost Card, Credit Limit , Rewards Balance, New Card, Application Status, Card Declined, International Fees, APR, Redeem Rewards, Credit Limit Change, Damaged Card, Replacement Card Duration, Improve
Kitchen & Dining	Recipe, Restaurant Reviews, Calories, Nutrition Info, Restaurant Suggestion, Ingredient List, Ingredient Substitution, Cook Time, Food Last, Meal Suggestion, Restaurant Reservation, Confirm Reservation, How Busy, Cancel Reservation, Accept Reservation
Home	Shopping List, Shopping List Update, Next Song, Play Music, Update Playlist, TODO List, TODO list Update, Calendar, Calendar Update, What Song, Order, Order Status, Reminder, Reminder Update, Smart Home
Auto & Commute	Traffic , Directions, GAS, GAS Type, Distance, Current Location, MPG, Oil Change when, Oil Change how, Jump Start, Uber, Schedule Maintenance, Last Maintenance, Tire Pressure, Tire Change,
Travel	Book Flight, Book Hotel, Car Rental, Travel Suggestion, Travel Alert, Travel Notification, Carry On, Timezone, Vaccines, Translate, Flight Status, International VISA, Lost Luggage, Plug Type, Exchange Rate
Utility	Time, Alarm, Share Location, Find Phone, Weather, Text, Spelling, Make Call, Timer, Date, Calculator, Measurement Conversion, Flip Coin, Roll Dice, Definition
Work	Direct Deposit, PTO Request, Taxes, Payday, W2, PTO Balance, PTO Request Status, Next Holiday, Insurance, Insurance Change, Schedule Meeting, PTO Used, Meeting Schedule, Rollover 401k, Income
Small Talk	Greeting, Goodbye, Tell Joke, Where are you from, How old are you, What is your name, Who made you, Thank you, What can I ask you, What are your hobbies, Do you have pets, Are you a bot, Meaning of life, Who do you work for, Fun Fact
Meta	Change a name, Change user name, Cancel, User name, Reset Settings, Whisper Mode, Repeat, No, Yes, Maybe, Change Language, Change Accent, Change Volume, Change Speed, SYNC Device

Figure 3: Domain wise intent clustering

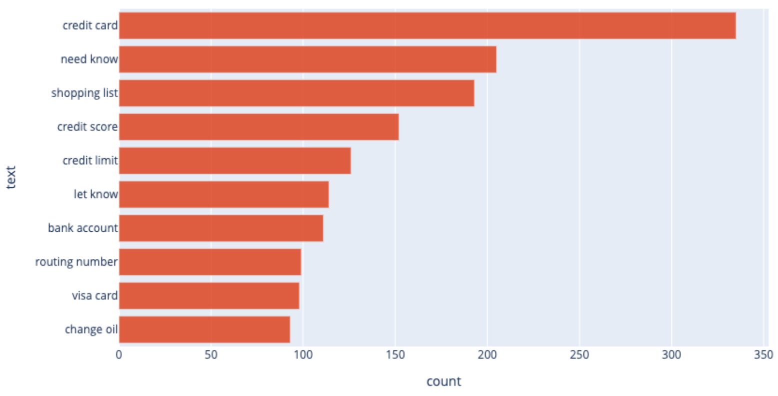


Figure 4: Bigram

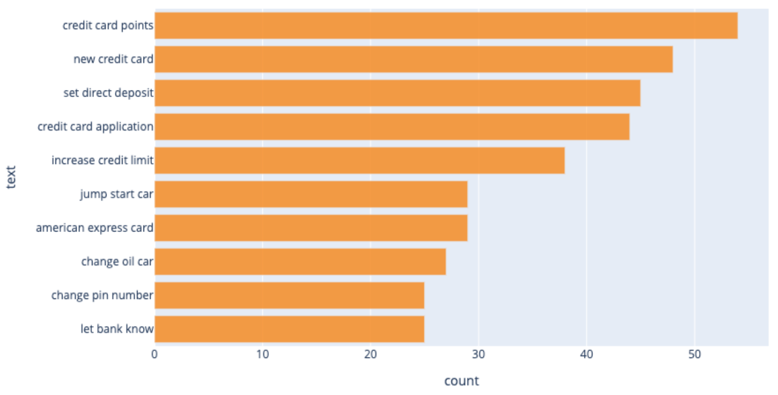


Figure 5: Trigram

   (a) Input_ids : Corresponding token indices of each token in the sentences
   (b) Attention Mask: adjust the input length by masking
   (c) Token_type_ids : which sequence a token belongs to incase if there's more than one sequence.

   The tokenizer itself adds two special tokens a classifier and a separator - CLS and SEP

5. Padding: All the queries in the dataset are not of the same size. This causes an issue when the sentences are converted into tokens. The input model needs to have a uniform shape. Padding helps in adding additional special tokens to have tensors in the rectangular shape. This is achieved by setting the padding to the max sequence length.

6. Truncation: The max length considered for a query is 64. If a sentence is smaller, it would be padded until max length. If the sentence has a greater length than the max length which would be too long to handle for a model, then it would be truncated to a shorter length. This is achieved by setting the truncation parameter to TRUE.

7. Tensor: The input to the model is given in the form of a vector which is a Tensor. The output from the tokenizer is converted to a tensor array.

8. Tensor Dataset: A training dataset was built by combining all the tensor arrays of input_ids and label_ids.

4.3 Train and Test Split

The proposed study is based on the use of supervised and unsupervised data. It compares how much minimum supervised data is required to train the model and achieve the highest accuracy and classify the intents correctly. The entire training dataset is divided into incremental portions. These new .tsv files were stored in different folders on Gdrive and the training process is repeated for every folder and the corresponding results were stored in evaluated.

The below table shows the different data divisions created from the full data available. The first row suggests that only 10% of labeled data was considered i.e 1500 records and 90% of unlabeled records i.e 13500 records. The next data files were created by adding another 10% of labeled data and decreasing 10% of unlabeled data keeping the total number of records 15000 constant. This process is repeated until we reach 90% of labeled data and 10% of unlabeled data. Since the data is stored in different folders, the training process for different data files can be easily carried out just by changing the name of the folder.

In Table 1 some portion of labeled data and unlabeled data was combined together to create the training dataset.

4.4 Model

4.4.1 Baseline Model : BERT

The emerging advancement in the field of natural language processing has improved its performance just by relying on simple input representation. The transfer learning

Table 1: Training Data Split

Labelled Record per each intent	**Total # of Labelled Records**	**Labelled Record per each intent**	**Total # of Unlabelled Records**
10	1500	90	13500
20	3000	80	12000
30	4500	70	10500
40	6000	60	9000
50	7500	50	7500
60	9000	40	6000
70	10500	30	4500
80	12000	20	3000
90	13500	10	1500

method involves making use of pre-trained models instead of making it from scratch and fine-tuning it for a specific task to achieve state-of-the-art performance using the vast amount of data. The capacity of BERT was explored by Haode et al. (2021) by fine-tuning it with 1000 labeled data – IntentBERT. This method used IntentBert for feature extraction and then applied a parametric classifier such as a logistic or non-parametric nearest neighbor. The method was compared with standard BERT, TOD-BERT. The results proved the high effectiveness of IntentBERT for a few shit intent detection and the ability to generalize across different domains.

