2018

Comparing the Effectiveness of Support Vector Machines and Convolutional Neural Networks for Determining User Intent in Conversational Agents

Kieran O Sullivan
Technological University Dublin

Comparing the effectiveness of support vector machines and convolutional neural networks for determining user intent in conversational agents

Kieran O Sullivan

A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Data Analytics)

Thursday 13th September, 2018

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institutes guidelines for ethics in research.

Signed: 

Date: 13/09/2018

Abstract

Over the last fifty years, conversational agent systems have evolved in their ability to understand natural language input. In recent years Natural Language Processing (NLP) and Machine Learning (ML) have allowed computer systems to make great strides in the area of natural language understanding. However, little research has been carried out in these areas within the context of conversational systems.

This paper identifies Convolutional Neural Network (CNN) and Support Vector Machine (SVM) as the two ML algorithms with the best record of performance in existing NLP literature, with CNN indicated as generating the better results of the two. A comprehensive experiment is defined where the results of SVM models utilising several kernels are compared to the results of a selection of CNN models. To contextualise the experiment to conversational agents a dataset based on conversational interactions is used. A state of the art NLP pipeline is also created to work with both algorithms in the context of the agent dataset. By conducting a detailed statistical analysis of the results, this paper proposes to provide an extensive indicator as to which algorithm offers better performance for agent-based systems. Ultimately the experimental results indicate that CNN models do not necessarily generate better results than SVM models. In fact, the SVM model utilising a Radial Basis Function kernel generates statistically better results than all other models considered under these experimental conditions.

Keywords: Conversational Agent, Intent Detection, Natural Language Processing (NLP), Machine Learning (ML), Convolutional Neural Network (CNN), Support Vector Machine (SVM),

Acknowledgements

I would first like to thank my supervisor **Dr Luca Longo**. Without his guidance and backing this thesis would not have been possible.

I would also like to thank my wife **Susan O Sullivan**. It was only through her love and support that I was able to complete this masters course.

Contents

List of Figures

List of Tables

List of Acronyms

SVM	Support Vector Machine	**NB**	Naive Bayes
CNN	Convolutional Neural Network	**DT**	Decision Trees
NLP	Natural Language Processing	**DT**	Bayesian Network
NLU	Natural Language Understanding	**cbow**	Continuous Bag-of-Words
ML	Machine Learning	**RBF**	Radial Basis Function
DM	Dialogue Management	**PCA**	Principal Component Analysis
NLG	Natural Language Generation	**GPE**	Geopolitical Entities
RNN	Recurrent Neural Networks	**FC**	Fully Connected Layer
HMM	Hidden Markov Models	**PL**	Pooling Layer
K-NN	K Nearest Neighbour		
LSA	Latent Semantic Analysis		
RBL	Rule-Based Learning		
BoW	Bag-of-Words		
TF-IDF	Term Frequency-Inverse Document Frequency		
OVO	One versus one		
OVR	One versus rest		
CN	Convolutional Layer		

Chapter 1

Introduction

This chapter will introduce the concepts of conversational agents and the application of Natural Language Processing (NLP) and Machine Learning (ML) in that context of these conversational systems. Next, the overall problem being addressed by this thesis will be detailed. Thereafter, the research objectives will be specified. This will be followed by a definition of the research methodologies and then by the experiment with a detailing of the scope and limitations of the experiment. Finally, an outline of the document, with an overview of the chapters will be provided.

1.1 Background

Conversational agents are computer systems which communicate with users employing natural language and fall into two broad categories; chatbots and task-oriented agents (Jurafsky & Martin, 2009). Chatbots are systems which attempt to converse with users without any tasks to be carried out, often with a goal of passing the Turing Test (Turing, 1950). Task-oriented systems (or agents), on the other hand, aim to provide a natural language interface to facilitate users in data access or invoking actions on computer systems.

The early conversational systems developed in the 1960's and 1970's, like ELIZA and PARRY, can be classified as chatbots (Weizenbaum, 1966; Colby, 1981). These systems utilised basic pattern matching systems to achieve some rudimentary nat-

ural language understanding. With the development of AIML in the 1990's, more sophisticated chatbots could be constructed, with this also leading to advancements in task-oriented agents (Wallace, 2009). In recent years, the development of modern NLP methods and ML algorithms has seen much growth in the ability for computers to process natural language, with abundant research carried out in these areas. However, a lack of conversational agent datasets based on task-oriented systems has lead to a shortage of research in this particular area.

1.2 Research Project/Problem

Conversational interface systems are beginning to make use of NLP with ML to solve complex problems that they face in effectively and efficiently processing natural language input. Nevertheless, few experiments have been carried out on the effectiveness of the NLP processes and machine learning algorithms within this domain. There is an extensive background of work using these processes and algorithms in other domains, in addition to a general area of language processing, but few are focused on the conversational system area and making use of conversational interface data.

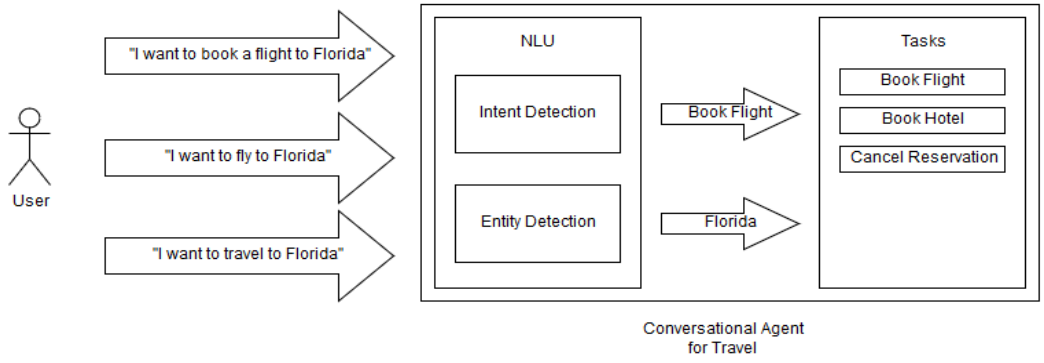

Figure 1.1: Example flow of data and components in a conversational agent system for travel bookings.

Figure 1.1 illustrates some of the components and data flow of a sample agent system. In this example, the user enters the text into the agent's interface which represents their desire to travel to Florida. Due to the nature of language, there are

many ways a user can write the statement. The Natural Language Understanding (NLU) component of an agent must then classify the incoming text to one of the tasks that the system is designed to handle; 'Book Flight' in this case. Traditional agent systems defined specific patterns of words which, when detected, would match to one of the systems tasks. At its core, NLP tries to solve the problem of understanding the meaning of a sentence, or what the users intent was in writing the supplied text (Feldman, 1999). The problem, therefore, becomes how NLP and ML techniques can be applied to the NLU component of a conversational agent, and what techniques and algorithms within those domains offer the best results in the context of conversational agents.

1.3 Research Objectives

1. **Literature Review** To conduct a detailed analysis of existing literature in the conversational agent context. Specifically, their evolution, the different components of agent systems, and how natural language input is processed. In addition, a review of common ML algorithms in the NLP space with particular attention to their use in agent systems. Additionally, an analysis of the state of the art procedures in the context of NLP tasks. Finally to define a research question based on the reviewed literature.

2. **Experiment Design** Firstly, to define a hypothesis H1 based on the research question from the previous chapter. The next objective is to design an experiment to test the defined hypothesis. This experiment will follow the CRISP-DM process for data analytics tasks. The objective of this experimental design is to generate a rigorous experiment which will ultimately provide as comprehensive an answer to the research question as possible, in the time-frame available.

3. **Experiment Implementation** To detail the actions taken and tools used at each step of the designed experiment, and to provide the results of the experiment. The impacts of the data processing step on the dataset are presented

along with the outcome of the grid search optimisation for each model. This is followed by the experimental results for all metrics of every model.

4. **Evaluation** The primary objective is to accept the hypothesis H1 defined in the experimental design. As such, the statistical analysis steps defined in the design are to be carried out. If the analysis results agree with the hypothesis H1 then it will be accepted, and otherwise rejected.

5. **Discussion** Finally, to analyse the results and evaluation in the context of the literature identified during the review. The ultimate goal here is to explain the results and evaluation of the experiment in the background of the literature. This discussion will help to suggest why the results agree or disagree with the existing literature.

1.4 Research Methodologies

The methodologies employed in this paper are classified as empirical, secondary, and qualitative research, utilising inductive reasoning. This paper is considered to take the form of empirical research as a detailed review of existing literature is carried out. From that detailed review, a general theory is formulated in the form of a research question. In this case, the theory is that Convolutional Neural Network (CNN) models achieve better results than Support Vector Machine (SVM) models for intent detection in natural language interfaces. A hypothesis is then defined based on the research question identified. The research hypothesis is clearly defined, testable, and contains well-defined variables, so that it may ultimately be accepted or rejected. The objective of this paper is quantitative research as the hypothesis defines a clear expected outcome with measurable variables. The experiment is subsequently designed to accept or reject the hypothesis generates measurable results, which are thereafter quantitatively evaluated using statistical techniques. This paper is secondary research as the conversational dataset which is used to contextualise the experiment to conversational agent systems was generated and published by El Asri et al. (2017). Finally,

this paper is inductive reasoning as the methods utilised in the experiment take a specific dataset of conversational agent data, and general models are constructed of both SVM and CNN algorithms. Furthermore, a detailed review of the existing general literature provides a specific hypothesis which is ultimately accepted or rejected by the experiment detailed in this paper. This provided a specific answer to the question defined in the literature review chapter.

1.5 Scope and Limitations

The overall scope of this experiment is an evaluation of supervised machine learning techniques for use as an intent detection process in the natural language understanding component of a conversational interface. The intention is to identify the optimal ML algorithm for use in the construction of conversational systems. Due to the nature of NLP pre-processing pipelines, CNN architectures, and SVM kernels, there are an almost limitless number of possible permutations for generating classification models. Exploring every possible variation for each model would prove to be an extremely time-consuming process. Thus the scope of this experiment is limited to the two machine learning algorithms which are most commonly identified as achieving the optimal performance in classification experiments. The SVM models will utilise a number of commonly available kernels with an exploration of their optimal parameters in this setting. Within the scope of this experiment, a number of CNN architecture variations have been defined based on the existing literature. Ideally, other kernels and architectures would be considered if the scope of the experiment could be expanded beyond the time limitations of this study. Additionally, the grid search implementations considered a narrow range of available SVM parameters for each kernel which would ideally be expanded to broader ranges. The CNN grid search only considered three parameters, but a more detailed exploration of the filter sizes and settings may result in an optimised CNN model.

Furthermore, a pre-processing pipeline has been devised which uses the state-of-the-art components which have been identified in the existing literature and which are

compatible with the ML models. The experiment suggests that the Pre-processing steps taken may considerably impact the performance of the models. As such, a broader consideration of the pre-processing steps and their impact may result in better model performance in this experiment.

The experiment is also scoped to utilise the Maluuba Frames dataset (El Asri et al., 2017). This dataset contains information which represents conversational agent data for a travel booking system. Intrinsically , the scope is limited to conversational agent systems and not chatbots. While the data available in the system is highly contextually relevant, it suffers from a very high degree of class imbalance. The use of random oversampling would have resulted in a large volume of repeated samples being added to the dataset (dataset increase from approx. 20,000 to 100,000+). The use of random undersampling would have given rise to a considerable volume decrease (dataset decrease from approx. 20,000 records to 280 records). Synthetic oversampling techniques would have had the consequence of many synthetic records being created, and the impact of synthetic records could not be determined prior to the implementation of the NLP pre-processing pipeline. In addition, ten-fold stratified validation was utilised for the actual experiment. However, due to time limitations, two-fold stratified validation was used for both the SVM and CNN grid search implementations. Ideally, ten-fold would be employed for the grid search as well.

1.6 Document Outline

This paper contains six chapters (including this introductory chapter). A summary of the five other chapters is presented in this section.

Chapter 2: Review of existing literature

This chapter provides a comprehensive literature review, first detailing the evolution of conversational agents from their inception in the 1960's, to the present day. Next, the architecture of several modern conversational agent systems is reviewed, including a discussion on the tools on which these systems are built. Followed closely by a re-

view of classification experiments in the NLP space where ML algorithms are used for classification. Two machine learning algorithms; SVM and CNN are next discussed in detail. This includes a basic description of both algorithms including; linear vs non-linear kernels in SVM, multi-class problems in SVM, the layers of CNN architectures, and definitions of several CNN architectures for NLP. Finally, gaps in the literature are identified and a research question is defined in this chapter.

Chapter 3: Experiment design and methodology

In this chapter, first and foremost, the research hypothesis is defined. Subsequently, an understanding of the data in the dataset is built up. Thereafter, the pre-processing steps to be carried out on the data are defined: lexicon normalisation, noise reduction, feature extraction, feature selection, and splitting the data into training and test sets. Following on, the details of the SVM and CNN models are defined from the frameworks to be used, to the mechanism for grid search in terms of parameter optimisation, to how the evaluation metrics will be measured. Finally, the steps for evaluating the results with statistical tests are detailed.

Chapter 4: Implementation and results

This chapter commences with a comprehensive analysis of the data in the dataset. Following on from the analysis, the details of each of the pre-processing steps, and how these steps impact the data in the dataset is explored. Specifics of the lexicon normalisation, noise reduction, feature extraction, and feature selection steps are catalogued at this juncture. Particulars of the grid search for the SVM and CNN models are presented and discussed, with the results of the execution of each metric and every model being displayed.

Chapter 5: Evaluation and analysis

Opening with an analysis of the normality of the results for every model, each of the three metrics are examined; precision, recall, and F1. Thereafter, a statistical analysis is carried out for each metric, where the results of all the models are compared. This

analysis determines if there is any improvement when comparing the results for each model, with a 95% confidence level. The results of the statistical tests are presented and, subsequently, an analysis is carried out. An overall investigation of the results is carried out at this juncture, with reference to the existing literature. This leads to the hypothesis either being accepted or rejected, with the results being discussed in relation to the overall research question and hypothesis. Finally, the strengths and limitations of the designed experiment are detailed.

Chapter 6: Conclusion

Presented in the final chapter is an overview of the research carried out, with a definition of the problem addressed by this experiment. Subsequently, the experimental design, implementation, evaluation, and results are presented. The contributions and impact of the paper are analysed and, ultimately, recommendations are made regarding future work based on the findings of this experiment.

Chapter 2

Review of existing literature

Reviewing existing literature related to conversational agents in the form of a history of their evolution and a review of what components make up modern agent systems is outlined in this first chapter. Subsequently, an overview of the empirical work carried out in the areas of NLP and Machine Learning are examined. Thereafter,a detailed look at common NLP steps and tools isfollowed by a comprehensive review of the two most popular ML algorithm identified in the empirical work; CNN and SVM. In conclusion, gaps in the literature are identified, and a research question is formed.

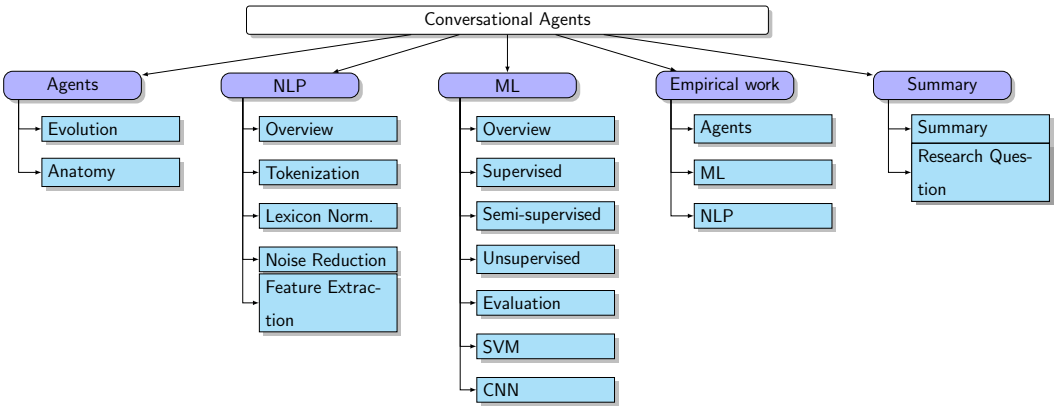

Figure 2.1: Areas of the literature review

2.1 Conversational Agents

A conversational agent refers to a system that allows natural dialogue interactions with humans through one or more language mediums, for example; text, or speech (Niculescu et al., 2014). Conversational agent systems first began to appear in the 1960's with the development of systems like ELIZA (Weizenbaum, 1966). ELIZA received natural language input from a user and scanned the input for keywords. When a keyword was identified, the containing sentence was transformed utilising the rule associated with that keyword. In the 1970s, the PARRY system was an agent which was created to model the reactions of an individual suffering from paranoid schizophrenia (Colby, 1981). PARRY built on the rule-based system from ELIZA by including a model of mental state which was impacted by the input from the user and, in turn, modified the responses of the agent.

In the 1990s the pattern matching employed by agent systems began to evolve with the development of the A.L.I.C.E system, which was initially designed as a replacement for ELIZA (Wallace, 2009). An XML based syntax called AIML was developed, and formed the basis for the system. AIML is a framework for constructing bots utilising pattern matching techniques in order to determine the users intent and respond accordingly. The AIML script defines sentences or sentence patterns which are matched against the natural language input from the user. The script also defines a template response for each input pattern. This development provided a foundation for future agents to be developed on top of AIML with the development of custom scripts.

One of the primary characteristics of each of the three conversational agents described; ELIZA, PARRY, and A.L.I.C.E., is that each performs a type of pattern match, and responds with a pre-scripted or patterned reply. The PARRY system does, however, maintain a limited state which is impacted by, and impacts on, the input and output. These types of systems are often referred to as chatterbots. Chatterbots can hold somewhat limited conversations with users but with little real understanding, and are typically limited to relaying the information that they have been scripted to provide.

? (?) survey the current state of conversational agents, highlighting that they are becoming more common in recent years. The rise in popularity of instant messaging platforms is leading to a resurgence of text-based agents. Furthermore, voice-based virtual private assistants are also becoming popular. One of the reasons given for these rises in popularity is that modern agents facilitate a natural language interface for tools that can be employed in everyday life. In order to differentiate these types of agents from the chatterbots which were prevalent in previous decades, the term botplications has been coined.

Bieliauskas and Schreiber (2017) delineate a knowledge engineering conversational agent which uses regular expression pattern matching in order to determine the users intent. Once the users intent has been identified, the system generates specific visualisations from its dataset based on the that intent. This represents a system which processes the user input, carries out actions on those inputs through the employment of external tools and processes, and subsequently responds to the user with appropriate results.

Graf, Krüger, Müller, Ruhland, and Zech (2015) describe a conversational agent system called Nombot which was designed to utilise an instant messaging platform for user input. The system allows users to track their food intake and weight through the message platform. AIML is utilised to parse the user input by implementing AIML input phrases with placeholders. When the appropriate phrase is detected, the placeholder information is extracted and the appropriate data is sent to or retrieved from a database. Because of the employment of AIML, the users of the system are either required to enter particular patterns for actions to be carried out, or many variations of natural language strings are required to be mapped in AIML.