4.4.2 Proposed Model : Semi supervised GAN-BERT

This research paper proposed to study the impact of the state-of-the-art BERT model along with the highly used unsupervised Generative Adversarial Model (GAN) on the intent classification tasks. This process focuses on fine-tuning the BERT model for unsupervised data with help of adversarial settings. The transformer-based BERT has the capability to produce high-quality vector representation of words or tokens and to help incorporate the vector representation of unlabelled material so the network can generalize it for the final task.

Previously Croce et al. (2020) investigated the Kernal-based GAN model. The implementation was based on projecting the sentences in low-dimensional embeddings. The approximation of implicit space was given by the Semantic tree kernel function. The semi-supervised GAN approach is preferred because it improves the quality of the multi-layer perceptron used and it does change the input representation. Though the GAN models are popularly used for image inputs, (Devlin et al., 2019) proved they can be adopted in combination with BERT for text-based datasets.

The design structure of the proposed model has three elements namely a BERT model, a generator and a discriminator. The BERT takes input from the training dataset, a random noise was given to the generator, and the output produced from 1st two components were given as input to the discriminator. BERT basically used to vector transform the input as it has the ability to produce good inner representation. The discriminator is supposed to classify the intents. One might ask about the need for a generator here, as the real examples are already present. The following section answered this question.

The training of the generator and discriminator ensures that the newly created examples lie close to latent space and the discriminator was trained to recognize them. If the discriminator was only trained on the training input dataset, at some point the training model learns all the patterns and finalised the model. This pattern would be learned by the generator and the discriminator can be easily fooled. This was best explained with a truck example. With given real data a discriminator learns there's a white patch on the right corner of every truck. When the generator created an image of a truck with the same white patch, the discriminator classified it as real. In contrast, when the discriminator was trained on a combination of real and fake data, it learned complex patterns. It improved the quality by understanding the data beyond the raw patterns.

5 Implementation

5.1 Baseline Approach

The implementation of the proposed method is carried out in google collab which allows using GPU for free. The implementation starts with initializing the session and setting up the runtime type as GPU. Once the GPU is set, import the GDrive on google collab to access the data files that have been created in pre-processing step.

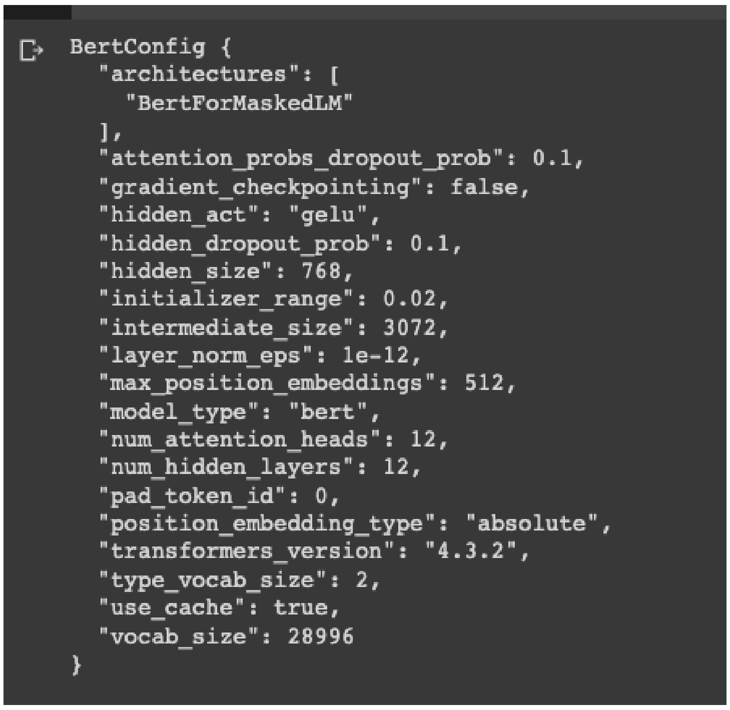
Set the datafile path correctly to use the labeled, unlabelled, and test data. Load the examples from the .tsv files and convert them into a data frame of two columns namely Label and text. A list of unique 150 labels was created and numbered since machines only deal with numbers. This is stored in a label_map.

The methodology is based on a pre-trained BERT model. The pre-trained model is downloaded from the hugging-face library. The compatible tokenizer is also downloaded from the library. All the imported labeled and unlabelled records are tokenized using the BERT tokenizer, padded as needed, and converted into tensors. The output arrays of all the tensors are combined to create the trained dataset which would be given as an input to the BERT model. The starting point of the training approach is BERT-base. The configuration of the BERT model used was given below in figure Figure 6. The GAN was implemented by extending applied BERT.

The Generative adversarial network comprises two modules, a generator, and a discriminator. The number of hidden neurons should be between the size of the input layer and the size of the output layer. A number of hidden layers in the generator, each of the size of output space. The generator is defined as a multi-layer perceptron with one hidden layer activated by the RELU function. The generator module takes the noise vector as an input drawn from a normal distribution (0,1). The noise_size is set to 100 as refereed in Padala et al. (2020). The generator then transforms this noise into a 786-dimensional vector which is used as a fake example in methodology. By introducing noise, we can get the GAN to produce a wide variety of data, sampling from different places in the target distribution. every time choosing random noise will generate new data. If random noise size is chosen every time then the model will generate the same data.

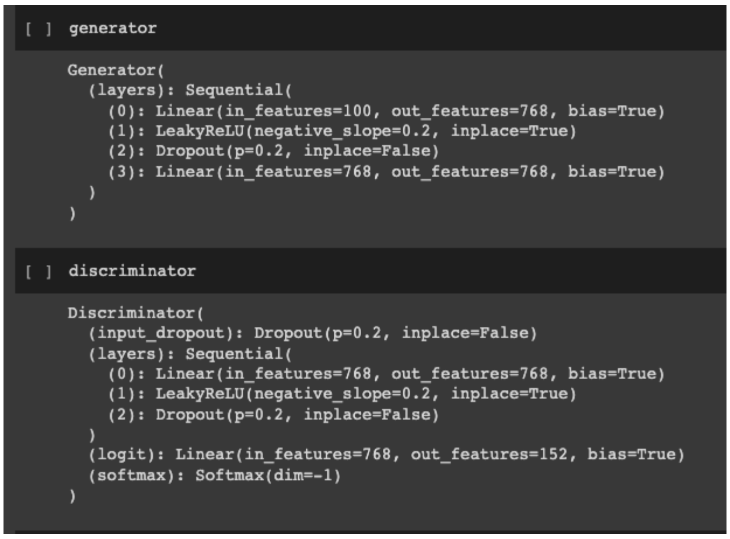
The discriminator is also defined with one hidden layer activated by the RELU function. It is followed by a softmax layer for the final classification task. The dropout rate of 0.1 is applied to both after the hidden layer. As per research Mordido et al. (2018), the quality and variation of the produced samples increase while using the dropout rate values of 0.1 and 0.5 across all the different-sized discriminator sets. The configuration of Generator and discriminator was given by below figure Figure 7.