Sarikaya et al. (2016) outline a system which utilises several machine learning SVM models in place of the traditional pattern matching used in previous examples. These models are employed to parse the input into commands which can be executed by an agent in order to carry out tasks on external systems, for instance playing music, web searches, setting alarms, and such likes. This machine learning approach has some advantages over the traditional pattern matching approaches in that it is often able

to determine the users intent from the input text without having to specify every variation of the text for pattern matching.

As evidenced by the papers above, the area of conversational agents has evolved since its first inception in the 1960's. Early agents utilised simple pattern matching, which gave way to more complex pattern matching techniques like AIML, for the purposes of translating user text input into a format which can be actioned by a computer system. However, due to advancements in NLP and ML, more recent versions of agents utilise these techniques as it is believed that these techniques allow for improved text analysis, with a more generalised ability to process variations of the same input.

2.1.1 Anatomy of a Conversational Agent

Wang and Yuan (2016) audited modern practices in the field and state that there are three primary components for conversational agents: NLU, dialogue management (DM), and natural language generation (NLG). A graphical representation of the components, their relationship, and how data flows through the system can be seen in Figure 2.2.

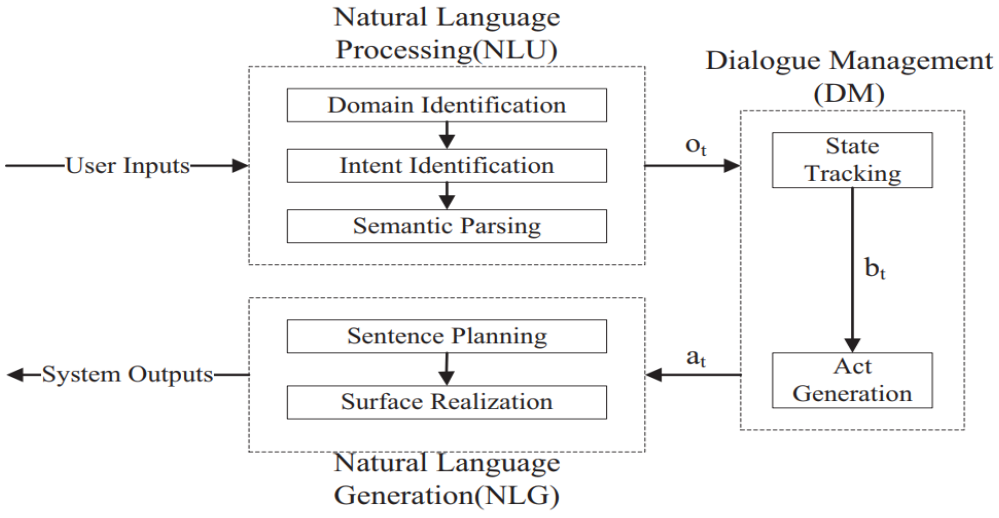

Figure 2.2: Primary components and sub-components of a sample conversational agent system (Wang & Yuan, 2016)

The NLU unit is made up of three sub-components; Domain identification, Intent

identification, and Semantic Parsing. Domain identification is the NLP task of identifying the particular problem domain to which the text relates. Intent identification is the NLP task of determining the intent of the statement or statements input to the agent. Semantic parsing is the construction of a machine-understandable command from the natural language input.

The DM unit contains two sub-components; State tracking and Act generation. The state tracking component receives the machine commands generated by the Semantic parsing component and modifies its internal state accordingly. The Act generation component reacts to the state changes made and determines the appropriate response.

The NLG unit contains two sub-components; Sentence Planning and Surface Realisation. The Sentence planning component is responsible for generating syntax as a response to the Act Generation step. Surface realisation is the process of creating a linear sentence which follows correct syntactical guidelines.

The system described by Sarikaya et al. (2016) follows a similar structure to the one defined by Wang and Yuan (2016). The system is broken down into three primary components; Input, Update state, Processing. The Input component translates the natural text into machine understandable factors. The Update state maintains the current state of the agent and determines the action to take based on the state, and generates the response. The Processing component applies the appropriate action to the external systems and procedures based on the output of the Update state component.

Rahman, Sohan, Zinnah, and Hoque (2017) characterises an overall structure for a conversational interface which is similar to those defined before. The NLU component, which they term the Natural Language Interface Engine, can be seen in 2.3. This paper delineates in more detail the individual steps that the engine takes in processing natural language input. The Lexical Analysis and Syntax step splits the input text into tokens (words) and subsequently utilises lexical analysis to stem the tokens. The Parts of Speech Tagging step associates each token with the appropriate parts of speech tags. The Named Entity Recognition step identified the variables that the agent will use

when processing commands. Sentence Dependency Parsing is exercised to generate a tree structure which identifies the relationships between different parts of the input text. Lastly, the Keyword Based Intent Analysis step utilises a keyword approach to identifying intent, much like the ELIZA and PARRY programs. The system makes use of several modern NLP concepts in terms of lexical analysis, part of speech tagging, and names entity recognition but utilises a keyword based intent identification step as was employed by the basic agents developed in the 1960's and 1970's.

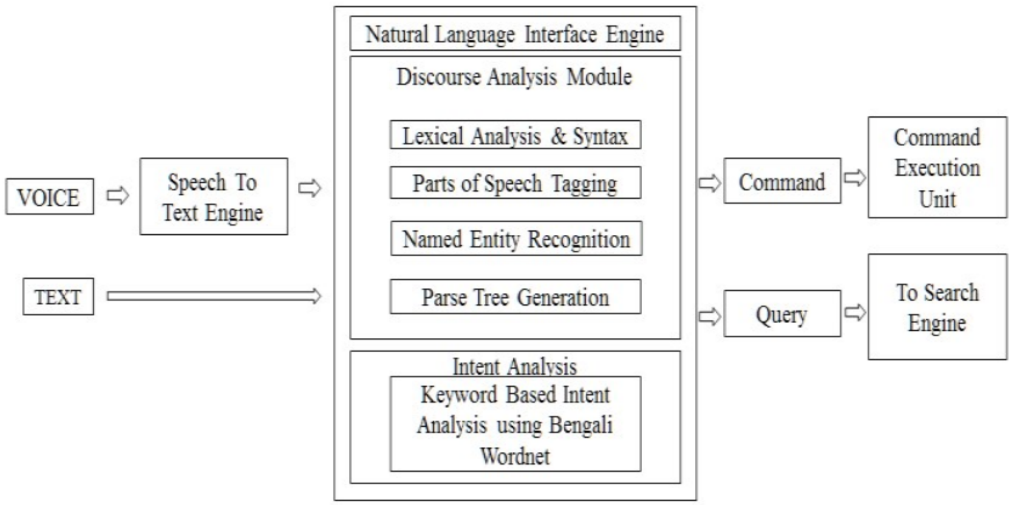

Figure 2.3: Primary components and sub-components of an alternate conversational agent system (Rahman et al., 2017)

The conversational agent system described by Draskovic, Gencel, Zitnik, Bajec, and Nikolic (2016) defines structures which are analogous to the components delineated previously, e.g. (Wang & Yuan, 2016). A number of NLP sub-components make up an equivalent to the NLU component. This is followed by a state management and command parsing component.

Each identified agent system architecture contains a component which translates natural language into a meaning and format, which can be processed by the agent, generally termed Natural Language Understanding. Modern agent systems are built around an NLU component which is based on the principles of natural language processing and machine learning, as opposed to the pattern based systems of the past.

2.2 Overview of Natural Language Processing

Natural language can be defined as the language that is utilised by humans to communicate on a daily basis. The many human languages in use have evolved over time and as such, it is difficult to define a set of rules for any language, which could allow for easier understanding by logical systems. At its core NLP can be described as the understanding of natural language inputs for the purpose of providing useful and meaningful responses to the input Bird, Klein, and Loper (2009). Improvements in the ability of conversational agents in the area of natural language understanding can be considered to be a critical area in advancing the utilisation of conversational agents (Ortiz, 2014).

2.2.1 Tokenization

Tokenization is the process of deconstructing text into smaller chunks called tokens (Webster & Kit, 1992). These tokens typically represent the smallest unit that a textual representation can be broken down to, while still retaining some meaning. At its simplest, this is achieved by regarding each word as a token. However, the process may also take punctuation into account when creating tokens and, in addition, may remove punctuation from the output. It is the resulting tokens which are processed and parsed to form the final input to the model in a text categorisation task.

2.2.2 Lexicon Normalisation

Lexicon normalisation is the process of reducing different variations of words to a single root. For example, better, best, and good, may all be distilled down to good. By replacing all instances of the example words with their root, classification algorithms have to deal with less variation in building classification models. There are two differing processes primarily employed in NLP for this purpose: Stemming, and Lemmatisation. Stemming is the process of mapping related words to a root form (Gharatkar, Ingle, Naik, & Save, 2017). A Stemming algorithm typically has a pre-

identified list of prefix and suffix character sets. Each word in the target corpus is processed to remove matching leading and trailing characters from the algorithms dataset. For example, with the suffixes -es, and -ed, the words studies and studied both become studi. The Lemmatisation process, on the other hand, utilises morphological analysis of words based on dictionaries embedded within the algorithms, to reduce words to their root meaning, also known as a lemma (Zeroual & Lakhouaja, 2017). For example, the lemma for better is good, resulting in the replacement of the word better with the word good.

Korenius, Laurikkala, Järvelin, and Juhola (2004) compared the effectiveness of both stemming and lemmatisation for text processing as input to a clustering algorithm for use with the Finnish language, in a classification task. The experiment conducted a statistical analysis of the results which concluded that there was a statistical improvement with 95% confidence when the models using a lemmatised input were compared to those using a stemmed input.

Zeroual and Lakhouaja (2017) compared stemming and lemmatisation for information retrieval purposes on several Arabic datasets. The data gathered illustrates that lemmatisation is a more effective process for identifying the occurrence of words with similar meanings. This indicates that the lemmatisation process is a valuable step in reducing noise for the purposes of NLP modelling. However, no machine learning models were created or evaluated in the context of the lexicon normalisation algorithms. Additionally, no statistical analysis of the experimental results was carried out.

Both of the identified papers suggest that the lemmatisation process generates better results for NLP tasks than stemming. However, neither is carried out on an English language dataset, and only one related to a text classification task.

2.2.3 Noise Reduction

Noise reduction is the process of reducing the occurrence of words and characters which provide little information in the context of the data under evaluation. This is related to lexicon normalisation in that both steps aim to reduce the input to just the most relevant information needed for carrying out the task under consideration thus it can

help to improve the accuracy of ML models (Furlan, Batanovic, & Nikolic, 2013).

Stop words are the most commonly employed words in a language or corpus. In English these are words like 'the', 'be', 'to', 'of', and 'and'. These commonly occurring words can occur so frequently that they provide little meaning to individual records and as such can increase confusion in the context of text categorisation tasks. Furthermore, the identification and removal or replacement of entities within the input text may also reduce noise. In effect, this is similar to the lexicon normalisation step as the identified entities are reduced to a core representation. Some entities which can be considered are; dates, places, names, and currency values. For example, a single city name may occur a single time in a travel booking dataset, i.e. "I want to travel to Vienna". This may be identical to other strings in the dataset with the exception of the destination city. By replacing all identified city names with a common representation (e.g. "I want to travel to [City]"), the overall meaning of the sentence is maintained while allowing for potentially easier pattern recognition during model training.

2.2.4 Feature Extraction

Bag-of-Words (BoW) is vectorisation process where the entire corpus of a dataset is first converted into a vector where each unique word is represented by a dimension in the vector. For each record to be vectorised, the words in the record are parsed to their corresponding dimensions in the BoW representation, and the value of that dimension is either set to 1 (Wallach, 2006). This mechanism which can be referred to as one-hot-encoding, informs the models which utilise the output from the bag-of-words transformation that particular words exist in the record, but not how many times those words occur. A variant of this approach is called Term Frequency (TF). Instead of binary encoding, this approach increments the values for that word, instead of setting the word to 1 (Zhao & Mao, 2018). Both processes output a fixed length vector based on the number of words in the corpus, e.g. a dataset with 1000 unique words would be represented by a vector with 1000 dimensions. TF-IDF is a further extension of the TF process. In this model, the corpus is converted to a fixed size vector with a dimension per unique word, as in the bag-of-words model. However, the

values for each dimension for a given record are based on a statistical computation of the frequency of the word in the record, offset by the frequency of the word in the overall corpus. This results in words which appear regularly in the overall corpus being given less weight than less frequently occurring words (Mishra & Vishwakarma, 2015).

Word embedding models are a form of vectorisation technique where vector representations are learned in a non-supervised manner, by machine learning models. Word2Vec for example takes a text corpus as input and generates a vector space where each unique word from the input corpus is represented by a dimension (like TF-IDF and BoW). The words are represented in the vector so that commonly associated words are located in close proximity in the vector representation. However, each word dimension is represented by a vector representation of embedding values. Within the context of Word2Vec there are two model architectures used to generate the output; continuous bag-of-words (cbow) and continuous skip-gram (Mikolov, Chen, Corrado, & Dean, 2013). GloVe is another popular word embedding technique which aims to combine the principles of both the cbow and continuous skip-gram models from Word2Vec in order to provide an algorithm which is superior to both (Sharma, Agrawal, Jain, & Kumar, 2017).

Lilleberg, Zhu, and Zhang (2015) conducted an experiment comparing Word2Vec and TF-IDF for the purposes of text classification using an SVM classification model. The results indicate that TF-IDF model achieved a better average accuracy than the Word2Vec model. The experiment does suggest that models, where both TF-IDF and Word2Vec were combined, achieved a higher classification accuracy. However, no statistical tests were carried out as to the significance of the differences in output for any of the results.

On the other hand, the experiment carried out by Y. Zhang and Wallace (2017) compared Word2Vec, GloVe, combined Word2Vec with GloVe, and one hot encoding (BoW) as the feature extraction process for CNN models built on several datasets. The experiment finds that the performance of Word2Vec versus GloVe is dependent on the dataset. The results of the one hot encoding and combined models are reported

as not being as good as either Word2Vec or GloVe. However only the GloVe results are reported and no statistical analysis is carried out.

2.3 Overview of Machine Learning

Mitchell (1997) describe machine learning as a computer program which can learn from performing tasks where the performance of the program improves with experience. That equates to the computer program iteratively improving its ability to carry out an assigned task over time. Machine learning is often utilised for two types of problems, prediction and classification. In prediction, a model is generated which attempts to generate an estimate for a continuous variable based on the data supplied to the model. In categorisation, the model attempts to assign the data to one or more label. Machine learning is typically divided into three areas; supervised learning, unsupervised learning, and semi-supervised learning.

Conversational agent Intent Detection is analogous with Text Categorisation within the broader NLP and ML context (Purohit, Dong, Shalin, Thirunarayan, & Sheth, 2015). A user enters a sentence or sentences, the NLU component of a conversational agent then applies a method in order to discern the intent of the user in relation to the applicable functions that the agent can carry out. The text input for conversational interfaces is typified by short lines of text, often one sentence in length. The input is, therefore, a short section of natural language and the output is a defined action that the conversational agent can action or respond to (Intent).

Machine learning can be categorised as a key research area in the context of NLU for conversational agents (Zue & Glass, 2000). Sebastiani (2002) details the approaches and benefits of using machine learning algorithms for text classification when compared to knowledge engineering approaches. They suggest that the introduction of machine learning techniques like SVM has lead to substantial improvements in the ability to automatically categorise text.

2.3.1 Supervised Learning

Supervised learning is characterised by the use of a labelled training set which contains both variables and expected output. ML algorithms utilise this information to alter their internal weights and parameters in order to minimise the error value generated by the model (Bonaccorso, 2017). This type of learning algorithm typically requires a large amount of labelled data in order to generate effective models, which can be an issue in some problem spaces (Al-Dmour & Al-Ani, 2016). Examples of supervised learning algorithms include: SVM, CNN, Recurrent Neural Network (RNN), K-Nearest Neighbour (K-NN), Decision Tree (DT), Naive Bayes (NB), and Hidden Markov Models (HMM).

Both CNN and RNN are examples of neural networks, but many other neural network algorithms have been defined. Typical modern neural networks are characterised as having an input layer, an output layer, and one of more hidden layers in between the other two. Each node consists of a number of nodes. In the input layer there is a node for each of the dimensions in the supplied vector. In the output layer, this takes the form of a node per expected result, e.g. if it is a three class classification problem the output layer may contain three nodes, one representing each output class. The hidden layer contains any number of nodes. Each node in every layer is connected to all of the nodes in the previous layer. Each connection typically has a weight and each hidden layer node contains a mathematical transform function for example, sigmoid. The input data feeds through each layer and is transformed utilising the weights connecting the nodes. Through the use of back propagation the weights of the neural network are altered in an attempt to reduce the error value generated for each input record (Nazzal, M. El-Emary, & A. Najim, 2008). In a CNN model the system described above is analogous to the fully connected layer. However, CNN models can feature many other layers in an effort to transform the data for classification or prediction, for example, convolutional layers for input transformation and reduction, dropout layers to prevent over-fitting and output layers for final translation to the expected result. RNN models expand on these principles by adding a temporal element to the clas-

sification mechanism. The previous state of the neural network becomes an input variable for the model so that the previous context of the network helps to inform the prediction or classification for the current record. This can help when processing sequential data like radio signals.

SVM model develop a hyperplane or multiple hyperplanes in high dimensional space, such that the optimal hyperplanes have the greatest distance to the points in the dimensional space of the model. This means that every input vector is mapped to the high-dimensional space and each point is classified as one or the other class label. The optimal separator therefore is the one which provides the most distance between the points representing the two classes. The training data is first introduced to the model and the hyperplanes are formed. When the data to be classified is supplied to the model and rendered on the feature space, the output of the SVM prediction or classification is based on the points location relative to the optimal separator (Burges, 1998).

Decision Trees are a group of algorithms based on the principal that a complex decision can be be formed through the union of multiple simple decisions. There are many different DT implementations but the core concept is recursive data partitioning, where the data is split into two or more branches based on the optimal split for that node using the specific metric for the algorithm (Ignatov & Ignatov, 2017). Metrics include Gini impurity, information gain, and variance reduction.

For the K-NN algorithm, input vectors are represented in a multi-dimensional feature space. During training all of the input vectors are mapped to the feature space, then during classification the individual input vector is mapped to the space and the most common label among the k nearest neighbours is assigned to the vector under classification (Ryoo, Arunachalam, Khanna, & Kandemir, 2018).

2.3.2 Unsupervised Learning

In contrast to supervised methods, in unsupervised learning unlabelled data is provided to the algorithm which then attempts to group each record into clusters based on the similarities of the features in the record (Al-Dmour & Al-Ani, 2016). Examples of

supervised learning algorithms include k-means clustering and Apriori association rule mining.

In k-means clustering the aim of the algorithm is to separate every input vector into k different groups or clusters. This has the result of identifying similar records within the input dataset as they are grouped together based on the identified similarities in the input dimensions. When developing a k-means clustering model the number of clusters is defined and then the model is trained with un-labelled data. The output is the identification of k different groups which demonstrate data similarities (Hartigan & Wong, 1979).