All three modules transformer, generator, and discriminator are put into training. For each record in the training data loader, the input was unpacked. This real data is



```
BertConfig {
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.3.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```

Figure 6: Bert Configuration



```
[ ] generator

Generator(
  (layers): Sequential(
    (0): Linear(in_features=100, out_features=768, bias=True)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=768, out_features=768, bias=True)
  )
)

[ ] discriminator

Discriminator(
  (input_dropout): Dropout(p=0.2, inplace=False)
  (layers): Sequential(
    (0): Linear(in_features=768, out_features=768, bias=True)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Dropout(p=0.2, inplace=False)
  )
  (logit): Linear(in_features=768, out_features=152, bias=True)
  (softmax): Softmax(dim=-1)
)
```

Figure 7: Generator and Discriminator Configuration

encoded into the transformer to generate the output. At the same time generator is being run to create fake samples similar to real input data. The discriminator is being run on output from the transformer and fake generated data to classify the data. The output from the discriminator was then separated from real and fake data. The GAN model is evaluated on the basis of generator and discriminator loss. The loss is calculated at every step.

After the completion of every epoch run, the performance measure of the model is evaluated by running the prediction on the test data loader.

The optimizer used in the implementation was the Adam optimizer. The results of the Adam optimizer are generally better than every other optimization algorithm, have faster computation time, and require fewer parameters for tuning. The research study by Ruder (2016) explained that the Adam optimizer has given high performance compared to other optimizers giving an optimized gradient descent. A scheduler was used to adjust the learning rate. The scheduler was initialized with a constant learning rate preceded by a warmup period. The warmup period starts with a learning rate much smaller than the defined rate and increases over a few epochs until it reaches the defined learning rate. Warmup was used to avoid the overfitting of the model.

6 Evaluation

This section describes the different experiments carried out and the results obtained. The results are analyzed to determine the reliability of the proposed model. It is evaluated by running different tests and comparing the result set.

6.1 Experiment

For the first experiment, the model was trained for 10 epochs. After evaluating the results, it was observed that the maximum accuracy for that run was achieved at the 8th epoch and it started dropping from the 9th epoch onwards. The model started to overfit the data. The number of correctly predicted records started reducing. For every experiment, the time taken for each step is calculated. It calculates the average discriminator's and generator's loss at each step. The evaluation steps calculate the model accuracy by comparing predicted records against the test dataset. All the statistics are being written in the logs. This was repeated for all data files created.

The implementation of intent classification is optimized to improve the model performance. It involves the use of Adam optimizer. The results of the Adam optimizer are generally better than every other optimization algorithm, have faster computation time, and require fewer parameters for tuning. A scheduler was introduced to adjust the learning rate. Create a schedule with a constant learning rate preceded by a warmup period during which the learning rate increases linearly between 0 and the initial learning rate set in the optimizer. Supposedly, warm-up fights early overfitting. initially, high learning (followed by slow decay) improves the performance of a neural network. It is argued that a low learning rate makes the network learn high-frequency patterns such as noise before low-frequency patterns and so an initially high learning rate supposedly fights early overfitting.