The Apriori association rule mining algorithm is designed to identify transactions with similar data subsets in the input dataset. A threshold value C is set and all records which are subsets of at least C transactions in the input data (Hipp, Güntzer, & Nakhaeizadeh, 2000).

2.3.3 Semi-Supervised Learning

Semi-supervised learning methods draw on the principles of both supervised and unsupervised learning in order to form a hybrid approach. It utilises a small amount of labelled data, combined with a large volume of unlabelled data in order to classify the records. Labelled data is costly to generate and semi-supervised learning attempts to compensate for that cost by employing a relatively small amount of labelled data (Cheung & Li, 2017). Examples of semi-supervised learning algorithms include are graph based models and transductive SVMs.

Graph based models apply transductive reasoning to graph theory in order to achieve semi-supervised learning(Vapnik, 1998). In graph theory the data is mapped to euclidean space and connections are formed between entities. Relationships are calculated from the data and the distance between nodes (or number of connections) is used as a measure of similarity between any two nodes. (Culp & Michailidis, 2008).

Transductive SVMs (TSVM) are an extension of supervised SVMs where the principles of transduction have been applied (Vapnik, 1998). When training a TSVM both labelled and unlabelled data is input to the model. The model generates the optimum

separating hyperplane function for both the labelled and unlabelled data (Joachims, 1999).

2.3.4 Evaluating Performance

Evaluation metrics are used to provide a measure of the effectiveness of ML algorithms. By measuring the the outcome of the models it allows different models to be compared in order to determine which is more effectiveness. Several common metrics are utilised; precision, recall, accuracy, F1 score. Table 2.1 details these evaluation metrics.

Utilising the complete dataset to train a model and then applying metrics to evaluate that model is not recommended. When the model performance is measured on all the available data it is not a good assessment of that models ability to generalise in order to classify or predict on data that it has not seen (Witten, Frank, Hall, & Pal, 2016). Therefore, data is typically split into training and test sets, where the training set is used to build the model and the test set is used to evaluate the model. This process is known as the holdout method. Typically the training set will be made up of the larger portion of the data. Typical splits are to utilise seventy to eighty percent of the data for training. An alternative approach is to utilise stratified k-fold cross validation. In this method, the data is split k times into stratified samples so that each sample is a proportional representation of the overall dataset. This results in k models being trained and evaluated. For each model creation k-1 folds are used to train the model and the remaining model is used for evaluation. This is carried out in sequence so that each fold is used a single time for evaluation. Ten-fold stratified cross validation has generally been found to produce the most accurate metrics for most data analysis tasks (Witten et al., 2016).

Table 2.1: Summary of common evaluation metrics

Metric	Formula	Notes
Precision	$\frac{tp}{tp+fp}$	The proportion of the positive predictions which were incorrect (Sarwar, Karypis, Konstan, & Riedl, 2000)
Recall	$\frac{tp}{tp+fn}$	The proportion of actual positives which were correctly identified (Sarwar et al., 2000)
Accuracy	$\frac{tp+tn}{tp+tn+fp+fn}$	The proportion of overall values which were correctly predicted (Sarwar, Karypis, Konstan, & Riedl, 2001)
F1 Score	$2 \cdot \frac{precision \cdot recall}{precision+recall}$	Harmonic mean of precision and recall (Yang & Liu, 1999)

2.4 Empirical work on Machine Learning in the Natural Language Processing space with a focus on Conversational Agents

Draskovic et al. (2016) defines a conversational agent for utilisation with a social media platform. The Maximum Entropy classifier is used for Intent classification, stating that "This paper will observe maximum entropy classifier (MaxEnt) as the most convenient one for textual message classification" (Draskovic et al., 2016, p. 2). However, no evidence is provided in the paper to support that assertion. The paper denotes a pipeline for NLU which includes part-of-speech tagging, ngram conversion, with the identification, tagging, and transformation of date inputs. The paper suggests that utilising a neural network model instead of a Maximum Entropy classifier would be more accurate, but no supporting information is provided. Additionally,, no experiment is carried out in order to determine the effectiveness of any element of the system.

Sarikaya et al. (2016) discuss some of the operational details of Microsoft's Cortana system. Specifically, the paper focuses on the NLU element of the system. The Cortana system utilises several different SVM models in order to carry out the NLU tasks; Domain detection, Intent detection, and Slot filling. The SVM models were evaluated against conversational data collected by Cortana, encompassing over 400,000 records, spread across ten different domains. A measure of the efficiency of the models over all of the datasets is presented for each of the three NLU tasks. For Domain and Intent detection the average accuracy over all datasets is presented as 86% and 83% respectively. For Slot filling the F1 score is presented as 85%. The paper posits that SVM models are an excellent choice for NLU, but makes no mention of other options for carrying out NLU tasks. Furthermore, there is little-detailed information regarding the datasets under consideration, or the SVM models used to perform the evaluation.

Lee and Dernoncourt (2016) suggest that RNN and CNN models offer better classification performance than other ML models for short text classification. Three different datasets are used to appraise the text and models were trained and evaluated for RNN, CNN, and Majority Class. A general NLP dataset consisting of 362,000 records is used for this experiment. The accuracy statistics for these models were recorded and compared to other experiments carried out on the same datasets in other papers. The CNN model utilises a single layer with Relu activation and max pooling, as such it can be described as a shallow CNN. Only one dataset compares the results to an SVM model, in this, the RNN is presented as having an accuracy score of 66.2% compared to the SVM models 57%. Another dataset is compared to both Graphical Model and NB models. In this, the CNN model provides better accuracy. The final dataset compares the models to HMM, memory-based learning, and Interlabeler agreement. CNN provides better accuracy than the comparable models on this final dataset. The experiment does compare several models in a short text scenario. However, the three datasets are based on human dialogue from meetings and other transcripts and, as such, are not an ideal representation if evaluating for conversational interfaces. The results are compared to a separate experiment where an SVM model is created, but in the context of NLP, the pre-processing steps could have a significant impact on the

resulting models. Without carrying out the same pre-processing steps for both the CNN and SVM models, an accurate comparison could be challenging to achieve. In addition, the experiment does not perform a statistical analysis of the results, a simple comparison of the measured values is executed.

Hijjawi, Bandar, and Crockett (2013) performs an experiment where several machine learning classifiers are compared for effectiveness against a general Arabic text dataset, consisting of 1,318 records. The purpose of the experiment was to identify the best classifier for employment as the intent identifier in a conversational agent called ArabChat. The classifiers evaluated were; NB, Bayesian Network (BN), OneR, ZeroR, and J48. A bag-of-words approach was used to convert the Arabic strings to vectors. The J48 DT algorithm was found to have the best overall performance when run against the Arabic corpus using 10-fold cross-validation and a paired t-test were used to determine the significance of the results. The J48 classifier was found to achieve the best results of the classifiers evaluated. The Arabic dataset was created for the experiment by selecting several different sources of text, these typically took the form of transcriptions of interviews, books, and news articles. These data sources are general NLP in origin and would not represent the optimal for an experiment in a conversational agent context.

Xu and Sarikaya (2013) analyse the use of CNN models for joint Intent detection and Slot filling. In this case, the models evaluated output both a categorised intent and identified slots of variables. A general NLP dataset consisting of 6,362 records was employed for this experiment. The designed CNN for Intent detection has a single convolutional layer with an output layer where a single unit is representing each categorised output. They report that a max pooling layer provided no significant difference in accuracy. A rectifier activation function is utilised, and a dropout rate of 50% is included. The dataset transcribed from voice conversations for an airline travel information system was employed. The error rate of the CNN for Intent detection is compared to other published results on the same dataset, namely an SVM model for intent detection. Both an independent CNN and the joint CNN created for the experiment maintained a lower error rate than the SVM model. The dataset under

consideration is not the most appropriate representation of the interaction for a conversational agent. As with the previous experiment, the generated model is compared to the results of an existing experiment, which may make an accurate comparison of the models challenging to achieve. This experiment does not conduct a statistical analysis. Instead, a comparison of the values between the models is presented.

Nii, Tsuchida, Kato, Uchinuno, and Sakashita (2017) implements an experiment where Word2Vec and CNN are combined to classify a nursing care dataset consisting of 8,313 medical notes collected in Japan. The CNN model utilised a single convolutional layer of 100 features in a range of sizes from two to five units. A tanh activation function was utilised. The convolution layer was followed by a max pooling layer and, subsequently, a dropout layer with a rate of 50%. Three final classification layers were evaluated, softmax, SVM, and K-NN. They identified K-NN as the optimal classification layer given the results generated. The results were compared to those generated by previous experiments with the same data but using SVM models for classification. It was also reported that more records were correctly classified by the CNN model than the SVM model. However, the previous experiment made use of bag-of-words for vectorisation and not Word2Vec. Furthermore, no statistical analysis of the results CNN results compared to the SVM results was carried out.

Firmino Alves, Baptista, Firmino, Oliveira, and Paiva (2014) perform an experiment where a collection of 2,270 Portuguese tweets are processed through a text classification process in order to determine positive or negative sentiment. Two machine learning techniques are employed; SVM and Naive Bayes. The bag-of-words algorithm is utilised to transform the text into vectors for processing. 10-fold cross-validation is used to evaluate the techniques, and accuracy, precision, recall, and F1 are the metrics employed. The SVM model is identified as generating the overall best results in the experiment. However, the metrics are merely compared without any statistical analysis step. Additionally, the dataset was constructed for the experiment by gathering tweets in the Portuguese language with specific hashtags. Thereafter, a combination of a process based on emoji characters and manual labelling was used to build the sentiment portion of the dataset for training an analysis. This type of

data processing may lead to more errors where data is mislabelled before training and evaluation.

Chen, Zhang, and Mark (2012) evaluated several classifiers for utilisation as intent detection in a dataset of 1,539 questions and statements taken from Yahoo answers and manually labelled. The classifiers evaluated were SVM with a linear kernel, C45, Random Forrest, Naive Bayes, and K-NN. The experiment was conducted using 5-fold cross-validation. The SVM model with linear kernel was reported as generating the best results. However, no mention is made to data processing steps taken e.g. vectorisation technique. In addition, no statistical tests were reported for the results comparing the models. Instead it is stated that the linear SVM model generated the best results and only the results for that model are presented.

Bhargava, Celikyilmaz, Hakkani-Tr, and Sarikaya (2013) designed an experiment where a labelled dataset of 27,565 spoken sentences in the conversational agent field was translated into a textual representation. Three machine learning models were trained to perform both intent detection and slot filling; a linear SVM, a combined SVM and HMM, and Conditional Random Fields. Ten-fold cross-validation was used to generate and evaluate the models. The paper concludes that SVM models are generally a better choice for Intent Detection but noted no statistical significance between the SVM and the combined SVM with HMM models. In this experiment, only the accuracy is utilised as an evaluator. Furthermore, there is no information provided regarding data pre-processing steps carried out, beyond the transcription of vocal commands to text. One interesting note is a portion of the experiment where the previous intent was included as an input into the model, resulting in an increase in model accuracy.

Ding, Liu, Duan, and Nie (2015) design a system for deriving intent from posts on a micro-blogging platform, using a CNN model. A dataset of 1,000 records was generated for this experiment. They first utilise a C&W model (Collobert et al., 2011) to generate word embeddings as the input to the CNN. A one-dimensional convolutional layer is employed, followed by a max pooling layer. A fully connected layer with a sigmoid activation function is used. Finally, an output layer utilising softmax activation is

employed as an output for the two variables. For comparison, two SVM models were also evaluated. Both used a linear kernel, but one utilised word embedding, and the other utilised bag-of-words encoding. The accuracy metric was employed to evaluate the models. It is reported that the CNN model achieved a higher accuracy than the SVM models. A single training/test/evaluation run was executed for the experiment, instead of multiple which would result in a more accurate representation of the results. No statistical tests were carried out, the accuracy results were simply compared.

Gaikwad and Joshi (2016) carried out an experiment to evaluate several models for the categorisation of text data from a micro-blogging system. A dataset of 8,000 records was generated for this experiment. The experiment compared SVM, Naive Bayes, and K-NN models. No further details are provided as to the exact settings used for each model. The data was processed to remove stop words, replace emoticons with words, and remove special characters. Term Frequency-Inverse Document Frequency (TF-IDF) was utilised for feature extraction, and this data was fed into the three models. The data was split into a 90% training split and a 10% evaluation split, but repeated evaluation was not employed. The accuracy, precision, recall, and F1 were used to evaluate the models. For almost all metrics SVM outperformed the other models. No statistical analysis was carried out on the results.

Hassan and Mahmood (2017) suggest a model where the output from the convolutional layer is passed to a recurrent layer long short-term memory, instead of a more traditional pooling layer. A general NLP dataset consisting of 259,855 records was utilised in this evaluation. Pre-trained word embeddings are used as the input to the convolutional layer, using Word2Vec. A 50% dropout rate was included in order to prevent over-fitting of the model but an early stopping strategy is also put into place. A backpropagation through time strategy was employed as the loss function, this is with recurrent networks. The suggestion is that by combining both CNN and Recurrent model elements, the model will achieve a better classification rate for sentiment analysis. The accuracy and error rates are compared to the rates reported from other experiments. The presented results report an improvement in accuracy over several models including SVM. The accuracy results are very similar to more traditional CNN

models to which it is compared. In fact, one of the CNN models reports a slightly higher accuracy. The reported error rate is compared to fewer models than the accuracy. The reported results are largely similar to the comparable models. A single test/training split was used for the evaluation instead of multiple evaluations as in k-fold cross-validation. The results of the new proposed model are compared to models from other papers which raises potential concerns due to the differences in preparing and evaluating the data and models between experiments. In addition, no statistical tests are carried out as to the significance of any improvements.

Dumais, Platt, Heckerman, and Sahami (1998) conducted an evaluation of several machine learning classifiers for the purposes of text classification on an general NLP English dataset of 9,982. The classifiers evaluated were; Find Similar, NB, BN, DT, and an SVM (linear). The accuracy metric was utilised to evaluate the models over a 70/30 split of the data. The SVM model reported the highest accuracy in the experiment. However, no statistical analysis was carried out, and the evaluation was carried out on a single data point for each models metric.

Imane and Mohamed (2017) conduct an experiment in order to evaluate several ML classifiers in the context of multi-label text categorisation. A dataset of 65,843 French death certificates is employed as the basis of the evaluation. First a number of data preparation tasks are carried out: tokenization, convert to lower case, remove stop-words and punctuation, and stemming. The data is subsequently split into 80% training data with 20% test data. Following this, the data is vectorized using TF-IDF, but one-hot-encoding and term frequency were considered. Three ML algorithms were evaluated: DT, SVM (linear kernel), and AdaBoost using DT. However, no further details as to the settings of the models are presented. The precision, recall, and F1 score for the three models are presented. The F1 score is the primary metric used to evaluate the models, with DT model generating the best results, with the SVM model providing the second best results which are very close to the DT results. The experiment performed a simple comparison of the single F1 score generated for each model, so no statistical analysis was carried out, and only a single score was utilised for the evaluation of the models.

Li and Shen (2017) investigates the introduction of Latent Semantic Analysis (LSA) on classification for several models; Naive Bayes, linear SVM, logistic regression, and a non-linear SVM (where the kernel is not identified). LSA is a method of dimensionality reduction which analysis words and their distributional semantics in order to generate a set of concepts, this is somewhat similar to a Principle Component Analysis (PCA) process. TF-DF is used as a vectorisation process. Each model is trained and evaluated on a dataset of 6,000 records from a micro-blogging platform, hence individual records can be considered short text. The precision, recall, and F1 are reported for each model. The experiment is subsequently run again but including an LSA step. The same metrics are employed and the numbers from both experiments are compared. The results indicate at least some improvement for each metric of every model when LSA is utilised. In this experiment the linear SVM model generated the optimal results for each metric. However, no statistical analysis is carried out, and the metrics are generated from a single training/test execution on the dataset.

R. Zhang et al. (2017) conduct an experiment to evaluate Two SVM models with a Rule-Based Learning (RBL) system. This experiment was conducted using 6,174 medical records in order to assign the correct classification to each one. The data was split into 60% for training, and 40% for evaluation and a single iteration of the experiment was conducted. Two SVM models were used, one with bag-of-words as input, and another using bag-of-words with ngrams included. English stop words were removed, and Lexical Variant Generation was performed. There are no details provided for the kernel or settings of the SVM models and no similar information provided for the Rule-Based Learning models. The results indicate little difference between SVM (using n-grams) and RBL with RBL resulting in higher precision, but SVM resulting in higher recall and F1. The SVM with bag-of-words did not generate comparable results to the other models. However, no statistical analysis was carried out on any of the results.

A summary of the identified literature can be seen in Table 2.2. Several of the experiments suggest that SVM is the optimum machine learning model to use for text categorisation tasks (Sarikaya et al., 2016; Firmino Alves et al., 2014; Chen et

Table 2.2: Summary of empirical research identified for ML in NLP. CA represents that the data used in the experiment is from a conversational context. ES indicates if the experiment carried out was empirical

Paper	CA	ES	Dataset Size	ML Models	Optimal ML
Draskovic et al. (2016)	Yes	No	N/A	Maximum Entropy	Maximum Entropy
Sarikaya et al. (2016)	Yes	No	400,000	SVM	SVM
Lee and Dernoncourt (2016)	No	No	362,000	RNN, CNN, Majority Class, SVM, Graphical Model, NB, HMM	RNN/CNN
Hijjawi et al. (2013)	No	Yes	1,318	NB, BN, OneR, ZeroR, J48	J48
Xu and Sarikaya (2013)	No	No	6,362	CNN, SVM	CNN
Nii et al. (2017)	No	No	8,313	CNN, SVM	CNN
Firmino Alves et al. (2014)	No	No	2,270	SVM, NB	SVM
Chen et al. (2012)	No	No	1,539	SVM, C45, Random Forrest, NB, K-NN	SVM
Bhargava et al. (2013)	No	Yes	27,565	SVM, HMM, CRF	SVM
Ding et al. (2015)	No	No	1,000	CNN, SVM	CNN
Gaikwad and Joshi (2016)	No	No	8,000	SVM, NB, K-NN	SVM
Hassan and Mahmood (2017)	No	No	259,855	CNN, RNN, SVM	CNN
Dumais et al. (1998)	No	No	9,982	Find Similar, NB, BN, DT, SVM	SVM
Imane and Mohamed (2017)	No	No	65,843	SVM, DT, adaBoost (DT)	DT
Li and Shen (2017)	No	No	6,000	NB, SVM, Logistic Regression	SVM
R. Zhang et al. (2017)	No	No	6,174	SVM, RBL	SVM

al., 2012; Bhargava et al., 2013; Gaikwad & Joshi, 2016; Dumais et al., 1998; Li & Shen, 2017; R. Zhang et al., 2017). However, many of the newer papers compare CNN models to older models (including SVM), and they suggest that CNN models offer better performance (Lee & Dernoncourt, 2016; Xu & Sarikaya, 2013; Nii et al., 2017; Ding et al., 2015; Hassan & Mahmood, 2017). Only two experiments could be identified which focus on the intersection of ML, NLP, and conversational agents (Draskovic et al., 2016; Sarikaya et al., 2016). Of these, only Sarikaya et al. (2016) conducted an experiment where a conversational agent dataset was evaluated with an ML model, SVM in this case. However, no other ML algorithms were evaluated in this experiment. The remaining experiments are conducted on general NLP datasets. Three experiments suggest that a DT model provided optimum results (Draskovic et al., 2016; Hijjawi et al., 2013; Imane & Mohamed, 2017). However, only one of these compared results to SVM models, and no comparison was made to CNN models. Additionally,Draskovic et al. (2016) detail a conversational agent system based on a DT model, with no consideration of other models or experiment conducted.

2.5 Supervised Machine Learning Algorithms

2.5.1 Support Vector Machine (SVM)

The basic principle of SVM models is the construction of a hyperplane which separates the data between two classifiers with the largest margin. Traditional machine learning algorithms mainly aim to minimise error on the training dataset in an approach called Empirical Risk Minimisation (ERM). SVMs, on the other hand, follow the principle of Structural Risk Minimisation (SRM) where the models' complexity is balanced with its success at fitting the training data. SRM based algorithms are generally regarded as providing better generalisation on unseen data than ERM based models. (Byun & Lee, 2002)

Linear SVM models are applied where the data representing the two classifiers under examination is linearly separable. However, linear classification models can

find it difficult to classify real-life problems. To combat this difficulty with linear models, a kernel function can be applied to SVM in order to provide a non-linear translation to the data. By applying a linear function, the data can be mapped into a higher dimensionality feature space which can allow for better linear separability. A representation of a linear SVM model can be seen in Figure 2.4 (Byun & Lee, 2002).

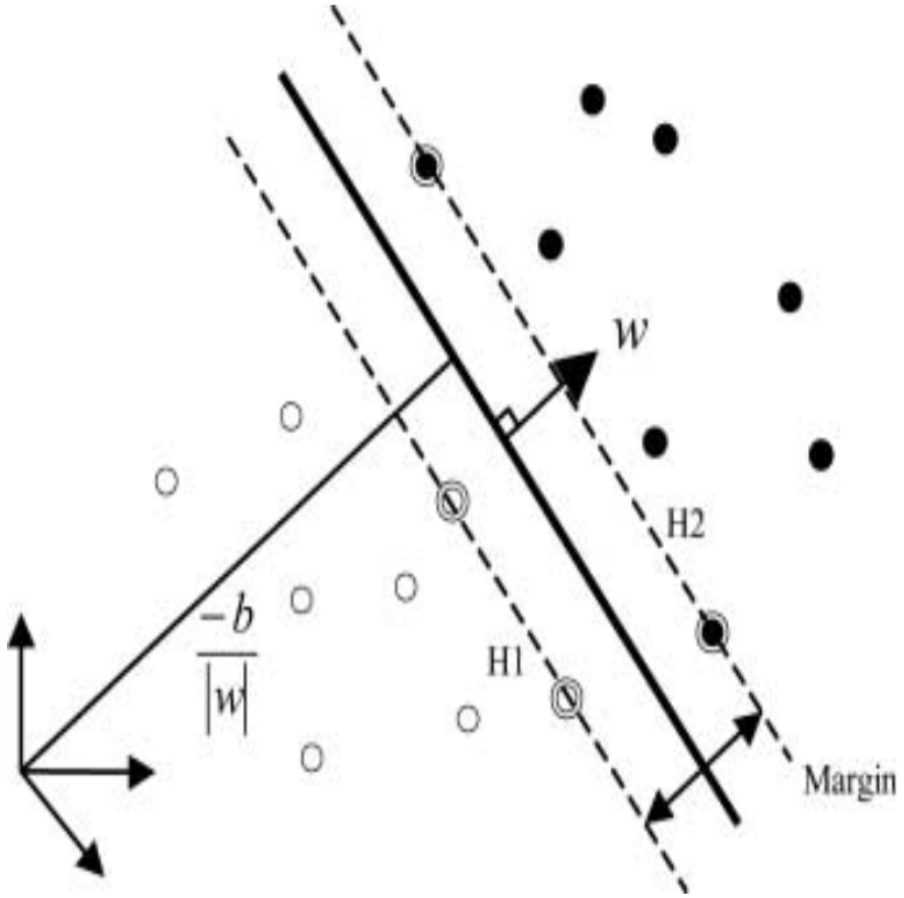

Figure 2.4: Representation of linear separating hyperplanes for the separable case. The support vectors are circled. (Burges, 1998)

There are a number of non-linear kernels defined for SVM but some of the most popular are Radial Basis Function (RBF), Polynomial, and Sigmoid. The equations for each of the kernels is presented below (Ayat, Cheriet, & Suen, 2005). The symbols;

r, d, and γ represent the parameters for the kernels.

- **Linear:** $K(X, Y) = X^T Y$
- **RBF:** $K(X, Y) = exp(XY^2/2\sigma^2)$
- **Polynomial:** $K(X, Y) = (\gamma X^T Y + r)^d, \gamma > 0$
- **Sigmoid:** $K(X, Y) = \tanh(\gamma X^T Y + r)$

While traditionally SVM is based on a two-class problem, they can be extended to multi-class problems. There are two primary approaches for achieving this: one versus one (OVO), and one versus rest (OVR). In the OVO mechanism each classifier is trained and evaluated against every other classifier and the best the results are amalgamated. In OVR each classifier is evaluated against a grouping of all other classifiers and the results are combined. (Byun & Lee, 2002)

2.5.2 Convolutional Neural Network (CNN)

CNN's can achieve strong results in the context of sentence classification within NLP (Y. Zhang & Wallace, 2017). As such many recent papers have examined the use of CNN's for multiple NLP tasks, and often compared the results to other machine learning algorithms. An example CNN architecture can be seen in Figure 2.5. A typical CNN architecture will accept a vectorised input which could contain multiple dimensions. This is followed by several different layers which can be constructed in many configurations, the primary layers are; Convolutional Layer (CN), the Pooling Layer (PL), and the Fully Connected Layer (FC). The convolutional layers are made up of a number of filters of a defined size which move over the input vector to create smaller transformed vectors from the input. The CN is typically followed by a pooling layer, which often utilises max or average pooling. The pooling layers transform the output from the previous layers into smaller inputs (often of a single neuron) for the following layer. Max pooling takes the highest value out from the previous layer, and average pooing takes an average of the output values. The pooling layer may subsequently be followed by a number of fully connected layers which represent a more

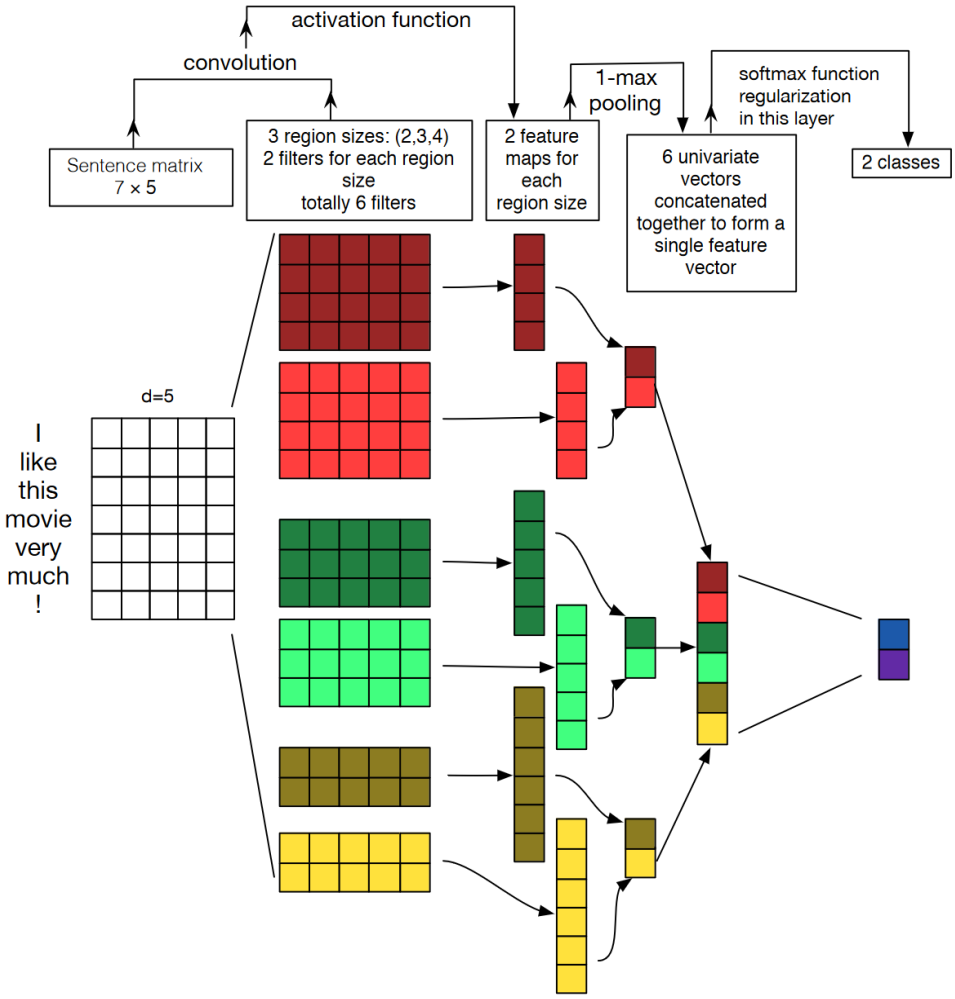

Figure 2.5: Illustration of an example CNN architecture with multiple filters and pooling layers (Y. Zhang & Wallace, 2017)

traditional neural network architecture. In this layer, there exist multiple neurons which are each connected to every other neuron with a weight variable. The inputs from the previous layer are processed with the connecting weights for the connection along with the activation function for that neuron. Fully connected layers are often employed to define the final output from the model. To combat over-fitting, a dropout is sometimes included in the FC layer. With a dropout layer, a random portion of the neurons in the layer are ignored during each training iteration.

CNN Architectures

Collobert and Weston (2008) describe a general architecture for multi-class CNN's in the context of NLP text categorisation. A representation of this CNN architecture can be seen in Figure 2.6. The architecture delineated takes a vectorised representation of text as input. This feeds into a convolutional layer with a number of filters of a size determined by the input. This is followed by max pooling layer. An optional fully connected layer is subsequently suggested. Finally an output layer with softmax activation with an output unit per class. This general model is utilised as the basis of a number of NLP related tasks on a large text corpus. The architecture is optimised for each task by including/excluding elements and utilising linear or non-linear activation functions for the convolutional and fully connected layers. However, there is no direct comparison between the results of the models against models built with different algorithms, so the true effectiveness is difficult to determine.

In Y. Zhang and Wallace (2017) a baseline CNN architecture is defined, and several options are evaluated for word-embedding, filter sizes, number of filters, activation function, pooling strategy, and regularisation. In order to evaluate each option individually, a baseline model is created and evaluated. The options selected for evaluation are thereafter trained and evaluated in turn, comparing the results to the baseline. Each evaluation is carried out using ten-fold cross-validation.

The results of the word-embedding analysis carried out by Y. Zhang and Wallace (2017) were reported previously. In summary, it was reported that either Word2Vec or GloVe generated the best results, dependent on the dataset under evaluation.

The experiment also examines multiple filter sizes along with combining several feature sizes in a single model. The optimal filter size is found to depend on the dataset being evaluated, with each having an optimum size. It is also reported that multiple filters of different sizes do not inherently result in better performance. This is especially evident when filters with substantial deviation from the optimum for the dataset are included. In relation to the number of filters, it is also reported to be dependent on the dataset (Y. Zhang & Wallace, 2017).

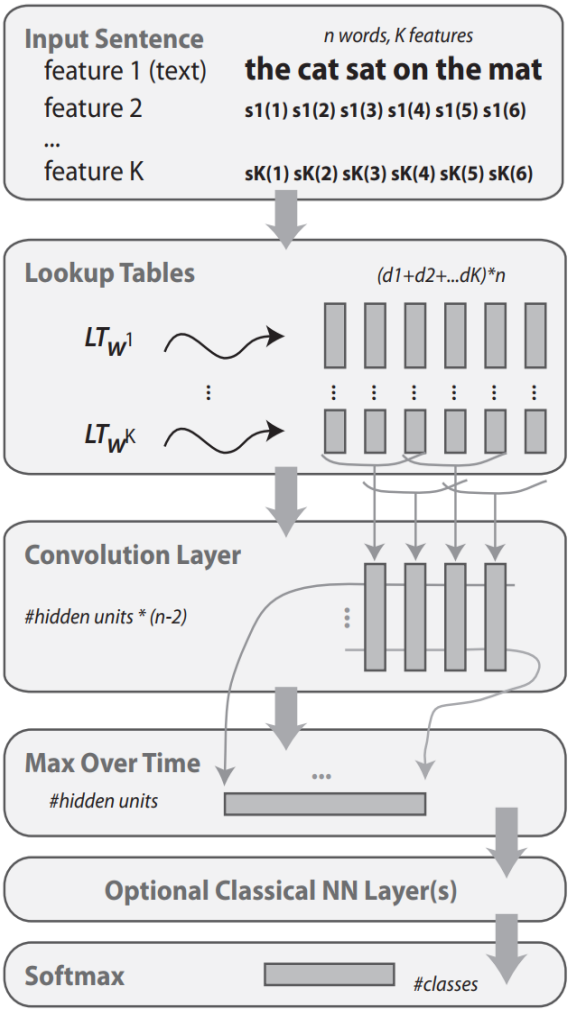

Figure 2.6: Proposed general CNN architecture for NLP tasks (Collobert & Weston, 2008)

For pooling strategy multiple sized pools using both average and max pooling. It is reported that max pooling with a single output is the optimum strategy for the datasets and models evaluated. Due to poor performance and slow training times the average pooling experiments were not completed (Y. Zhang & Wallace, 2017).

In respect to regularisation both dropout (with multiple rates) and L2 norm are evaluated. The effect of dropout is found to be dependent on the dataset, but for many of the evaluated datasets, it was found that dropout and L2 norms had little impact on the effectiveness of the model, but it is pointed out that it may have more

of an effect with multiple layer models. The suggestion is to utilise a small dropout rate between 0.0 and 0.5 (Y. Zhang & Wallace, 2017).

The activation functions considered are ReLU, tanh, sigmoid, SoftPlus, Cube function, and tanh cube. ReLU and tanh are reported as achieving the best performance for eight out of the nine evaluated datasets. SoftPlus performed best on the remaining dataset. It is suggested that for shallow models, exploration of both ReLU and tanh be carried out (Y. Zhang & Wallace, 2017).

2.6 Summary of literature review

Conversational agents systems originated in the 1960's with simple pattern matching mechanisms for dealing with user text input. Over time these pattern matching systems evolved with the utilisation of tools like regex and AIML. Within the context of these systems, there are many different components which combine in order to form a modern agent. The use of machine learning algorithms has become a significant area of focus within the conversational agent domain and there are several areas where natural language processing can be applied to the components of conversational agents, including the area of Intent Identification.

A modern NLP pipeline typically consists of several steps; tokenization, lexicon normalisation, noise reduction, and feature extraction. Many algorithms can be applied to each of the steps. TF-IDF is effective for feature extraction, and lemmatisation for lexicon normalisation. Some tools can be applied for noise reduction but the actions to be carried out are largely dataset dependent. Several papers have been identified which compare ML models for text classification in an NLP context. However, only a single experiment has been identified which conducts an evaluation of any ML models against a conversational agent dataset for the purposes of intent detection (Sarikaya et al., 2016). In this experiment only SVM models are considered and evaluated. Several of the identified experiments in NLP have identified SVM models as offering the best performance for text categorisation: (Sarikaya et al., 2016; Firmino Alves et al., 2014; Chen et al., 2012; Bhargava et al., 2013; Gaikwad & Joshi, 2016; Dumais et al.,

1998; Li & Shen, 2017; R. Zhang et al., 2017). However, several more recent papers in NLP have identified CNN models (Lee & Dernoncourt, 2016; Xu & Sarikaya, 2013; Nii et al., 2017; Ding et al., 2015; Hassan & Mahmood, 2017) as offering better results than other ML algorithms, including SVM models. In addition, very few papers could be identified which consider the optimal classification technique within the context of conversational systems. Finally, few of the identified papers conduct a thorough statistical analysis of the results when comparing ML classifiers in an NLP context.

SVM models attempt to construct a hyperplane which separates data with the largest margin. When the data is linearly separable a linear kernel can be applied to the data in order to identify the effective hyperplane. In cases where the data is not linearly separable, non-linear kernels like polynomial, RBF, and sigmoid can be applied. Each kernel has a number of parameters which can be employed to optimise the performance of SVM models. Typically SVM models are applied for binary classification problems, but through the use of OVR and OVO mechanisms, SVM can also be applied to multi-class problems.

CNN architectures can be made up of several different layers, typically consisting of; convolutional layers, pooling layers, and fully connected layers. CNN's can be applied to many different areas including text classification in NLP. Collobert and Weston (2008) has defined a CNN architecture for carrying out general NLP tasks. Like SVM there are a number of options which can impact the performance of the CNN models, from the size and number of filters, to the activation and optimisation functions. The parameters utilisation are dependent on the overall architecture of the model, the data under evaluation, and the problem being solved.

2.6.1 Gaps in research

The identified research suggests that there are two machine learning algorithms at the forefront of NLP research for the purposes of text classification; Support Vector Machines and Convolutional Neural Networks. Most literature in which both CNN and SVM models are compared assert that in the context of NLP tasks, CNN models can achieve better performance than SVM models. In the majority of cases where CNN

models are not evaluated, SVM models are indicated as offering the best performance. Additionally, only SVM models have been evaluated within the context of NLU for conversational agents. However, there are few identified experiments which compare the two algorithms in a rigorous manner, by conducting detailed statistical tests on the results and considering multiple metrics.

By designing and conducting a comprehensive and rigorous experiment to evaluate the performance of both algorithms, and including statistical tests to evaluate and compare the results, this paper proposes to determine if CNN models are superior to SVM models in the area of intent detection. Additionally, by building and evaluating the models on a dataset which is generated by, or closely mimics the interaction of users and a modern conversational agent, this experiment proposes to be a valid evaluation of the utilisation of ML algorithms within the context of NLU for conversational agent systems.

2.6.2 Research Question

> "Are CNN more effective than SVM at determining intent, for use in conversational agents?"

This research question will be investigated in detail in the next chapter through the formalisation of a research hypothesis and the design of an experiment to test that hypothesis.

Chapter 3

Experiment design and methodology

Experiment design and methodology begins by defining a research hypothesis based on the research question defined in the previous chapter. An experimental design is presented which is designed to ultimately approve or reject the defined hypothesis.

3.1 Hypothesis definition

The research question that this paper has identified suggests that CNN models may generate better results than SVM models when used as an intent detection mechanism for conversational agent systems. A supposition that can, therefore, be drawn from this research question is that CNN models produce better results when trained and evaluated using a conversational agent dataset. To accept the hypothesis, the results from evaluating the models must be better for CNN models when compared to SVM models. To define a hypothesis that can ultimately be accepted or rejected through an experimental implementation, measure-able metrics which define 'better results' need to be specified. By utilising several metrics, the hypothesis will require a comprehensive experiment for it to be accepted or rejected. Hence, the answer to the research question could also be considered to be more comprehensive. The three metrics which the hypothesis will define to determine if the CNN results are better are;

precision, recall, and f1 score. Precision is a measurement of the agreement between positive labels assigned by the classifier compared to the actual data labels. Recall is the effectiveness of the classifier at identifying positive labels. F1 is a measurement of the relationship between the positive labels assigned by the model when compared to the actual labels (Sokolova & Lapalme, 2009).