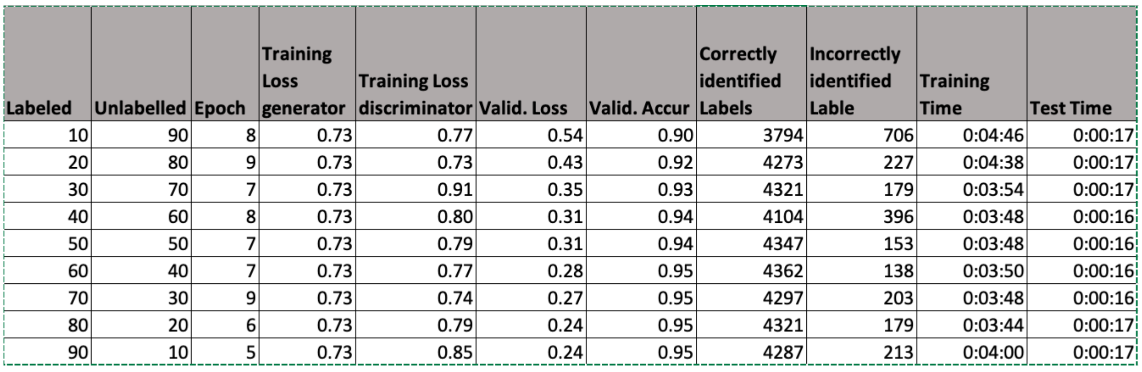
6.2 Results

The experiments were characterized by different training scenarios i.e variation in a number of labeled and unlabeled data. The results obtained from the experiments are explained below. The measures of the approach to support the development of the proposed deep learning model when exposed to a few examples for intent classification is reported. The performance is measured by comparing statistical results found after every experiment. The methodology implemented GAN-BERT by extending the BERT-base model.

To elaborate on the methodology, the Generator is implemented with one hidden layer. It is activated by the RELU function. The input passed to the generator consists of the noisy vector drawn from the normal distribution. The output obtained from this results in 768- a dimensional vector which is nothing but fake data examples. The discriminator is also defined with one hidden layer activated by the RELU function. This is followed by a softmax layer for final prediction. The hidden layer is followed by the dropout rate of 0.1.

The training was carried out in increasing sets of annotated samples to understand the impact of the labeled data. In order to measure the performance, the training sample started with the 10% of annotated data and it was increased gradually. The balance between labeled and unlabeled data was maintained by keeping the number of samples the same in the training set. The labeled training and testing dataset consist of 15000 and 4500 records respectively spread across 150 in-tents. Along with that, 13500 unlabeled records are available for semi-supervised learning. As discussed earlier, to implement and comprehend the performance of few-shot learning, a training dataset was created by combining labeled and unlabeled data. For the first dataset, 10% of labeled data and 90% of unlabeled data were considered. For the next dataset labeled records were increased by 10% and unlabeled records were reduced by 10%. Each set of data was trained for 10 epochs. Below tables shows the maximum accuracy achieved by the model for different sets.

Out of all the 90 runs carried out, the records with maximum accuracy for each dataset are listed below in figure Figure 8 . It shows that the number of epochs required to achieve the accuracy is different for each training dataset. The common observation is as the labeled data is increased, the number of epochs required to train the model is less to achieve the same level of accuracy. The quality of the training data highly affects the prediction.



Labeled	Unlabelled	Epoch	Training Loss generator	Training Loss discriminator	Valid. Loss	Valid. Accur	Correctly identified Labels	Incorrectly identified Lable	Training Time	Test Time
10	90	8	0.73	0.77	0.54	0.90	3794	706	0:04:46	0:00:17
20	80	9	0.73	0.73	0.43	0.92	4273	227	0:04:38	0:00:17
30	70	7	0.73	0.91	0.35	0.93	4321	179	0:03:54	0:00:17
40	60	8	0.73	0.80	0.31	0.94	4104	396	0:03:48	0:00:16
50	50	7	0.73	0.79	0.31	0.94	4347	153	0:03:48	0:00:16
60	40	7	0.73	0.77	0.28	0.95	4362	138	0:03:50	0:00:16
70	30	9	0.73	0.74	0.27	0.95	4297	203	0:03:48	0:00:16
80	20	6	0.73	0.79	0.24	0.95	4321	179	0:03:44	0:00:17
90	10	5	0.73	0.85	0.24	0.95	4287	213	0:04:00	0:00:17

Figure 8: Accuracy table

The results obtained from this method were stored in a dataframe. A few different graphs were plotted to comprehend the results. All the graphs were given below.

1. Accuracy Vs Epoch

   The below graph in figure Figure 9 was plotted against the accuracy achieved by end of every epoch for each experiment carried out for different data files.

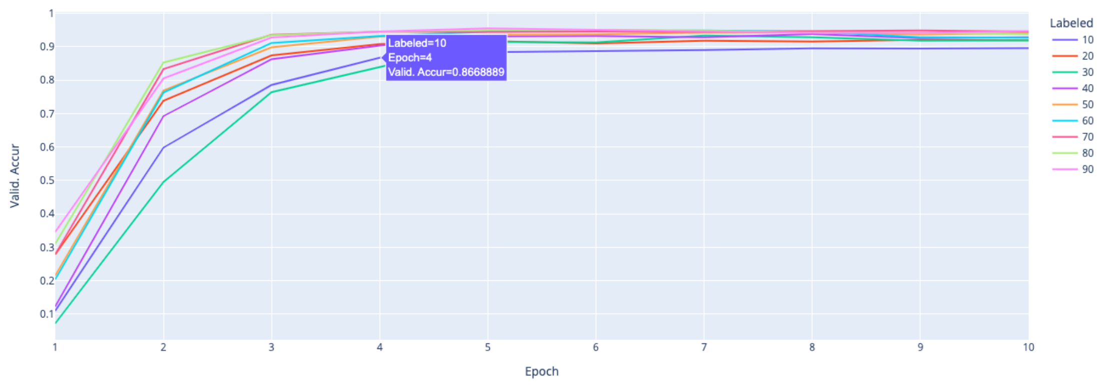


Figure 9: Accuracy obtained over Epochs

   This graph shows the following observation:

   (a) The highest annotated training set (90% labeled, 10% unlabeled) started with a better performance as compared to other training sets at epoch 1

   (b) The maximum accuracy was achieved by the last case where 90% labeled and 10 and unlabeled data was considered. The accuracy achieved was 95.5% at 5th epoch.

   (c) The case with 80-20% of labeled and unlabeled data has shown a spike in accuracy change in the initial 3 epochs. It achieved an accuracy of 93.42% at the end of the 3rd epoch.

   (d) The case where a reduced number of labeled data was involved, 30-70%, achieved accuracy close to the above-mentioned case, 93.31% by training the model for 7 epochs.

   (e) Close to the accuracy mentioned in the above case, 60-40% case achieved the accuracy of 93.2% in 4 epochs.

   (f) The case where only 10% of labeled data was involved, took 10 epochs to achieve the accuracy of 89.5%.

   (g) The accuracy curve does not seem to be overfitting the data.

   (h) The graph summarizes as the amount of annotated data is increased, it takes fewer epochs of training to achieve the same accuracy.

2. Valid Loss vs Epoch

   This is one of the most used plots during the training of a neural network. This graph describes the training process and direction in which the network learns.

Here the loss is logged at the end of each epoch. The optimization process tries to minimize the loss over training, hence lower the value, the better the performance. This ensures the loss is calculated across every data item and gives a quantitative loss measure. The curve plotted against every iteration might give the loss on a small portion of the dataset used. A graph of loss calculated at every epoch was plotted below in the figure Figure 10

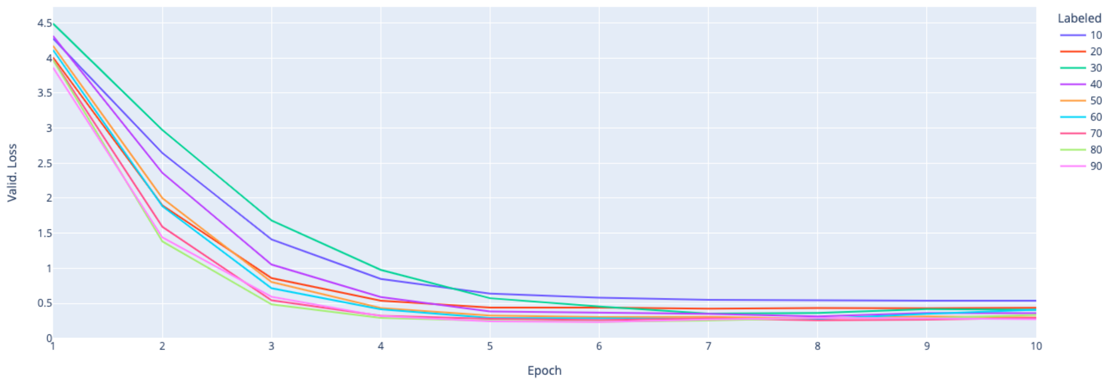


Figure 10: Valid Loss over Epochs

This graph shows the following observation:

(a) The loss decreased exponentially until the 5th epoch, then a steady trend has been observed for every experiment.

(b) Highest annotated training sets seems to have reduced the loss by more than 90% by the end of the 6th epoch.

(c) The lowest annotated training sets took more epochs to reduce the loss using the optimizer.

(d) The graph summarizes, that the optimizer has reduced more than 95% of loss over training the model for 10 epochs.

3. Loss of Generator And Loss of Discriminator

   This graph plots the loss of generator and discriminator over epoch for all the experiments carried out and given below in figure Figure 11. For the initial epoch in the training phase, the losses oscillate as both the generator and the discriminator try to find the equilibrium. After this point, both the losses vary around a point which indicates stable learning.

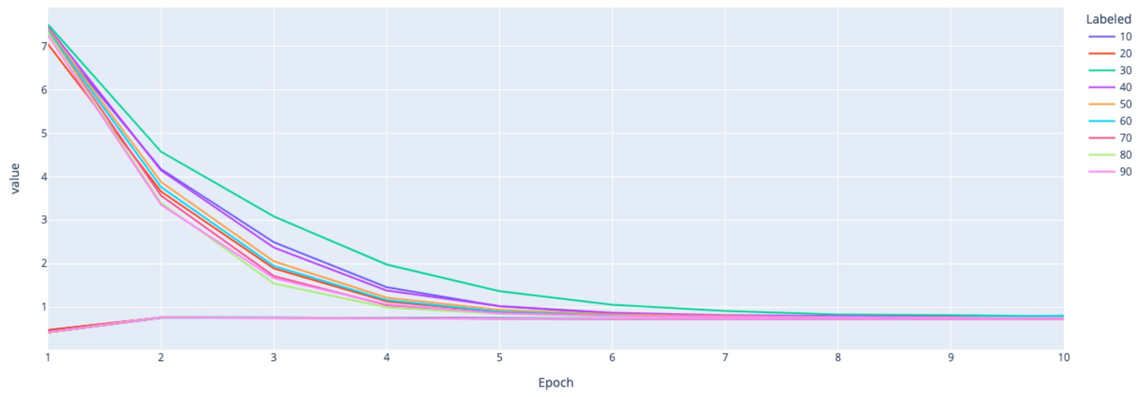


Figure 11: Loss of Generator and Loss of Discriminator

During the experiments, after every epoch run, the evaluation is carried out against the test dataset which is common for all the datafiles. The number of correctly and incorrectly identified labels are printed at the end of each epoch. The graphs are plotted for both the values and are given below.

4. Correctly Identified Labels

   The amount of correctly identified intents were shown in below figure Figure 12.

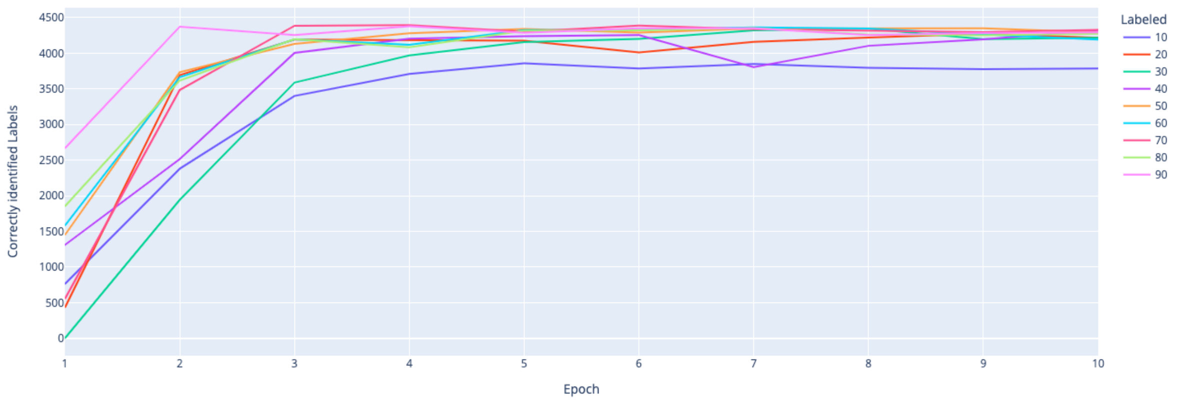


Figure 12: Correctly Identified Intents

The finding from the graphs was as follows:

(a) The highest annotated dataset has predicted more than 50% of the test data at 1st epoch wherein the datafile with 30-70 of them did not identify a single record correctly.

(b) A steady parallel growth was observed between performances of all the test cases except 20-80% and 70-30% has shown a massive conversion of identifying the labels correctly between epoch 2 and epoch 3.

(c) The max number of correct predictions happened at epoch 3 for 70-30% datafile.