By defining a hypothesis based on the supposition that CNN models are better, by using a conversational agent dataset to provide context, and by measuring and comparing the defined metrics for the models using statistical tests, a hypothesis is defined which can ultimately be accepted or rejected through an experimental process. By accepting or rejecting the hypothesis, the research question can be answered. If the hypothesis is accepted, the answer to the question is that 'yes', CNN models produce better results in this context. Otherwise, the answer becomes no, CNN models do not necessarily produce better results. A graphical representation of the thesis can be seen in Figure 3.1.

- **H1:** CNN models achieve a statistically higher (with 95% confidence) precision, recall, and F1 score than SVM models when utilised to determine the intent of users in a conversational interface dataset.
- **H0:** CNN models do not achieve a statistically higher (with 95% confidence) precision, recall, and F1 score than SVM models when utilised to determine the intent of users in a conversational interface dataset.

3.2 Methodology

The experiment will follow many of the steps outlined in the CRISP-DM model (Shearer, 2000). The steps in question are Data Understanding, Data Preparation, Modelling, and Evaluation.

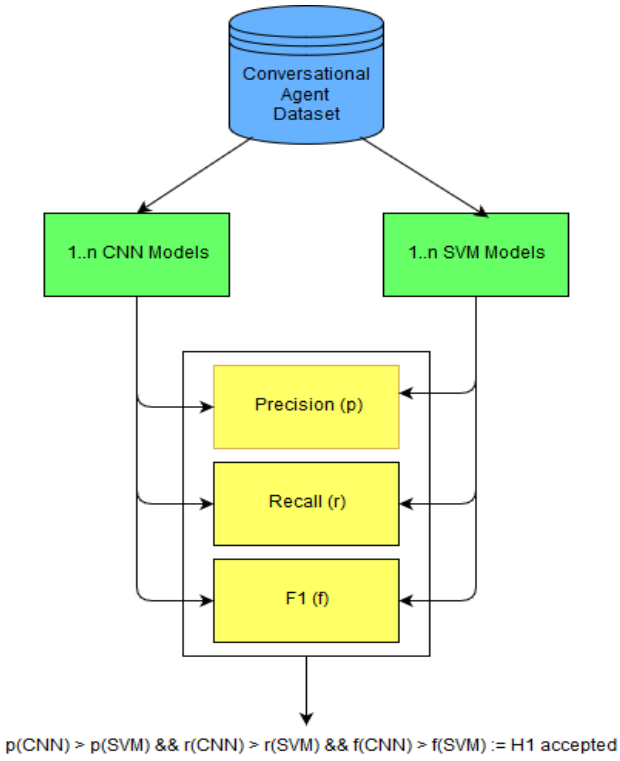

Figure 3.1: Graphical representation of research hypothesis

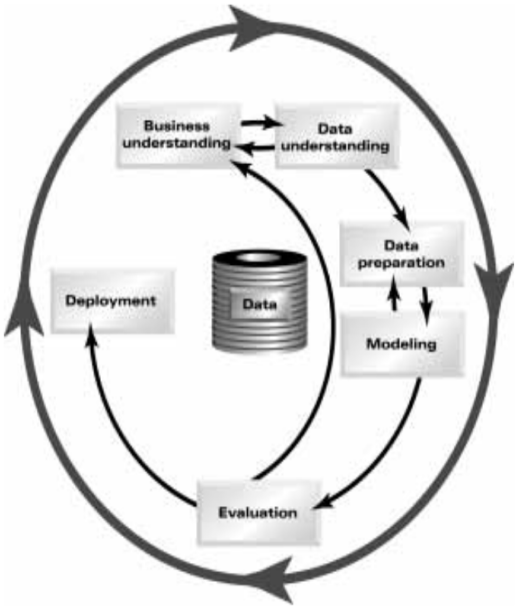

Figure 3.2: Representation of the CRISP-DM model for data analysis (Shearer, 2000)

3.3 Experiment Overview

The experimental process carried out is illustrated in Figure 3.3. Here the NLP data preparation steps can be seen as Lexicon Normalisation, Noise Reduction, Feature Extraction, and splitting the data. Next, the modelling steps can be seen where the SVM and CNN grid search and optimised modelling tasks are carried out. Finally, the evaluation step is designed to compare and evaluate the results from the modelling steps to ultimately accept or reject the hypothesis. Not represented in the figure is an initial step where an understanding of the dataset utilised for the experimental evaluation is gathered.

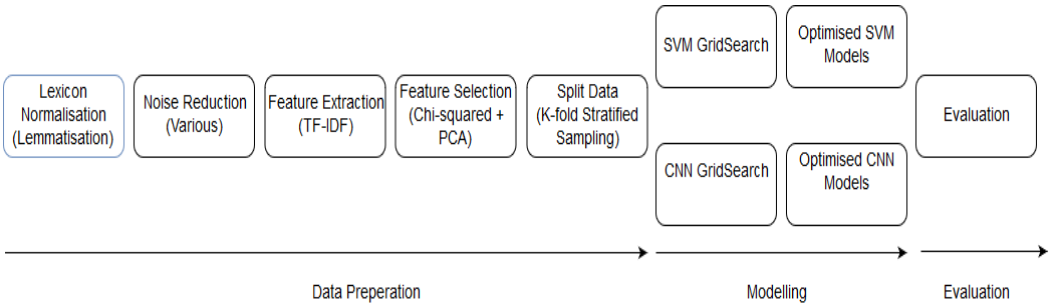

Figure 3.3: Overview of the experimental process with the associated CRISP-DM phases on the bottom and the NLP, modelling, and evaluation steps illustrated as blocks in the chain.

3.4 Data Understanding

The data understanding step is dedicated to gathering a greater understanding of all data elements that make up the dataset.

3.4.1 The Dataset

The Maluuba Frames dataset (El Asri et al., 2017) will be employed to evaluate the effectiveness of the algorithms in this context. The dataset consists of 19,986 individual statements which were collected in a Wizard-of-Oz fashion. Two users communicated

via a chat interface for the purpose of booking flights and accommodation, where one user took the place of the computer agent in the conversational interface paradigm.

The dataset consists of a single file containing a JSON representation of the Frames data. This data includes many data points but for intent detection, only the statements entered by the users, and the primary intent identified for that statement, will be extracted. Every statement and its matching intent will be extracted from the JSON file, and stored for review and processing. If a statement has no identified intent, then the intent will be set to 'NO_INTENT'.

3.5 Data Preparation

In the data preparation step, the individual statements will be cleaned to remove bad or noisy data, the text will be converted into vectors for modelling, and the data will be split so that separate data is utilised for training and evaluating the models.

3.5.1 Missing Data

Any statement which has been associated with the 'NO_INTENT' intent will be excluded from the dataset before any further data preparation actions are carried out. The statements without an identified intent may have no common features which could be used for classification. Without an identified commonality to the statements, they could be misclassified by the models, leading to poorer performance scores.

3.5.2 Lexicon Normalisation

Comparing the two examples utilised to illustrate Stemming and Lemmatisation in the literature review, we can see that using a Lemmatisation process on better and good would result in no change to the date. However, using Lemmatisation on 'studies' and 'studied' would result in their replacement with 'study'. As a result, a Lemmatisation process will be applied to the dataset to perform lexicon normalisation.

3.5.3 Noise Reduction

By utilising the Python Natural Language Toolkit (NLTK) and its POS Tagging system, a majority of the nouns representing geopolitical entities (GPE) can be identified. GPE is the term employed by the NLTK toolkit to identify location names. As the tagger analyses sentence structure, it can correctly identify fictional place names as GPEs. However, it does also identify some non-nouns as GPEs because of their context within the sentences. Each word identified by the POS Tagger will be replaced with the token: [GPE]. By replacing the word with a token we keep the high-level meaning of the word in the sentence, but reduce the variables in the dataset.

Date and time information occurs in the dataset in order to indicate periods when holidays could be taken and to indicate when flights and accommodation can and have been booked. The specific date related information that is provided for information could impair the accuracy of the model. However, the presence of date information could be valuable to the model in determining intent. By replacing the date information with a generic token we provide the model with enough information to indicate that date information was supplied, while reducing noise to the model with unique tokens representing each value. The data will be manually reviewed and common date and time patterns will be identified. A regex will be defined which covers the determined patterns. The regex will be executed against each statement, and matching patterns in the statement will be replaced with the token [DATETIME]. Due to the nature of the free-form input in conversational interfaces, and the volume of data, it is not possible to identify a pattern which covers 100% of cases.

Currency based information is used in the dataset to inform about budgets and to communicate actual prices. As with the date information discussed above, a regex pattern will be identified which matches the majority of date and time patterns that have been pinpointed in the data. Every matching instance of currency will be replaced with the token: [CURRENCY]. As with the date time date, it will not be possible to identify a pattern which covers 100% of cases.

Punctuation characters occur frequently in written text but may not provide much

information to a classification model. The text will be reviewed in order to identify common punctuation characters. Each of the identified punctuation characters will be removed from the text.

Commonly employed words, also known as stop words, may not be relevant to the context of the statements and therefore are often described as noise for the purposes of classification exercises. A common list of stop words will be identified from reviewing the data. This list will be built by reviewing common stop words from the English language combined with the most common words which occur in the data. The stop words will take into account the context of the data. For example, this is a travel-related dataset and a common pattern which could occur is 'from [GPE] to [GPE]', in this context it may be better to keep from and to in the dataset.

A common feature of many chat interfaces are emoji. As the dataset was collected using a chat interface, the text will be reviewed for the presence of emoji characters. Some emoji can be utilised to convey information but that is not always the case. During the review, it will be determined if it is beneficial to retain or remove the emoji characters from the statements. If it is identified as beneficial, a pattern will be determined which can be used to identify and remove the characters.

3.5.4 Feature Extraction

Both the SVM and CNN algorithms accept vectors of a fixed size as their input. A vectorisation technique will be applied to convert the statements into fixed-length vectors. There are a number of algorithms utilised in the NLP space for the purposes of converting text to vectors.

The Bag-of-words (BOW) model is a vector representation of text where every existing word in the corpus is represented by a single integer number in the vector. This model disregards grammar and word order but does maintain multiplicity. For example, the phrase; "To be or not to be, that is the question" contains eight distinct words, resulting in a vector with eight columns. The columns representing 'to' and 'be' would contain a value of two, and all other columns would contain a value of one.

TF-IDF is a numerical factor which represents the importance of a word (or set of

words) in a statement in relation to its importance (frequency) in the overall corpus. The TF-IDF algorithm value increases as words are employed more frequently in the statement being analysed, but this is offset by the number of occurrences in the overall corpus. This effectively applies a weighting to de-prioritise words which occur very frequently in the overall dataset. In addition, to single words (unigrams), TF-IDF can also be utilised to consider combinations of words as bigrams (two words), trigrams (three words), etc.

Word2Vec is a collection of algorithms in the word embeddings space where shallow neural networks are used to generate a vector space from a corpus of text. The vector representations which have a common context within the corpus are located in close proximity within the vector space. Word2Vec contains two primary algorithms cbow and Skip-gram.

TF-IDF has advantages in this context over BOW in that it weights words based on their frequencies both in the statement being categorised and the overall text corpus. In addition, the ability to consider not just single words but pairs of words can help with accuracy in an NLP context. Word2Vec has similar advantages and has recently become a popular choice for feature extraction tasks. However, Word2Vec cannot be directly applied as an input for an SVM model due to the nature of its output, as it generates multi-dimensional vector representations of the text. Instead, the generated vectors would need to undergo another transformation to reduce the vectors representing words into a representation with a single dimension. CNN models, on the other hand, can make use of two-dimensional convolutional layers in order to process the output of Word2Vec models directly. The experiment by Lilleberg et al. (2015) also suggests that TF-IDF can generate better results than using Word2Vec when building linear SVM models.

Because of the extra transformation step needed to carry out Word2Vec with SVM, TF-IDF will instead be employed as the input as it is supported by both SVM and CNN models. The feature extraction step of the process will generate a vector space where each column represents a unigram or bigram from the overall corpus. Until the data cleaning exercise is completed and the feature extraction process has been

executed, there is no way to determine its size.

3.5.5 Feature Selection

The TF-IDF feature extraction process outlined above will create at least a single feature per word in the dataset, resulting in an enormous vector representing each record. Large vectors will increase the time required to generate machine learning models. In addition, by reducing the features to only the most important ones which can be identified as important for the intents being identified, the noise is further reduced for the modelling process. Therefore, a feature selection process will be applied to reduce the vector size to support faster modelling and evaluation. In this case, two feature selection mechanisms will be employed Chi-squared and PCA.

Firstly, a Chi-squared test will be run for every feature in the vector space. The Chi-squared test is a measure of independence of two features. In this case the independence of every feature with the intent. A significance value can be determined for each feature, the lower the significance value, the more correlated that feature is to the intent. Any feature which indicates a significance of ≤ 0.05 will be retained and the remainder will be discarded.

Secondly, PCA will be applied to further reduce the vector size. PCA transforms the possibly correlated features in the vectors into uncorrelated features called principal components. A PCA will be run on the complete vector representation from the Chi-squared algorithm. One of the outputs from a PCA is the explained variance. By graphing the explained variance as a scree plot, the impact of the principal components can be observed. The number of significant principal components for the dataset can be determined by observing the impact of the data in the scree plot.

3.5.6 Imbalanced Data

As seen in Figure 4.1, the classes in the dataset are highly unbalanced. There are several potential techniques for addressing sample imbalance. Over and under sampling are the primary mechanisms utilised, where oversampling adds more minority records,

and under-sampling removes majority records until all classes are in balance. A hybrid approach is also possible where the data is first over-sampled and then under-sampled to provide a balanced dataset.

For oversampling, Synthetic Minority Over-sampling Technique (SMOTE) can be applied to generate new artificial records for the dataset. To generate synthetic data k-nearest neighbours are determined for each data point and new data points are generated between the data point and its neighbours. However, due to the nature of NLP and its vector representations, this is not a viable approach for this experiment. Random oversampling, however, is a potential option where the minority classes are sampled with replacement until the dataset is balanced.

With under-sampling, Tomek links can be applied to remove very similar records from the dataset. With this algorithm, the k-nearest neighbours are mapped and very closely mapped records can be removed as they 'overlap' with other data points. Another option for under-sampling is simply to randomly remove data points from the majority classes until a balance is achieved. A drawback with under-sampling is that you are excluding valid data records from the dataset.

In the dataset, there is a vast disparity between the number of samples in the minority class (14) and the majority class (8,239). If oversampling is applied, approximately 160,00 new records would be created by random oversampling. This is eight times larger than the original dataset and as such oversampling will not be carried out. If under-sampling a vast majority of the data in the dataset would be discarded to balance the dataset to the smallest class. As such under-sampling will not be carried out. A hybrid approach would result in less artificial data being created, and less data being discarded, but ultimately still suffers from the issues with both approaches.

Instead, the modelling and evaluation steps of the experiment will attempt to take into account the imbalance. The modelling step will apply a weighted error value for each class to apply a greater penalty when incorrectly identifying the minority classes. The scikit-learn compute_class_weight class will be employed to determine the appropriate weights for each class. Also, the F1 score will be utilised as the primary metric for evaluation as it is less susceptible to imbalanced data when compared to

precision or recall alone.

3.5.7 Training and Evaluation Data

K fold stratified cross-validation will be employed to evaluate each model. The dataset will be split into ten separate sections, and each section will retain the same representation of the classes as the overall dataset. All models will be trained and evaluated ten times. For each training and evaluation, nine of the ten sections will be utilised to train the model, and the final section will be utilised to evaluate the model. For every iteration, a different section will be employed to evaluate the dataset so that every record in the data is used for evaluation once.

3.6 Modelling

The goal of this experiment is to compare the SVM and CNN machine learning algorithms for intent detection. Both algorithms have multiple options. As a result, several different configurations of each model will be trained and evaluated on the dataset.

3.6.1 SVM

The python scikit-learn packages SVC module will be employed to build the SVM models. There are a number of different parameters that can be provided for building SVM models and each can have an impact on the ability of the SVM model to identify the intents. There are four SVM kernels available in the scikit-learn package. Each kernargel represents a different algorithm which is applied for pattern analysis within the SVM implementation. The kernels are Linear, Polynomial (Poly), RBF, and Sigmoid. Some of the other variables which can be applied to the SVM process are dependent on the kernel.

For this experiment, the best parameters for each of the four kernels will be identified, and those parameters will be applied when training a model for each of the

kernels. To identify the best combination of parameters, a grid-search will be employed. The grid-search process will use the prepared vectors and two-fold stratified cross-validation to execute each kernel with every possible combination of parameters (for that particular kernel). The F1 score will be utilised as the evaluation metric as it represents a composite evaluator for both the precision and recall. This results in every combination of parameters being trained and evaluated two times. The F1 score will be calculated and averaged for every iteration of each parameter combination, and the combination with the highest averaged F1 score will be applied for SVM modelling for the experiment.

The reason two-fold stratified cross-validation is employed for the grid-search instead of ten-fold, is down to time required to run the entire search. Preliminary tests of SVM model generation using the dataset in a ten-fold configuration showed that each model generation and evaluation took approximately thirty minutes, resulting in over one-thousand, seven-hundred and ninety-five minutes of CPU execution time (or over seventy-four days). The dimensionality of the dataset had been reduced to a minimum, therefore to control the time variable in the equation, it was found that the volume of data used to generate the model needed to be reduced. Using two fold validation, the time needed to train each model was approximately five minutes, and overall, fewer models would be generated.

The SVM algorithm is designed to work for only two classes, so every record is classified as one of the two classes. However, there are ways of using SVM for multi-class datasets like the one in this experiment, these are referred to as decision functions. There are two possible decision functions for the SVC module: OVO and OVR. OVO attempts to classify each classifier against each other, one at a time. OVR, on the other hand, attempts to classify a single classifier compared to all others as a single grouped entity. Both of the potential decision functions will be evaluated for each of the kernels.

The C, or penalty parameter is the error term which represents the balance between training error and testing error. This parameter is valid and will be evaluated for, each of the kernels. The following range of values has been identified for this parameter:

0.001, 0.01, 0.1, 1, 10, 100, 1000. Gamma represents the range of influence that each training vector has when building the model. This parameter is relevant for the poly, RBF, and sigmoid kernels only (not the linear kernel). The following range of values has been identified for this parameter: 0.01, 0.1, 1, 4, 16. Degree represents the flexibility of the decision boundary for the kernel. Higher values represent more flexible decision boundaries. The following range of values has been identified for this parameter: one, three, six, twelve.

Each kernel will be set up with a random seed number of 42. The class weights will be set to the values determined in the imbalanced data step. All other variables will employ the default values set by the SVC library. The default tolerance for the stopping criterion is 1e-3. Coef0 which is the independent term utilised by the poly and sigmoid functions which will have a default value of 0.0. A shrinking heuristic will be applied by the SVC library to speed up optimisation. The cache size and max iterations parameters will be determined during the experiment based on the resources available.

For the linear kernel, the potential parameters represent fourteen possible combinations of the decision function and C parameters. This represents twenty-eight models being generated and evaluated on the dataset for this kernel. The poly and RBF kernels parameters represent seventy parameter combinations for the decision function, C, and gamma. This equates to one-hundred and forty models generated for training and evaluation by the grid-search for each of the two kernels. The parameters for the poly kernel represent two hundred and eighty parameter combinations for the decision function, C, gamma, and degree parameters. This equates to five-hundred and sixty models generated for training and evaluation by the grid-search. This leads to a total of eight-hundred and sixty-eight training and evaluations by grid-search.

Once the optimal parameter combination has been identified for each kernel, each one will be trained and modelled against the ten, k-fold stratified cross-validation datasets. The precision, recall, and F1 score will be calculated for each of the iterations by using the scikit-learn tools in weighted mode. This will provide ten values for the three evaluation metrics, for each optimised kernel configuration. These results will be

recorded and utilised to compare and evaluate the effectiveness of the SVM models.

3.6.2 CNN

The python keras package utilising a Tensorflow backend will be employed to build the CNN models. There are several different patterns available when building CNN's, ranging from simple shallow models with a minimum number of layers to deep learning models with several convolutional and fully connected layers. Additionally, there are several options that can be used when building each model.

Five architectures have been identified through initial research and execution. These architectures are based on the work of Collobert and Weston (2008) as seen in Figure 2.6. The best initial results were found by having a large number of small filters with a pooling layer, followed by fully connected layers. The addition of a dropout layer in both the shallow and deep models had an adverse impact on the performance of the models and has therefore not been included. Once a shallow architecture with acceptable results was identified, it was extended to follow similar principles with multiple convolutional and fully connected layers to form a 'deep' architecture. A grid search will be carried out on a range of parameters to identify the best performing parameter combination for each architecture.

The grid search will utilise the entire training set in a two-fold cross-validation split. Each identified architecture configuration will be executed against every fold, and the F1 score will be calculated. The average F1 score for the folds will be employed to identify the best performing combination for that architecture. The identified optimal configuration for each CNN pattern will be utilised to evaluate the CNN classification mechanisms on the dataset.

Each of the five architectures will accept the transformed PCA records as input. Each will have a final fully connected layer of twenty units utilising a softmax activation. The twenty units of the final layer will result in twenty outputs, one for each of the intents identified for the data. The softmax activation will ensure that the sum of the output values for each record is 1. The output with the highest value is the predicted intent for that record. Each of the models will employ accuracy as its

evaluation metric for training the network. A categorical cross-entropy loss function will be applied to attain the expected categorical output for the final layer.

The options to be explored for each architecture are; pooling, activation, and optimiser. Both max and average pooling will be evaluated for each architecture. Each pooling layer in each instance will utilise the same pooling mechanism. The following list of activations will be evaluated: ReLU, softmax, elu, selu, softplus, softsign, tanh, sigmoid, hard_sigmoid, and linear. The same activation will be employed for each convolutional and fully connected layer of the architecture instance, with the exception of the final fully connected layer which will utilise softmax as defined previously. The following optimisation functions will be evaluated: adam, sgd, rmsprop, adagrad, adadelta, adamax, and nadam.

Every iteration will utilise an early stopping mechanism in order to terminate the training before over-fitting can occur. The early stopping callback will monitor for loss, will define a minimum change in loss of 0.005, and will terminate if there is no change to the loss value for the instance after ten epochs. Batches of 100 records will be supplied to each instance for training.

The first shallow architecture will consist of a single convolutional layer containing one hundred filters of a single dimension and will utilise a stride of one, and global pooling. This will be followed by the output layer which was previously discussed. This is represented visually in Figure 3.4.

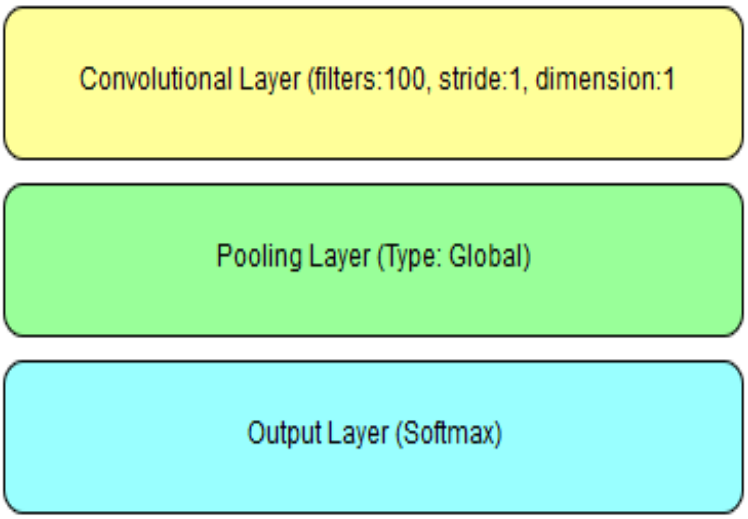

Figure 3.4: Representation of the layers of the Shallow_1 CNN architecture

The second shallow architecture will consist of a single convolutional layer containing one hundred filters of a single dimension and utilising a stride of one, and global pooling. This will be followed by a fully connected layer of one hundred units, which is followed by the output layer. This is represented visually in Figure 3.5.

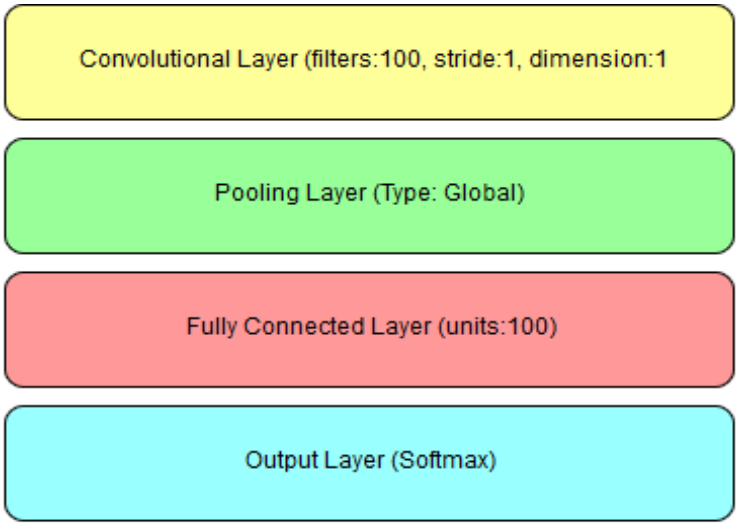

Figure 3.5: Representation of the layers of the Shallow_2 CNN architecture

The third shallow architecture will consist of a single convolutional layer containing one hundred and fifty filters of a single dimension and utilising a stride of one and will utilise global pooling. This will be followed by a fully connected layer of one hundred

and fifty units, which is followed by the output layer. This is represented visually in Figure 3.6.

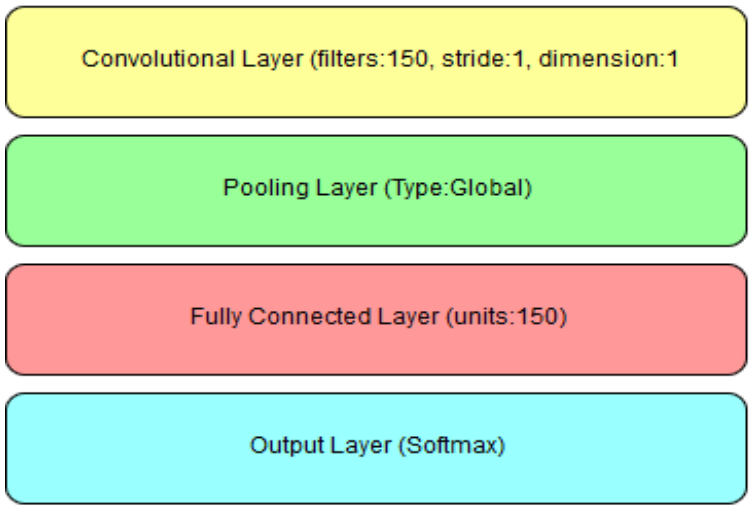

Figure 3.6: Representation of the layers of the Shallow_3 CNN architecture

The first deep architecture will consist of three convolutional layers, each containing one hundred filters of a single dimension and utilising a stride of one and utilising non-global pooling with a width of a single unit. These layers will be followed by a global pooling layer. This will be followed by a fully connected layer of one hundred units, which is followed by the output layer. This is represented visually in Figure 3.7.

The second deep architecture will consist of six convolutional layers, each containing one hundred filters of a single dimension and utilising a stride of one and utilising non-global pooling with a width of a single unit. These layers will be followed by a global pooling layer. This will be followed by two fully connected layers, each of one hundred units. This is followed by the output layer. This is represented visually in Figure 3.8.

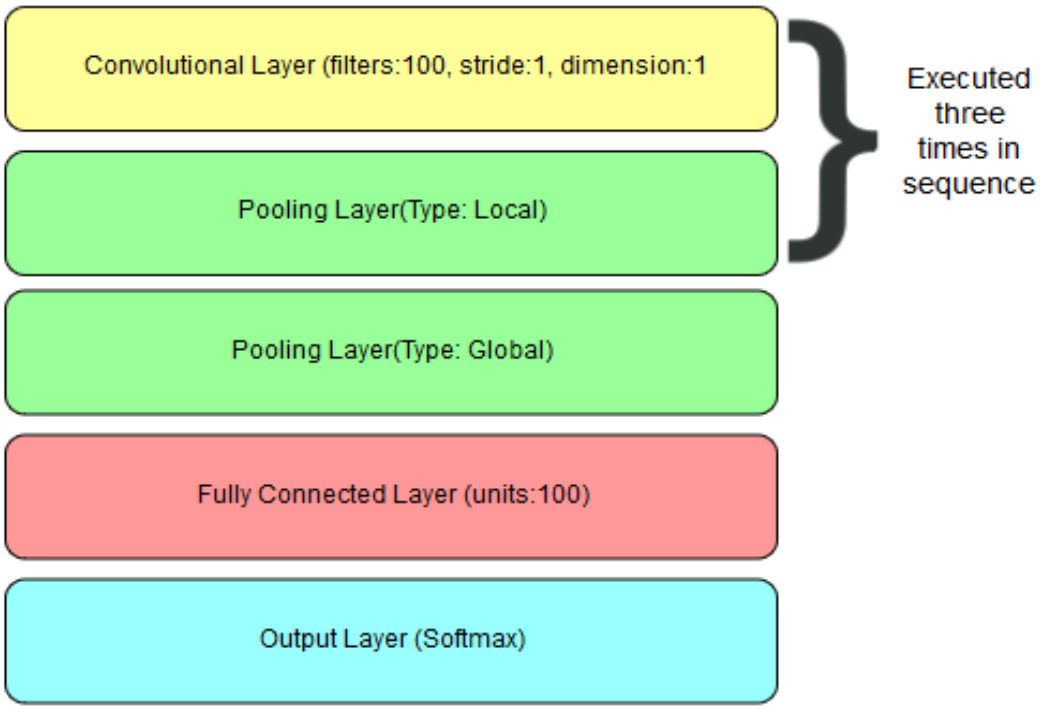

Figure 3.7: Representation of the layers of the Deep_1 CNN architecture

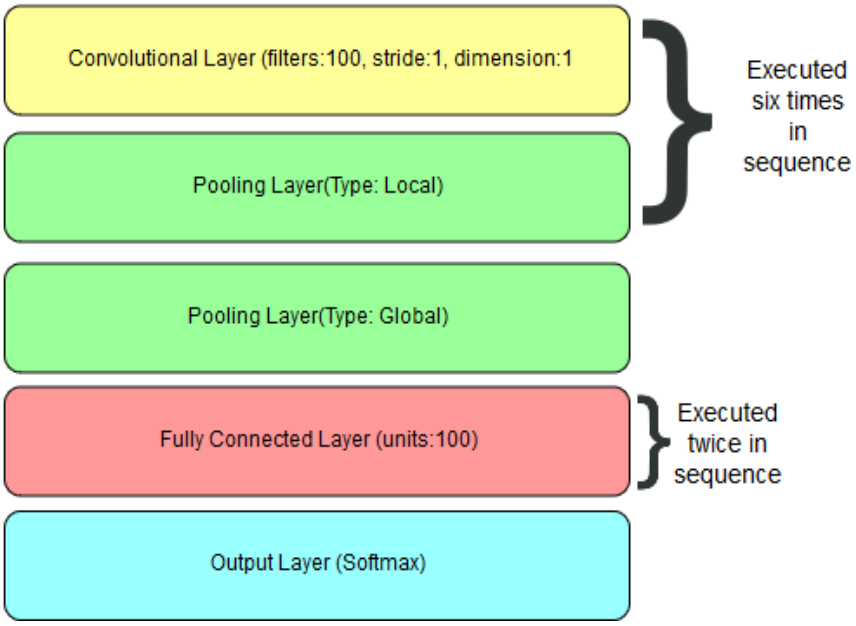

Figure 3.8: Representation of the layers of the Deep_2 CNN architecture

There are 112 combinations of parameters for each of the five models, resulting

in 700 total combinations. Each will be executed twice due to the two-fold stratified cross-validation, resulting in 1,400 models being executed and evaluated. As with the SVM grid search, the F1 score will be employed as the evaluation metric. The scikit-learn packages, f1_score class, will be utilised to calculate a weighted F1 score due to the multi-class nature of this dataset. The score will be calculated for each model execution and the average over the two folds will be calculated. The individual model with the best F1 score for each of the five architectures will be used to evaluate the effectiveness of the CNN models on this dataset.

Once the optimal parameter combination has been identified for each architecture, each one will be trained and modelled against the ten, k-fold stratified cross-validation datasets. The precision, recall, and F1 score will be calculated for each of the iterations by using the scikit-learn tools in weighted mode. This will provide ten values for the three evaluation metrics, for each optimised architecture configuration. These results will be recorded and utilised to compare and evaluate the effectiveness of the CNN models.

3.7 Evaluation

Four core SVM models have been identified (Linear, Polynomial, RBF, Sigmoid). Five CNN models have been identified (three shallow and two deep). This provides nine models to be evaluated. Once the optimal parameter set for each of the identified models has been identified, every model will be trained and evaluated against each of the folds in the defined ten-fold cross-validation set, utilising the optimum identified parameters. The F1 score, precision, and recall will be calculated for each iteration of every model, resulting in a total of thirty evaluation metrics per model.

For each of the nine datasets, a ShapiroWilk test will be executed to determine the normal distribution of each result set. Thereafter, further statistical tests will be carried out in order to determine if there are significant differences between the results for each model. Dependent T-Tests will be executed comparing each result for every model, to the matching results for all other models. The Dependent T-Test will be

executed if the data is normal and there are no significant outliers. If the assumptions are not met then Wilcoxon signed-rank tests will be carried out instead.

If the executed test indicates that the CNN model results offer a significant improvement (with 95% confidence) over the SVM model results for each of the three metrics, the null hypothesis will be rejected, and H1 will be accepted. Otherwise, if the tests indicate that there is not a significant improvement for any metrics, the null hypothesis will be accepted, and H1 will be rejected.

3.8 Summary of design

A number of data preparation steps will be carried out on the dataset in order to prepare for the modelling step. Firstly records with no intent will be discarded. This is followed by a Lemmatisation process will be carried out in order to determine the root of words in the dataset. A noise reduction process will be enacted to replace instances of dates, times, and currencies. Punctuation and emoji characters will also be removed.

Once data cleanup has been completed, a TF-IDF algorithm will be applied in order to convert the text statements into vectors. In order to reduce the dimensionality of the dataset, a Chi-squared test will be executed to identify the significant variables in the data and the remaining non-significant variables will be ignored. Finally, a PCA will be run to identify the most effective transformed vectors.

The prepared dataset will be split using ten-fold stratified cross-validation in order to provide ten splits of the data for training and evaluation. As the data is imbalanced, a weighting calculation based on the occurrences of the intents within the dataset. These weights will be supplied to each model generation in order to combat the imbalance within the data.

Four SVM kernels are available in the chosen toolkit. A GridSearch will be carried out in order to identify the optimum parameter combination for each kernel. The grid search will be carried out using a two-fold cross-validation pattern. The F1 score will be employed as the single evaluation metric for both grid search actions.

Five CNN algorithms have been identified for the experiment. A GridSearch will be carried out in order to identify the optimum parameter combination for each algorithm. The grid search will be carried out using a two-fold cross-validation pattern. The F1 score will be used as the evaluation metric.

Each of the optimal models will be trained and evaluated against each of the ten stratified folds of the dataset. The F1 score, precision, and recall metrics will be calculated for each iteration of each model. The normality for each result set will be determined. Dependent T-Tests or Wilcoxon signed-rank tests will be carried out to compare the results of all models. If the CNN models show a significant improvement over the SVM models with 95% confidence the null hypothesis will be rejected and otherwise it will be accepted.

3.8.1 Delimitation and scope

The main focus of this experiment was the evaluation and comparison of the effectiveness for SVM and CNN algorithms for the purposes of intent detection in the context of a natural language interface.

This experiment will be carried out using only the data available in the Maluuba Frames dataset (El Asri et al., 2017). A number of SVM and CNN models have been identified and optimised for the purposes of this experiment. Some selections of tools and methods were made as a consideration of resources available.

3.8.2 Strength and limitations of approach taken

The main strength of this experiment is its evaluation against a dataset which closely mimics the interactions of a true conversational user interface. All bar a single identified experiment utilised general NLP datasets for building and evaluating the machine learning models. The single identified experiment which employed a conversational dataset only evaluated a single model (SVM) with no comparison to other models.

In this experiment a number of variations of both SVM and CNN models are defined, optimised, and evaluated. For SVM, models based on four common kernels

(one linear and three non-linear) are evaluated. For each of these models a grid search is carried out in order to find the optimal parameter configuration, so that each model generates the best possible results in the context of this experiment.

In addition, a number of CNN architectures are defined based on the existing research, ranging from shallow models with single CN, PL, and FC layers, to deep models with multiple instances of each layer. A grid search is employed in order to identify the optimal combination of activation function, pooling type, and optimisation function. As with the SVM models, this helps to ensure that each of the models generates the maximum results for the experiment.

As this is an experiment in the NLP domain, the NLP processes and tools utilised are incredibly important to the overall results of the experiment, and can have a significant impact on the overall performance of the models. The NLP process utilised in this experiment has been created to achieve the optimal results based on the domain (travel) and problem (intent detection for conversational interfaces). The cutting-edge mechanisms for vectorisation (TF-IDF), and lexicon normalisation (lemmatisation) have been identified and applied as part of this process. In addition, specific steps for noise reduction like replacement of date, currency, and location information have been put in place.

There are a number of potential weaknesses in the experiment. Firstly, the imbalance present in the dataset may adversely impact the training and evaluation of the models. This imbalance may impact some models more strongly than others, resulting in incorrect results for some models. The experiment attempts to deal with this imbalance but further research would be required in order for the impact to be determined.