(d) When the model was trained for other data files, it successfully identified more than 90% of test data by end of 10 epochs except for 10-90% file, for which only 85% data was successfully identified.

5. Incorrectly Identified Labels

The amount of incorrectly identifies intents was shown in below figure Figure 13.

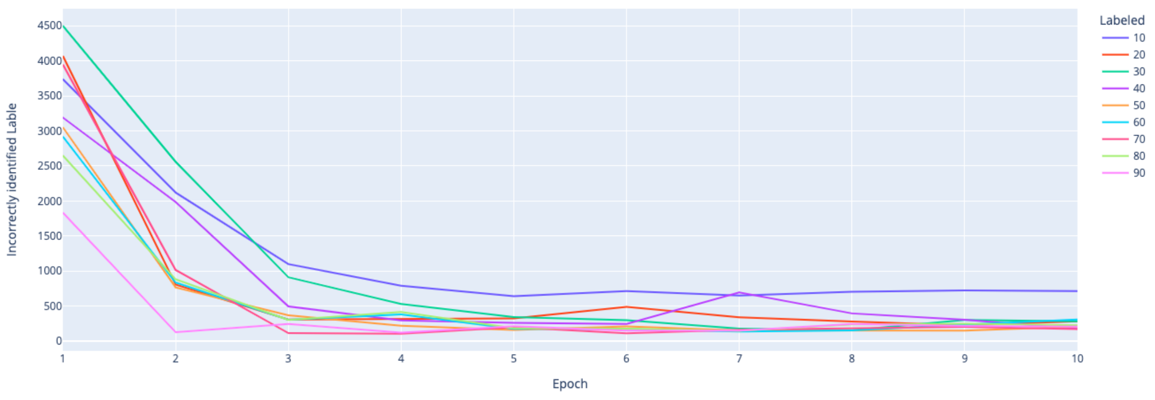


Figure 13: Incorrectly Identified Intents

The finding from the graphs was as follows:

(a) The graph shows, that the model trained for 30-70% data has shown maximum incorrectly predicted data at epoch data. But the training was improved over the epochs and the number of records reduced exponentially over the next 2 epochs.

(b) The model when trained for 40-60%, it identified more than 90% data correctly at epoch 6 but it reduced at epoch 7 and increased again from epoch 8 onwards.

6. Confusion Matrix

The result obtained from the models for all the datafiles at each epoch was examined and the datafile containing 60% labeled data and 40% of unlabeled data has shown 94.44% accuracy and classified 97% of test records correctly. The confusion matrix for this scenario was plotted and mentioned below in figure Figure 14. An evident diagonal line proved that a good amount of intents were classified correctly for all the intent types. A few small dots in between shows incorrectly identified intents.

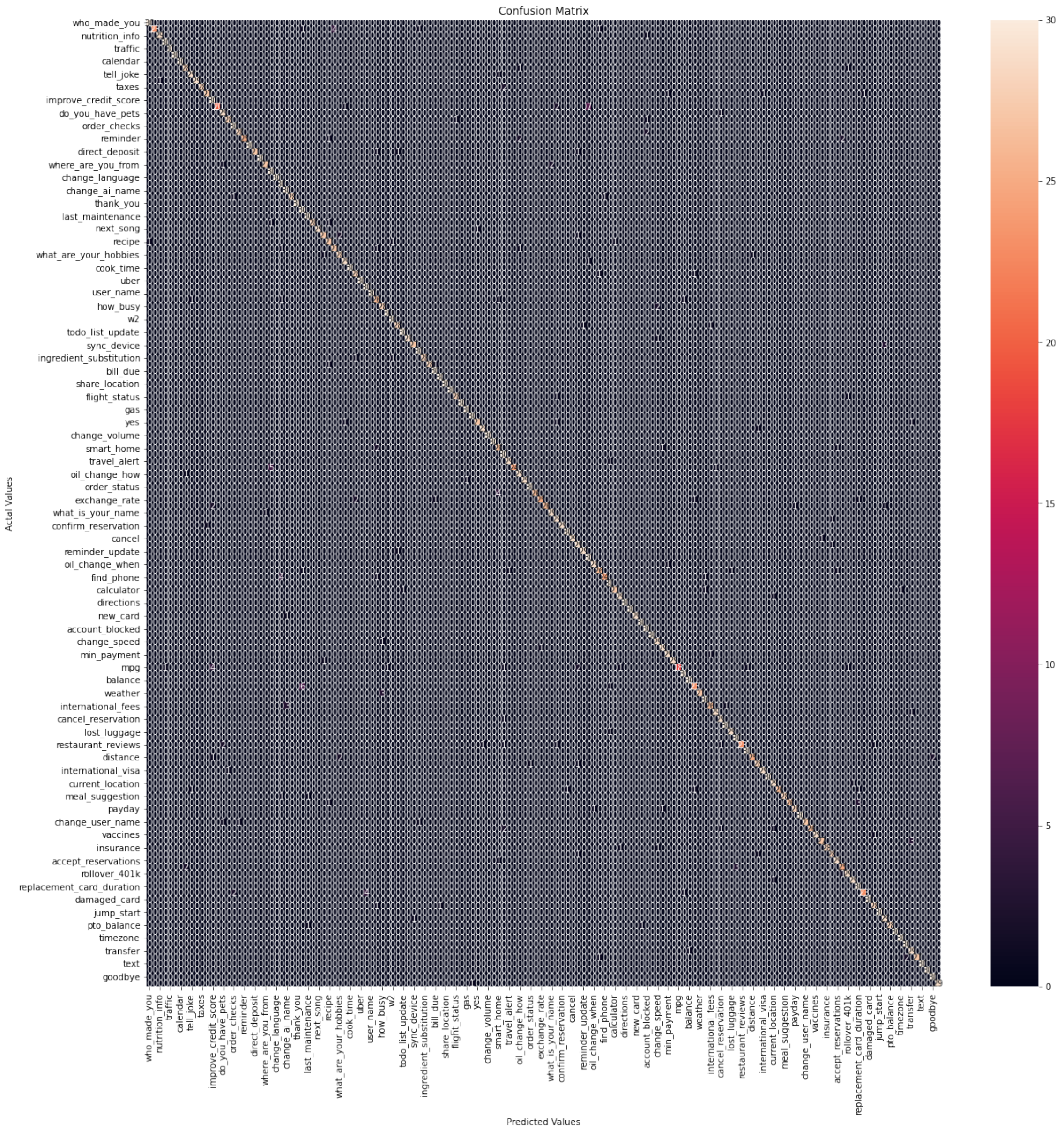
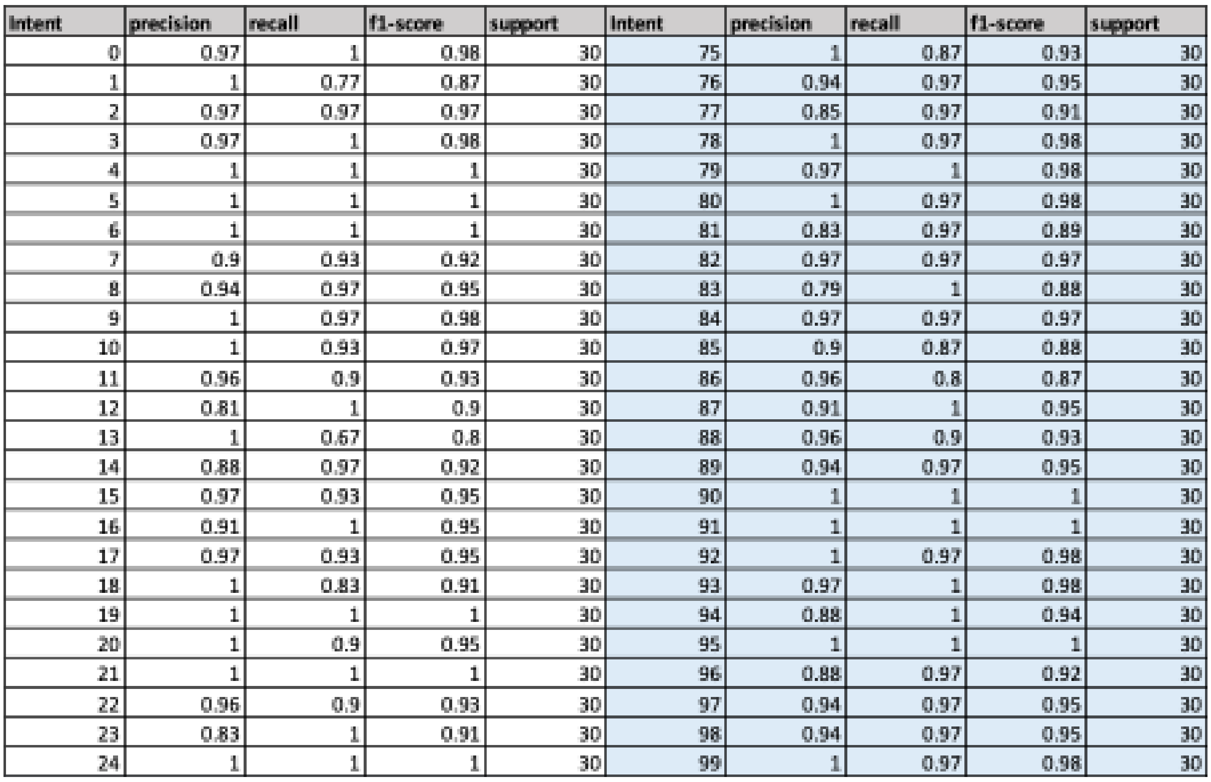


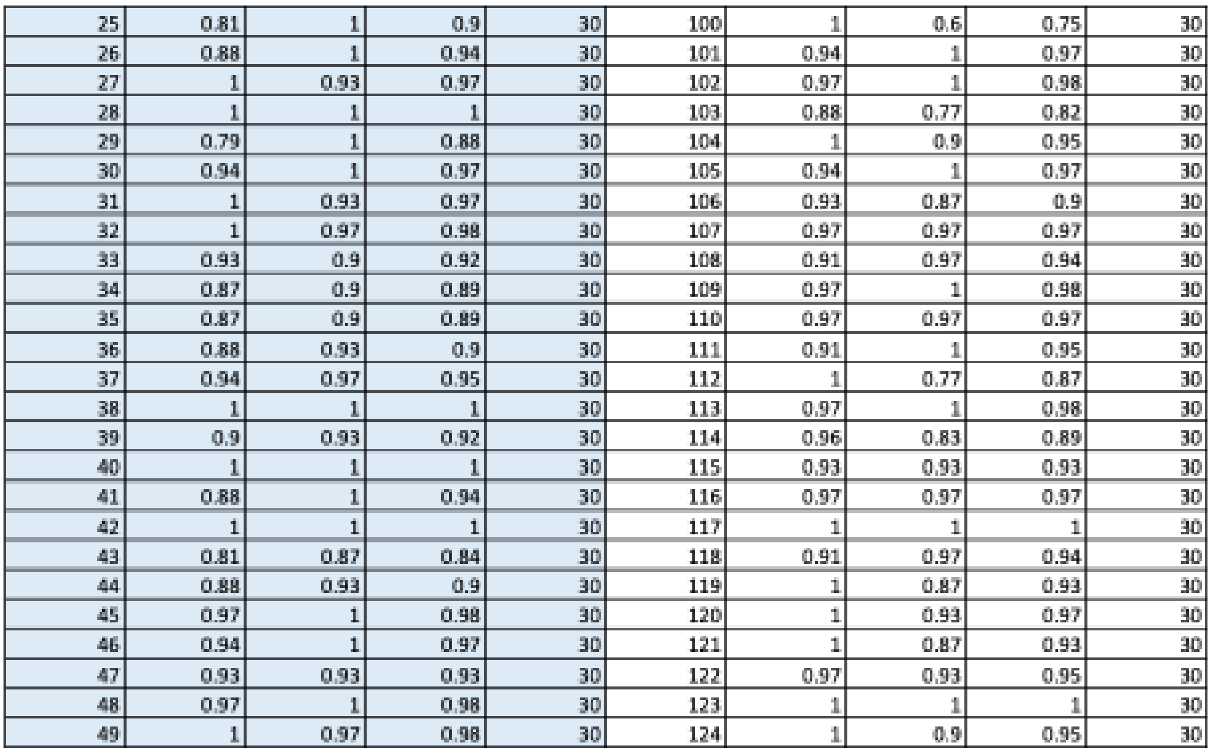
Figure 14: Confusion Matrix

7. F1 Score

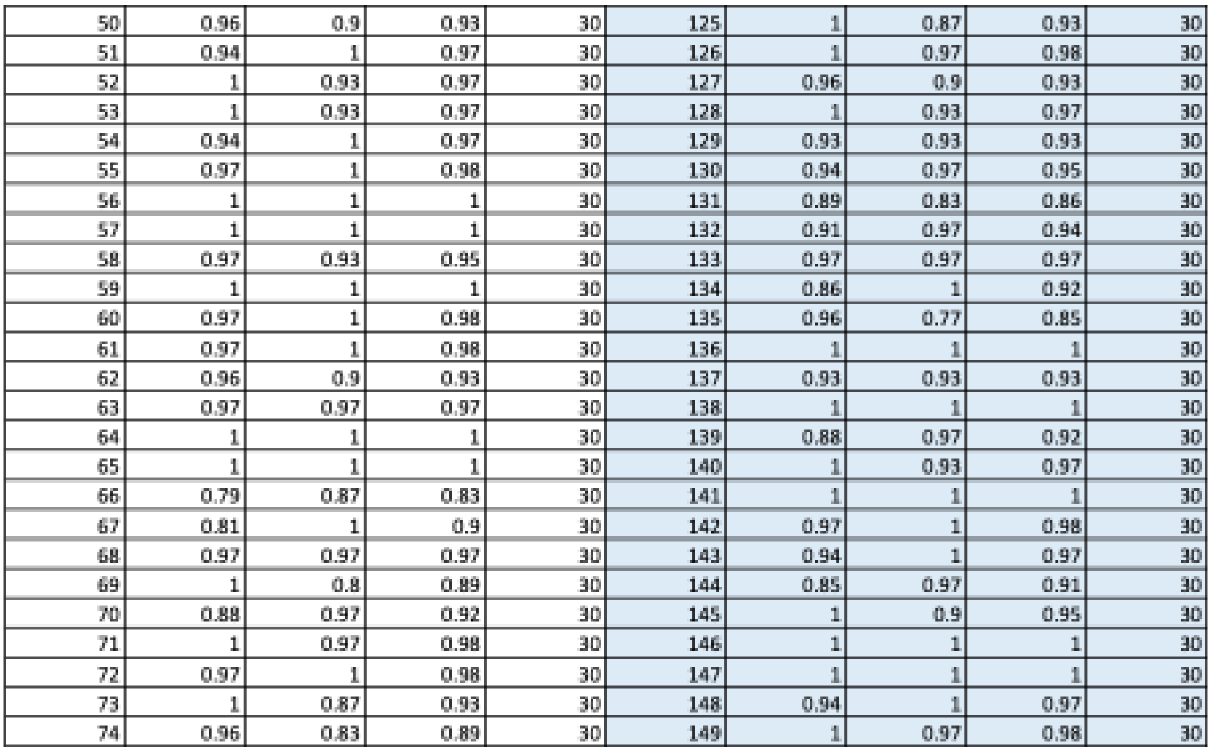
   F-measure is an evaluation metric that suggests the performance of the model. It is calculated as the harmonic mean of the precision and recall value. The F1 score obtained for a datafile containing 60% labeled data and 40% was given below in Figure 15



Intent	precision	recall	f1-score	support	Intent	precision	recall	f1-score	support
0	0.97	1	0.98	30	75	1	0.87	0.93	30
1	1	0.77	0.87	30	76	0.94	0.97	0.95	30
2	0.97	0.97	0.97	30	77	0.85	0.97	0.91	30
3	0.97	1	0.98	30	78	1	0.97	0.98	30
4	1	1	1	30	79	0.97	1	0.98	30
5	1	1	1	30	80	1	0.97	0.98	30
6	1	1	1	30	81	0.83	0.97	0.89	30
7	0.9	0.93	0.92	30	82	0.97	0.97	0.97	30
8	0.94	0.97	0.95	30	83	0.79	1	0.88	30
9	1	0.97	0.98	30	84	0.97	0.97	0.97	30
10	1	0.93	0.97	30	85	0.9	0.87	0.88	30
11	0.96	0.9	0.93	30	86	0.96	0.8	0.87	30
12	0.81	1	0.9	30	87	0.91	1	0.95	30
13	1	0.67	0.8	30	88	0.96	0.9	0.93	30
14	0.88	0.97	0.92	30	89	0.94	0.97	0.95	30
15	0.97	0.93	0.95	30	90	1	1	1	30
16	0.91	1	0.95	30	91	1	1	1	30
17	0.97	0.93	0.95	30	92	1	0.97	0.98	30
18	1	0.83	0.91	30	93	0.97	1	0.98	30
19	1	1	1	30	94	0.88	1	0.94	30
20	1	0.9	0.95	30	95	1	1	1	30
21	1	1	1	30	96	0.88	0.97	0.92	30
22	0.96	0.9	0.93	30	97	0.94	0.97	0.95	30
23	0.83	1	0.91	30	98	0.94	0.97	0.95	30
24	1	1	1	30	99	1	0.97	0.98	30



25	0.81	1	0.9	30	100	1	0.6	0.75	30
26	0.88	1	0.94	30	101	0.94	1	0.97	30
27	1	0.93	0.97	30	102	0.97	1	0.98	30
28	1	1	1	30	103	0.88	0.77	0.82	30
29	0.79	1	0.88	30	104	1	0.9	0.95	30
30	0.94	1	0.97	30	105	0.94	1	0.97	30
31	1	0.93	0.97	30	106	0.93	0.87	0.9	30
32	1	0.97	0.98	30	107	0.97	0.97	0.97	30
33	0.93	0.9	0.92	30	108	0.91	0.97	0.94	30
34	0.87	0.9	0.89	30	109	0.97	1	0.98	30
35	0.87	0.9	0.89	30	110	0.97	0.97	0.97	30
36	0.88	0.93	0.9	30	111	0.91	1	0.95	30
37	0.94	0.97	0.95	30	112	1	0.77	0.87	30
38	1	1	1	30	113	0.97	1	0.98	30
39	0.9	0.93	0.92	30	114	0.96	0.83	0.89	30
40	1	1	1	30	115	0.93	0.93	0.93	30
41	0.88	1	0.94	30	116	0.97	0.97	0.97	30
42	1	1	1	30	117	1	1	1	30
43	0.81	0.87	0.84	30	118	0.91	0.97	0.94	30
44	0.88	0.93	0.9	30	119	1	0.87	0.93	30
45	0.97	1	0.98	30	120	1	0.93	0.97	30
46	0.94	1	0.97	30	121	1	0.87	0.93	30
47	0.93	0.93	0.93	30	122	0.97	0.93	0.95	30
48	0.97	1	0.98	30	123	1	1	1	30
49	1	0.97	0.98	30	124	1	0.9	0.95	30



50	0.96	0.9	0.93	30	125	1	0.87	0.93	30
51	0.94	1	0.97	30	126	1	0.97	0.98	30
52	1	0.93	0.97	30	127	0.96	0.9	0.93	30
53	1	0.93	0.97	30	128	1	0.93	0.97	30
54	0.94	1	0.97	30	129	0.93	0.93	0.93	30
55	0.97	1	0.98	30	130	0.94	0.97	0.95	30
56	1	1	1	30	131	0.89	0.83	0.86	30
57	1	1	1	30	132	0.91	0.97	0.94	30
58	0.97	0.93	0.95	30	133	0.97	0.97	0.97	30
59	1	1	1	30	134	0.86	1	0.92	30
60	0.97	1	0.98	30	135	0.96	0.77	0.85	30
61	0.97	1	0.98	30	136	1	1	1	30
62	0.96	0.9	0.93	30	137	0.93	0.93	0.93	30
63	0.97	0.97	0.97	30	138	1	1	1	30
64	1	1	1	30	139	0.88	0.97	0.92	30
65	1	1	1	30	140	1	0.93	0.97	30
66	0.79	0.87	0.83	30	141	1	1	1	30
67	0.81	1	0.9	30	142	0.97	1	0.98	30
68	0.97	0.97	0.97	30	143	0.94	1	0.97	30
69	1	0.8	0.89	30	144	0.85	0.97	0.91	30
70	0.88	0.97	0.92	30	145	1	0.9	0.95	30
71	1	0.97	0.98	30	146	1	1	1	30
72	0.97	1	0.98	30	147	1	1	1	30
73	1	0.87	0.93	30	148	0.94	1	0.97	30
74	0.96	0.83	0.89	30	149	1	0.97	0.98	30

Figure 15: F1 Score Table

In Table 2 performance of the model was given for 60-40 datafile.

Table 2: Evaluation Score.

Accuracy			**0.95**	**4500**
macro avg	0.95	0.95	0.95	4500
weighted avg	0.95	0.95	0.95	4500

6.3 Discussion

The results obtained from different experiments provide important data that may help to justify the proposed research question. The semi-supervised model yielded a range of outputs based on the variation of the data provided. The results may also be compared against the previously applied techniques to get insights. As mentioned in the paper that utilized BERT and got an accuracy of 96.2% accuracy on the same dataset. The result table shows that though none of the experiments produced accuracy greater than 96.2%, even the case where only 10 labeled examples were considered gave a reasonable performance of 89.5%. The common observation which was discussed in the result section was, that as the number of labeled examples increased, the performance of the method was seen to increase. There's not much difference between the performance obtained when the model was trained with 50, and 60,70 labels. In other words, the performance obtained by 70 labels can also be achieved by 50 labels by adjusting the epoch runs.

7 Conclusion and Future Work

The proposed research methodology has shown great potential in semi-supervised learning for the targeted task of intent classification. The has shown great results with low availability of training data. However, this is only one of the approaches to handling the unsupervised data and different few-shot learning methods could be combined to compare the results. In the real case scenario, while developing digital assistance a lot of different intents need to be considered. The study has proved that the implemented method is extremely lightweight in terms of resources, easy to implement, and can be reused. This can be carried out on a standard laptop's CPU in several minutes. This quality allows or facilitates the development of intent classifiers without having access to large computational resources, which increases the fairness and equality in the research field.

8 Acknowledgements

The author would like to thank Dr. Abubakr Siddig for his guidance and continuous support throughout the project. The author would also like to express their gratitude to the National College of Ireland for immense support in terms of research facilities and infrastructure provided.

References

Abedin, Afia, Mamun, Islam, A., Nowrin, Jahan, R., Chakrabarty, Amitabha, Mostakim, Moin, Naskar and Sudip. (2021). A deep learning approach to integrate human-level understanding in a chatbot.

BAHA, T. A., HAJJI, M. E., ES-SAADY, Y. and FADILI, H. (2022). Towards highly adaptive edu-chatbot.

Braun, D., Mendez, A. H., Matthes, F., Langen, M. and Yu, P. S. (2017). Evaluating natural language understanding services for conversational question answering systems, *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* .

Chiu, Yu-Ching, Bi, N., Tsai, Richard and Tzong-Han (2021). Enhancing multi-modal intent classification in assembly scenarios through multi-task learning., *20th IEEE International Conference on Machine Learning and Applications (ICMLA)* .

Coucke, A., Saade, A., Ball, A., Bluche, T. and Caulier, A. (2018). Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces.