Secondly, the optimum parameters for each model were identified using two-fold stratified cross-validation instead of the ten-fold stratified cross-validation that the experiment ultimately utilises. Ideally, ten-fold stratified cross-validation would be employed in order to determine the optimum parameters for all the models. Two-fold validation was applied in order to reduce the time needed to execute the grid search so that they could be delivered in the time-frame needed in order to complete this thesis.

Thirdly, there are other potential configurations of both algorithms which could have been considered. For example, the SVM algorithm has many other kernels available, other than the ones employed in this experiment. The potential scope of CNN architectures is even greater due to the possible number of architectures and options which could be combined to build CNN models. A number of shallow and deep architectures have been defined but it is possible to create an almost unlimited number of combinations.

Chapter 4

Implementation and results

In this chapter, the dataset will be analysed and then pre-processing steps will be implemented in order to prepare the data for modelling. The gridsearch will be implemented for each model, and the results will define the particular models to be generated on the prepared dataset. Finally, the results of the optimised models will be provided.

4.1 Data Understanding

4.1.1 The Dataset

A Python script was written to extract and analyse the data from the JSON format provided by the dataset. The dataset consists of an array of top-level JSON entities described as a 'dialogue'. Each dialogue consists of an array of entities named 'turns', where each turn represents a statement from a user. Every turn contains a 'text' attribute as its first data point, and this attribute is the statement entered by the respective user. All turns also contain a 'labels' array. The majority of labels arrays contain an 'acts' array. Any turn which does not contain any acts will have their intent defined as 'NO_INTENT'. For all remaining records, the first element in the args array contains the intent within an element labelled as 'name'. The statement and intent are extracted from this JSON format into a data structure within the python script.

4.1.2 Statements and Intents

Individual statements in the dataset have been categorised into one of twenty different intents (classifiers) as seen in Figure 4.1. A 21st classifier 'NO_INTENT' has been added to represent the records with no classifier. The number of instances for each intent varies considerably. The most common is 'inform' which consists of 8,239 instances, while 'reject' has the least number of instances at 13. There are 138 of the records assigned to 'NO_INTENT'.

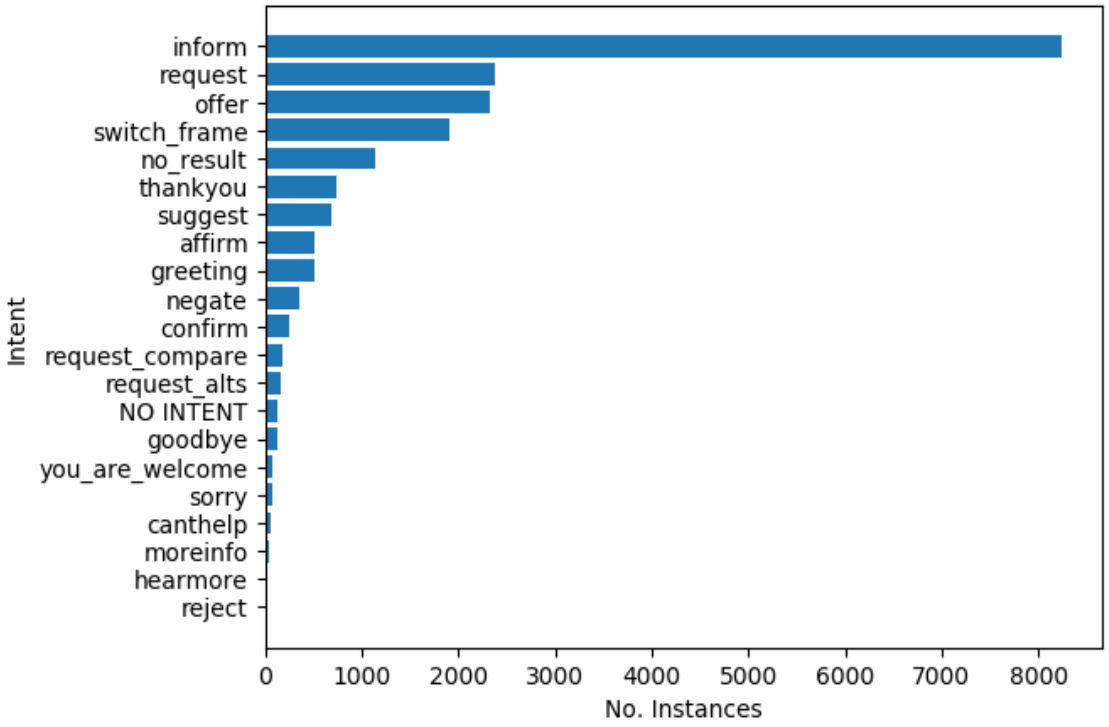

Figure 4.1: Illustration of the intents in the unprocessed dataset compared to the number of occurrences of that intent. A clear imbalance can be seen.

There are an average of 12.6 words per statement, with a median word count of nine. The longest statement contains 112 words, and the shortest containing just one single word. The distribution of word counts can be seen in Figure 4.2.

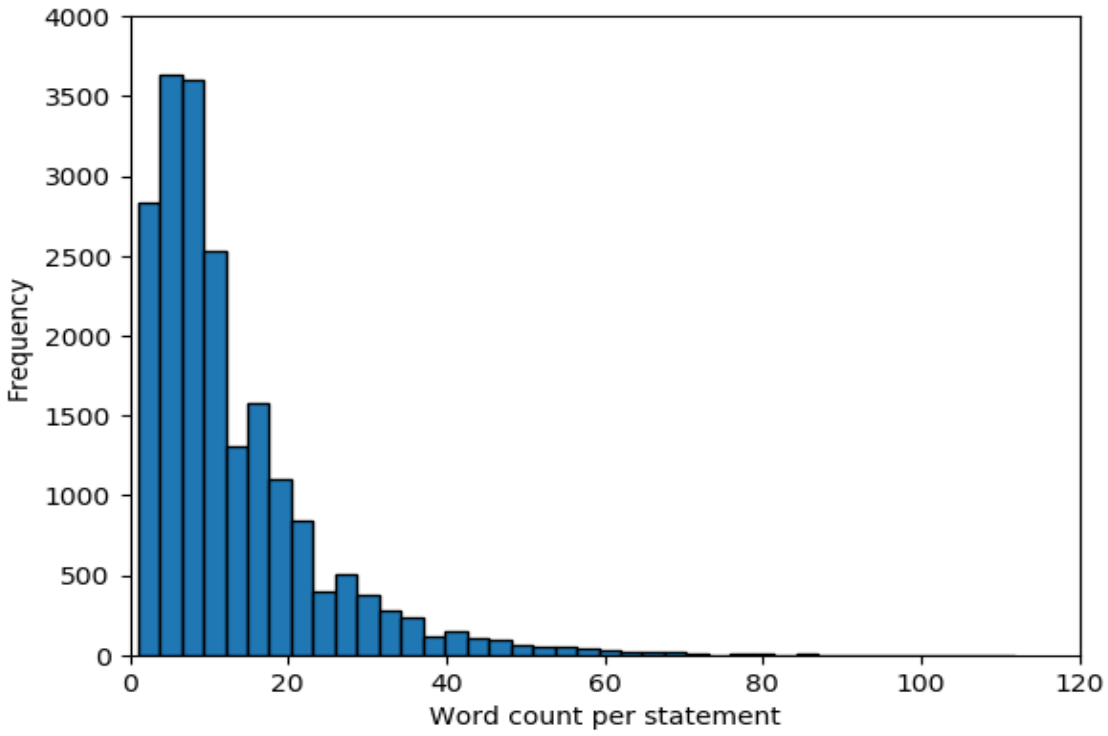

Figure 4.2: Frequency of word counts per statements in the unprocessed dataset. This illustrates that the statements tend to be short, with a large portion being made up of less than 20 words.

Of the statements in the dataset, 9,469 contain a full stop, 7,700 contain a question mark, and 2,613 contain an exclamation point. Also, there are several other punctuation symbols present. There are 428 instances of emoji characters occurring in the dataset. The dataset contains some words which occur very frequently, and some which appear quite rarely. Table A.1 lists the words which arise over 500 times in the dataset. Words such as 'the' occur very frequently and potentially increase noise in the dataset. However, there are some common words which may be extremely relevant to the categorisation process, for example, 'hotel' is the fourteenth most common word. As this dataset was built around booking flights and accommodation, this could be a significant word in the process of determining intent. There are 7,127 words which arise only a single time in the dataset. Similarly to words which commonly appear, infrequent words can increase noise in the data. These infrequent words are often; prices quoted for trips (in a variety of formats), slang, dates (in a range of formats),

and nouns (some real and some imaginary place names, e.g. Detroit, and Atlantis).

4.2 Data Preparation

4.2.1 Missing Data

A total of 138 records in the dataset have not been categorised with an intent and will be excluded from the data for this experiment. The removal of the records with no identified intent has no significant impact on the data. The graph of statements by the number of instances can be seen in Figure 4.3.

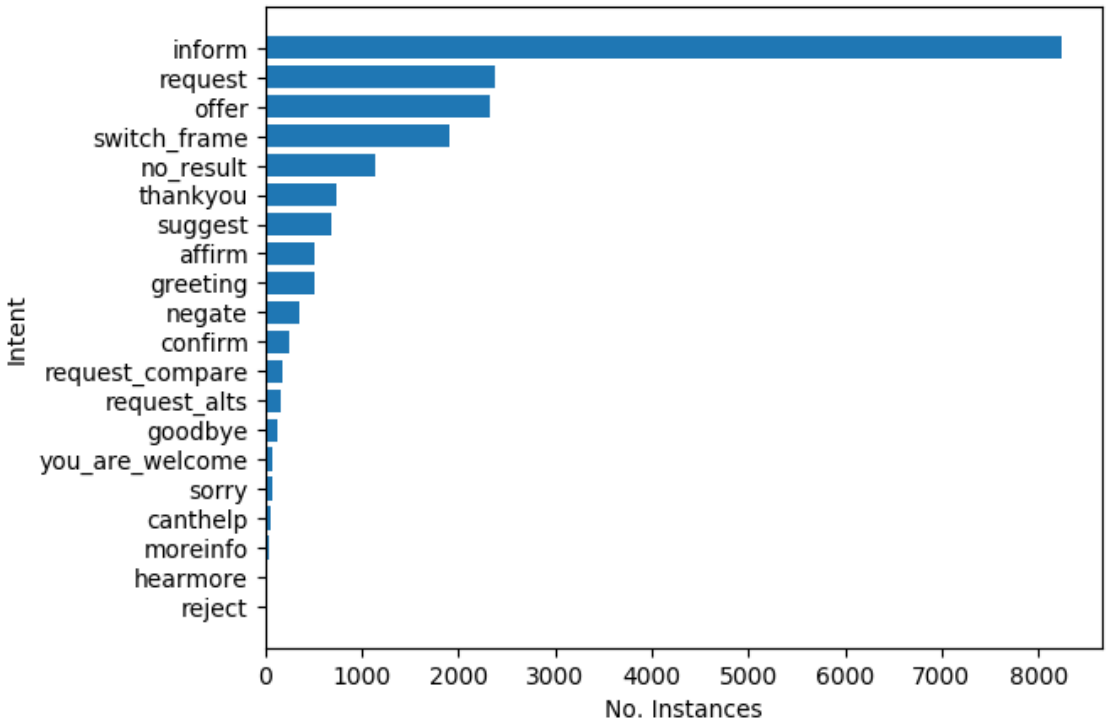

Figure 4.3: Illustration of the intents in the processed dataset compared to the number of occurrences of that intent. The no$_i$*ntentcategoryhasbeenremoved*.

4.2.2 Lexicon Normalisation

The python NLTK package will be employed to perform the normalisation process. The WordNetLemmatizer contained in this package will be executed against each word

in the corpus. If the algorithm returns a new value, this is the identified lemma of the specified word and will replace the original word in that statement.

4.2.3 Noise Reduction

The first phase of noise reduction will be carried out at a statement level for [DATETIME], [GPE], and [CURRENCY]. This means that each statement will be isolated and pattern matched for replacement of the pattern with the identified token. A Regex pattern string (see Table A.3) has been identified which handles the majority of date patterns that have been identified in the data. Every identified instance of currency will be replaced with the token: [DATETIME]. When GPE replacement is carried out it results in 7,502 instances of identified GPE's being replaced with [GPE]. The date time-related noise reduction results in the replacement of 4,692 instances of dates and times in the statements, with the [DATETIME]. Similarly, currency-related replacement results in 3,233 instances of currency being replaced by [CURRENCY].

The next phase of noise reduction will be carried out at the individual word level. Every statement will be split on space character in a process known as tokenisation. The individual tokens will be analysed for instances of emoji, punctuation, and such likes, where the relevant replacement will be carried out. Once all replacements are executed, each statement will be re-constituted by joining every non-empty token back together. Each of the 428 identified instances of Emoji characters in the dataset will also be removed. This will be achieved through a simple replacement of each identified instance with an empty string. Emoji tokens are defined as beginning and ending with the '::' characters. The following words will be removed from the dataset: the, i, you, a, and, , for, is, have, in, would, of, can, that, are, this, be, it, on, we, like, do, at, with, me, your, will, my, there, but, or, not, if, any, get, has, want, just, i'm, as, and also. All remaining common words are potentially relevant to this particular task. After completing the tidy up of the data, there is an average of 7.62 words per statement, with a median word count of 6. The longest statement contains 76 words and the shortest contain just a single word. The distribution of word counts can be seen in Figure 4.4.

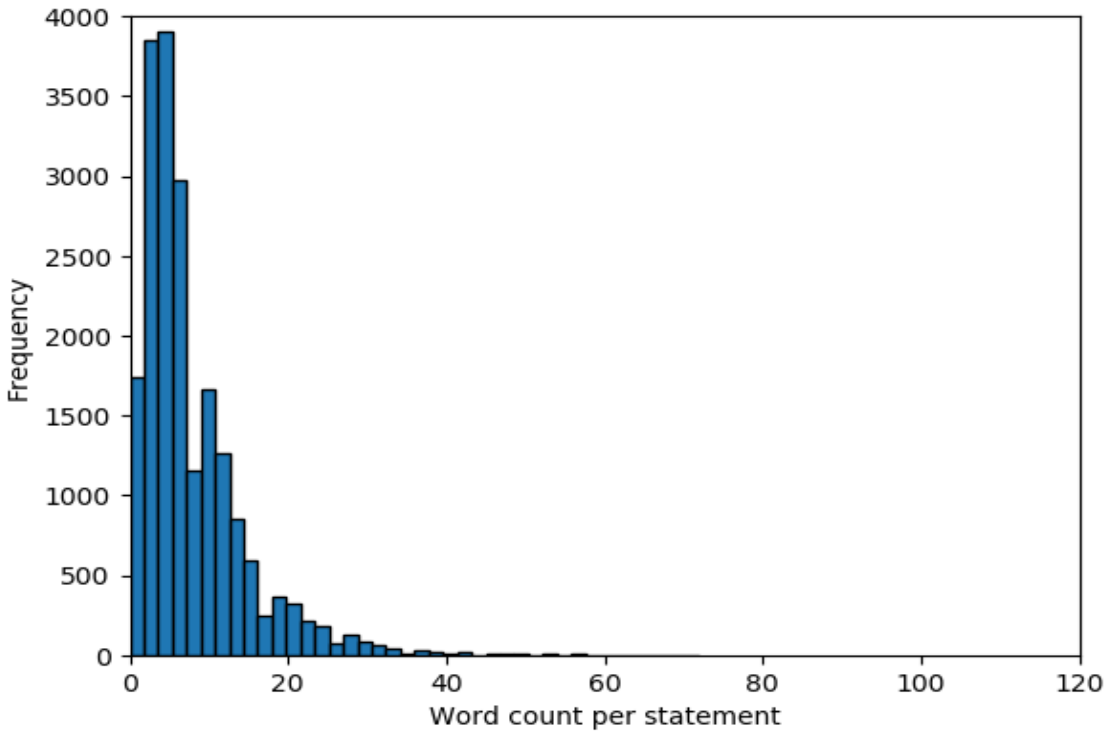

Figure 4.4: Frequency of word counts per statements in the processed dataset. When compared to Figure 4.2 we can see that there has been a shift to the left, indicating that the statements are now generally shorter after processing.

Outlined in the Table A.2, the 500 most frequently occurring words in the processed dataset can be seen. When comparing to the most frequent words in the original dataset as seen in Table A.1, it can be observed that there are now fewer commonly occurring words at a count of 51 to 78. The entity tokens: '[gpe]', '[datetime]', and '[currency]', are now among the most common words. The most frequently arising word is '[gpe]' with 7502 occurrences, compared with 'to' which appears 8,550 times in the original dataset. It can also be observed that many of the commonly occurring words in the processed dataset are travel related, i.e. hotel, star, package, trip, book, budget, rating, economy, guest, cost, parking, etc.

4.2.4 Feature Extraction

The scikit-learn package's TfidfVectorizer will be employed to convert each statement into its vector representation. This vectorisation mechanism has several defined parameters which can be specified. The following parameters will be applied, and all other parameters defined by the implementation will utilise the default values specified by the package. The sublinear_df parameter will be set to 'True' to employ logarithmic frequency in the transformation. The min_df will be set to '5' to ignore words which occur fewer than five times in the dataset. The norm will be set to 'I2' which causes all feature vectors to have a Euclidean norm of 1. The ngram_range will be set to '1,2' to employ unigrams and bigrams. Unigrams represent single words, and bigrams represent word pairs. By including word pairs, we encode that data as specific columns in the data which forces the pairs to be considered regarding significance and modelling. Trigrams were considered, but the increased size of the feature set resulted in memory problems on the available hardware. The stop_words will be set to 'None' as corpus specific stop words have already been identified and removed. The TfidfVectorizer generates a vector map with 8,174 columns.

4.2.5 Feature Selection

Executing a Chi-squared test (using scikit-learn) against the data indicates that 1128 have a p-value of ≤ 0.05. This indicates a high significance for those columns within the dataset. The remaining non-significant vectors will not be utilised for modelling and evaluation.

In addition to the Chi-squared test, a PCA is run to further reduce the number of vectors. The PCA is executed using scikit-learn. The PCA is first executed to calculate the components for all 8,174 columns with the svd_solver parameter set to 'full'. The explained variance for the PCA can be seen in Figure 4.5. This demonstrates that there is a diminished return with the transformed data at around 200 PCA components. The PCA is subsequently executed to restrict the transformation to 200 components, also using svd_solver set to 'full.