Croce, D., Castellucci, G. and Basili, R. (2020). Gan-bert: Generative adversarial learning for robust text classification with a bunch of labeled examples. in proceedings of the 58th annual meeting of the association for computational linguistics.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Guo, Shiguang and Wang, Q. (2022). Application of knowledge distillation based on transfer learning of ernie model in intelligent dialogue intention recognition, *Sensors* **22**(3).

Haldar, Rishin, Mukhopadhyay and Debajyoti (2011). Levenshtein distance technique in dictionary lookup methods: An improved approach.

Haode, Z., Yuwei, Z., Li-Ming, Z., Jiaxin, C. and Guangyuan, S. (2021). Effectiveness of pre-training for few-shot intent classification.

Hu, Peng, M. ., Junjie, Zhang, Wenqiang, Hu, Jingxiang, Qi, Lizhe, Zhang and Huanxiang. (2021). An intent recognition model supporting the spoken expression mixed with chinese and english.

Huggins, M., Alghowinem, S., Jeong, S., Colon-Hernandez, P., Breazeal, C. and Park, H. W. (2021). Practical guidelines for intent recognition: Bert with minimal training data evaluated in real-world hri application.

Joulin, E., Grave, Bojanowski, P. and Mikolov, T. (2016). Bag of tricks for efficient text classification, *Corr* .

Kresnakova, M., V., Sarnovský, M., Butka, P. and K., M. (2020). Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification.

Lair, N., Delgrange, C., Mugisha, D., Dussoux, J.-M., Oudeyer, P.-Y. and Dominey, P. F. (2020). User-in-the-loop adaptive intent detection for instructable digital assistant.

Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A. and Tang, L. (2019). An evaluation dataset for intent classification and out-of-scope prediction.

Le-Tien, T., Nguyen-D-P, T. and Huynh-Y, V. (2022). Developing a chatbot system using deep learning based for universities consultancy, *16th International Conference on Ubiquitous Information Management and Communication (IMCOM)* .

Lew, M., Obuchowski, A., Kacprzak, E. and Pluwak, A. (2021). Conversation clustering adaptation for intent recognition,, *20th IEEE International Conference on Machine Learning and Applications (ICMLA)* **22**(3).

Luo, B., Feng, Y., Wang, Z., Huang, S., Yan, R. and Zhao, D. (2018). Marrying up regular expressions with neural networks: A case study for spoken language understanding. in proceedings of the 56th annual meeting of the association for computational linguistics.

Martin (2020). Camembert: a tasty french language model. in proceedings of the 58th annual meeting of the association for computational linguistics, *Association for Computational Linguistics* p. 7203–7219.

Mezzi, Ridha, Yahyaoui, A., Krir, M. W., Boulila, W. and Koubaa, A. (2022). Mental health intent recognition for arabic-speaking patients using the mini international neuropsychiatric interview (mini) and bert model, *Sensors 22* **22**(3).

Mordido, G., Yang, H. and Meinel, C. (2018). Dropout-gan: Learning from a dynamic ensemble of discriminators, *ArXiv* **abs/1807.11346**.

Padala, M., Das, D. and Gujar, S. (2020). Effect of input noise dimension in gans, *ArXiv* .

Ruder, S. (2016). An overview of gradient descent optimization algorithms.

Schmidt, A. and Wiegand, M. (2017). A survey on hate speech detection using natural language processing. in proceedings of the fifth international workshop on natural language processing for social media.

Vasquez Correa, J., Guerrero-Sierra, J., Pemberty-Tamayo, J., Jaramillo, J. and Tejada-Castro, A. (2021). One system to rule them all: a universal intent recognition system for customer service chatbots.

Wang, Y., Yao, Q., Kwok, J. T. and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning., **53**(3).

Xia, C., Zhang, C., Yan, X., Chang, Y. and Yu, P. S. (2018). Zero-shot user intent detection via capsule neural networks, *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* .