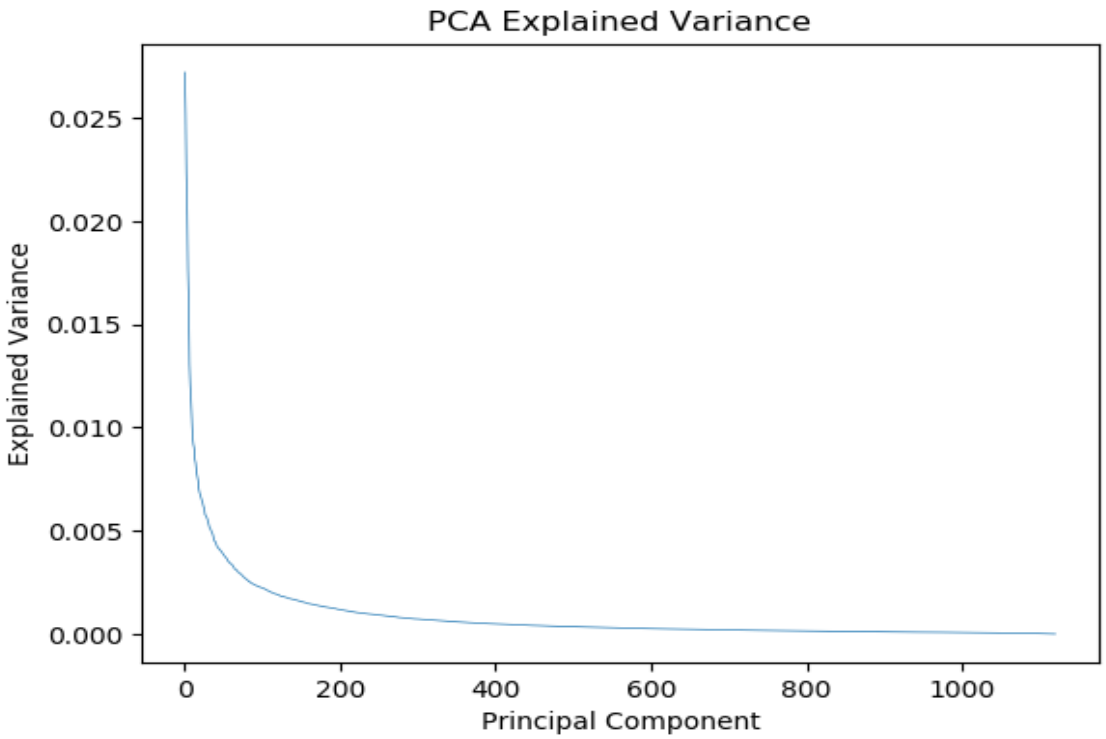

Figure 4.5: PCA Explained Variance of the dataset for dimensionality reduction. This illustrates that the impact of the vectors declines after approximately the first two-hundred.

4.2.6 Imbalanced Data

The compute_class_weight class in scikit-learn is utilised to determine the balanced class weights. The class_weight parameter will be set to 'balanced' to determine a balanced weight for each intent. The weights for each intent can be seen in Table A.4.

4.2.7 Training and Evaluation Data

The scikit-learn StratifiedKFold function is employed to split the data into ten folds of almost identical size and composition. This function is initialised with a random_state of '42' and shuffle will be set to True. All other parameters retain the default values specified by the scikit-learn package implementation.

4.3 Modelling

4.3.1 SVM

Within the scikit-learn packages model selection namespace there is a GridSearchCV method for identifying optimal parameter sets. GridSearchCV accepts a classifier, a cross-validation definition, a dictionary containing the parameters to execute, and numerous other parameters like the scoring mechanism to use and the number of jobs to execute in a multi-threaded environment.

In this instance, the classifier is an SVM definition created using the scikit-learn SVC implementation. Separate SVC definitions were created for each kernel ('linear', 'rbf', 'poly', and 'sigmoid') and decision function ('ovr' and 'ovo'). Decision functions are not a criterion that the GridSearchCV can accept in its dictionary of parameters to search. This leads to eight SVC definitions being employed as the classifier for the GridSearchCV function. Each SVC implementation was initialised with a random_state of '42' and a cache_size of '2,000' (based on available resources). The weighted intents defined for each class were supplied using the class_weight parameter, as represented in Table A.4 for the list of class weights .

The cross-validation instance, in this case, is the StratifiedKFold method from the scikit-learn model selection namespace. The n_splits parameter representing the number of splits to make in the data was set to '2'. the random_state parameter representing the initial random state for the instance was set to '42', and shuffle was set to True in order to shuffle the data before it was split.

An appropriate param_grid dictionary was constructed for each of the SVM kernel types. The linear model param_grid contained only the list of potential C values already outlined. The poly kernel param_grid contained the list of C, gamma, and degree values. The RBF and sigmoid kernels param_grids contained the C and gamma values.The other parameters set for the GridSearchCV instance are n_jobs, representing the threads to utilise for the modelling and evaluation. In this case, the n_jobs was set to '8' as the exercise was being carried out on an eight core machine. The scoring

parameter was set to 'f1_weighted', which represents using a weighted F1 score from each iteration as the scoring mechanism for identifying the best parameters.

Execution of the GridSearchCV hasidentified the best parameter combinations for each SVC instance, these are outlined in Table 4.1. For each kernel, the best criterion and score are identical for both the ovr and ovo decision functions. This indicates that there is no difference in impact for the decision function utilised in this problem. As such the ovr decision function will be employed to build the final SVM models for each kernel, as it is the default parameter utilised by the SVC model. This will result in four SVM models being generated for the final evaluation of SVM models for this problem and dataset. Each kernel will utilise the identified best criterion combination for C, gamma, and degree as specified in Table 4.1. The RBF kernel shows the most promising results from the grid search, followed by the polynomial, linear, and sigmoid.

Table 4.1: Optimum parameter configuration for each kernel from the grid search of the SVM algorithm. Ovr and ovo results are identical for each kernel.

Kernel	Decision Function	C	Gamma	Degree	Score
RBF	ovo	10	4	N/A	0.69
RBF	ovr	10	4	N/A	0.69
Poly	ovo	1000	3	1	0.67
Poly	ovr	1000	3	1	0.67
Linear	ovo	1000	N/A	N/A	0.64
Linear	ovr	1000	N/A	N/A	0.64
Sigmoid	ovo	1000	0.01	N/A	0.63
Sigmoid	ovr	1000	0.01	N/A	0.63

4.3.2 CNN

Each of the identified 700 models is built as a sequential model in the keras package by iterating through all possible parameter combinations. As the input for each model is the output from the PCA, which is a single dimensional vector, 1D convolutional

and pooling steps will be employed for all models. Each architectural model will be built by adding the appropriate convolutional, pooling, and fully connected layers to each model. Each layer will utilise the appropriate activation for that model, with the exception of the final fully connected layer for each model which employs a softmax activation and a fixed size of 20 units. The model is compiled using the defined loss ('categorical_crossentropy'), optimizer (dependent on model instance), and metrics ('accuracy') for that iteration.

The first shallow architecture termed 'Shallow_1' is a sequential model, with a Conv1D as its first layer. This layer contains 100 filters, with a kernel_size of 1, padding is 'valid', a stride of 1, and an input_shape of (None, 200). The activation parameter will be the appropriate activation for this model instance. This layer is followed by a GlobalMaxPooling1D or GlobalAveragePooling1D layer. Pooling type is dependent on the configuration of this model instance.

Termed 'Shallow_2', the second shallow architecture is a sequential model, with a Conv1D as its first layer. This layer contains 100 filters, with a kernel_size of 1, padding is 'valid', a stride of 1, and an input_shape of (None, 200). The activation parameter will be the appropriate activation for this model instance. This layer is followed by a GlobalMaxPooling1D or GlobalAveragePooling1D layer. Pooling type is dependent on the configuration of this model instance. The pooling layer is followed by a fully connected layer with 100 units and the appropriate activation for the instance.

A third shallow architecture termed 'Shallow_3' is another sequential model, with a Conv1D as its first layer. This layer contains 150 filters, with a kernel_size of 1, padding is 'valid', a stride of 1, and an input_shape of (None, 200). The activation parameter will be the appropriate activation for this model instance. This layer is followed by a GlobalMaxPooling1D or GlobalAveragePooling1D layer. Pooling type is dependent on the configuration of this model instance. The pooling layer is followed by a fully connected layer with 150 units and the appropriate activation for the instance.

The first deep architecture termed 'Deep_1' is a sequential model, with a Conv1D as its first layer. This layer contains 100 filters, with a kernel_size of 1, padding is 'valid', a stride of 1, and an input_shape of (None, 200). The activation parameter will be

the appropriate activation for this model instance. Pooling type is dependent on the configuration of this model instance. This layer will be followed by a MaxPooling1D or AveragePooling1D layer. Each combined Conv1D and pooling layer will be repeated until the model contains three of the layers. These three pairs of layers are followed by a GlobalMaxPooling1D or GlobalAveragePooling1D layer as appropriate for the instance. The pooling layer is followed by a fully connected layer with 100 units and the appropriate activation for the instance.

A second deep architecture termed 'Deep_2' is a sequential model, with a Conv1D as its first layer. This layer contains 100 filters, with a kernel_size of 1, padding is 'valid', a stride of 1, and an input_shape of (None, 200). The activation parameter will be the appropriate activation for this model instance. Pooling type is dependent on the configuration of this model instance. This layer will be followed by a MaxPooling1D or AveragePooling1D layer. Every combined Conv1D and pooling layer will be repeated until the model contains six of the layers. These six pairs of layers are followed by a GlobalMaxPooling1D or GlobalAveragePooling1D layer as appropriate for the instance. The pooling layer is followed by two fully connected layers with 100 units each and the appropriate activation for the instance.

In order to train and evaluate each instance, the scikit-learn StratifiedKFold class is employed to generate two stratified splits of the data. To achieve this, the n_splits property is set to '2', the random_state is set to '42' and shuffle is set to 'True'. This results in two folds, each consisting of training and evaluation data.

Each fold is iterated and some transformations are carried out in order to convert the data in the format expected by keras. A scikit-learn LabelEncoder class is applied to encode the text-based intents as integer numbers. The class weights determined for the imbalanced data are also encoded using this mechanism. In addition, keras expects the input in the format of a tensor. The numpy reshape method is employed to alter the shape of the input values for both the training and test data. In addition, the categorical_crossentropy loss function expects one hot encoded values so the intents for the training data are encoded using keras to_categorical for num_classes set to '20'. The intents for the test data are not encoded.

The fit method is called for each model where the training and test data for the fold are utilised. A batch_size of 100 is employed. Each fit has its epochs parameter set to 5,000 but an EarlyStopping callback is defined to terminate the execution early. The EarlyStopping callback is defined with the monitor param set to 'loss', patience set to '10', and min_delta set to '0.005'. The class_weight is set to the encoded class weights determined previously.When the fit is completed for each model the predict method will be called with the test data supplied. This returns the predicted intent generated by the model for each of the test cases. The scikit-learn f1_score method is applied to determine the score. It accepts the predicted intents, along with the correct intents, and an average parameter set to 'weighted' in order to account for the imbalanced multiple classes. This score is recorded for that model instance. Once all model instances have been evaluated against both folds, the scores for each instance are averaged and recorded. The results for the highest scoring implementation of each model can be seen in Table 4.2

Table 4.2: Optimum parameter configurations for each of the CNN architectures, from the results of the grid search.

Name	Activation	Pooling Type	Optimizer	Score
Shallow_1	softsign	max	nadam	0.69
Shallow_2	ReLU	max	adam	0.68
Shallow_3	ReLU	max	nadam	0.70
Deep_1	ReLU	avg	adadelta	0.69
Deep_2	selu	avg	adagrad	0.66

The results above agree somewhat with the existing literature reviewed. Y. Zhang and Wallace (2017) indicated that ReLU activation was amongst the most effective for the shallow models evaluated in the paper. In this case, the GridSearch identified ReLU as the optimum activation for three of the architectures, two of which are shallow. Two remaining patterns utilise softsign or selu, which were not identified as the best performing but the activation function may be dependent on the dataset under

evaluation and the other elements of the model. For example, the Shallow_1 model consists of a CN and an output layer, but the other shallow models both feature an FC layer in between. It could be that the inclusion of this FC layer impacts the model so that ReLU generates the best performance. Nii et al. (2017) also utilised ReLU activation for a shallow model, but no comparison was made with other activation functions.

Max pooling proved to be the most effective pooling technique for shallow architectures, whereas average pooling proved most effective for the two deep architectures. Y. Zhang and Wallace (2017) identified Max pooling as the most effective mechanism for shallow architectures. Nii et al. (2017) employ max pooling for its shallow architecture. No identified papers utilised Average pooling, but no deep architectures were established.

4.4 Results

4.4.1 Precision

The generated results for the precision metric for each model are shown in Table 4.3. The SVM model with an RBF kernel demonstrates the highest precision score for 80% of the folds, while the SVM model with a Sigmoid kernel demonstrates the highest precision score for 20% of the folds.

A box plot showing the range of comparable results for each model can be seen in Figure 4.6. This illustrates that the SVM model using the RBF kernel attained the highest overall precision scores. A probability density plot of the values is also shown in Figure 4.7, from this it can be seen that the RBF results appear to be improved but the difference compared to the sigmoid kernel is not very pronounced.

Table 4.3: Weighted precision metric results for each fold of every SVM and CNN model. The highest precision score for each fold is highlighted in bold.

Fold	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	Linear	RBF	Poly	Sigmoid
1	0.693	0.686	0.696	0.683	0.699	0.689	**0.722**	0.689	0.713
2	0.691	0.693	0.683	0.655	0.704	0.686	**0.723**	0.686	0.705
3	0.690	0.692	0.685	0.669	0.679	0.678	**0.701**	0.678	0.684
4	0.684	0.680	0.684	0.670	0.693	0.700	**0.725**	0.698	0.715
5	0.675	0.680	0.683	0.682	0.676	0.685	**0.711**	0.685	0.691
6	0.707	0.691	0.691	0.699	0.694	0.685	**0.727**	0.686	0.708
7	0.686	0.692	0.674	0.681	0.700	0.687	**0.718**	0.684	0.714
8	0.682	0.687	0.686	0.674	0.691	0.686	0.703	0.686	**0.707**
9	0.680	0.672	0.686	0.673	0.696	0.687	0.709	0.688	**0.710**
10	0.696	0.687	0.692	0.690	0.705	0.696	**0.722**	0.696	0.718

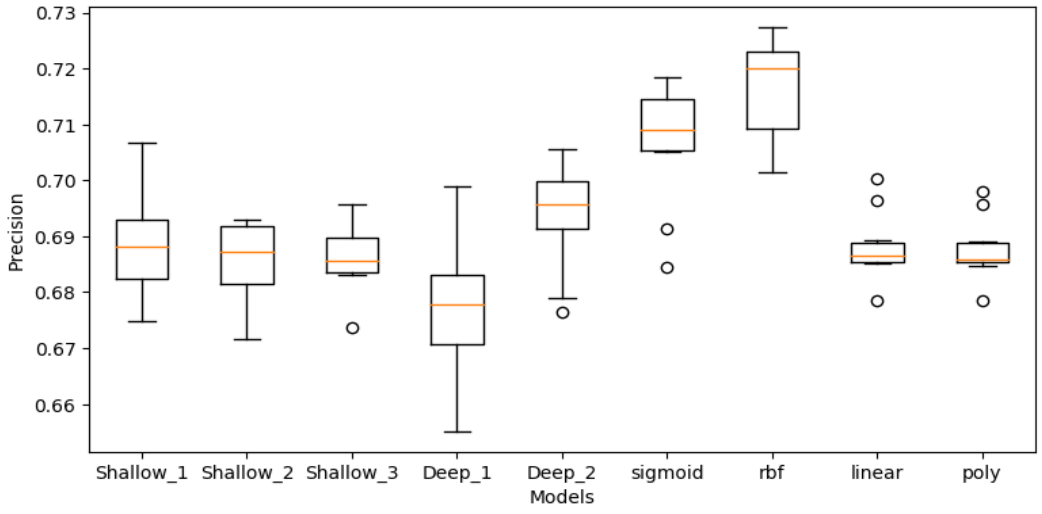

Figure 4.6: Boxplot representation of the weighted precision metric for each SVM and CNN model

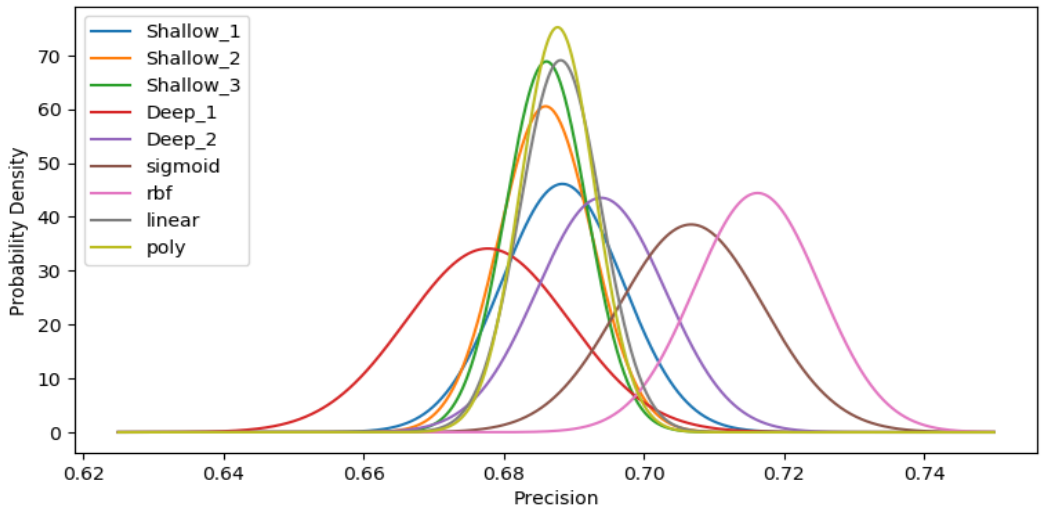

Figure 4.7: Probability Density of the weighted precision metric results for each SVM and CNN model

4.4.2 Recall

The generated results for the precision metric for each model are shown in Table 4.4. The SVM model with an RBF kernel demonstrates the highest recall score for all of the folds.

Table 4.4: Weighted recall metric results for each fold of every SVM and CNN model. The highest recall score for each fold is highlighted in bold.

Fold	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	Linear	RBF	Poly	Sigmoid
1	0.670	0.654	0.677	0.659	0.633	0.645	**0.718**	0.649	0.636
2	0.664	0.652	0.662	0.632	0.633	0.643	**0.713**	0.646	0.627
3	0.658	0.670	0.654	0.639	0.634	0.627	**0.692**	0.627	0.600
4	0.654	0.656	0.657	0.630	0.623	0.655	**0.713**	0.655	0.630
5	0.646	0.623	0.665	0.647	0.642	0.637	**0.701**	0.641	0.613
6	0.669	0.650	0.672	0.675	0.642	0.639	**0.717**	0.645	0.628
7	0.655	0.667	0.661	0.639	0.620	0.640	**0.709**	0.640	0.629
8	0.652	0.656	0.674	0.646	0.658	0.635	**0.692**	0.638	0.621
9	0.647	0.634	0.634	0.634	0.634	0.640	**0.697**	0.688	0.627
10	0.668	0.656	0.667	0.668	0.657	0.696	**0.710**	0.651	0.640

A box plot showing the range of comparable results for each model can be seen in Figure 4.8. The SVM model with an RBF kernel demonstrates the highest recall score for all of the folds. A probability density plot of the values is also shown in Figure 4.9, this clearly illustrates that the RBF model results are an improvement over the recall results from other models.

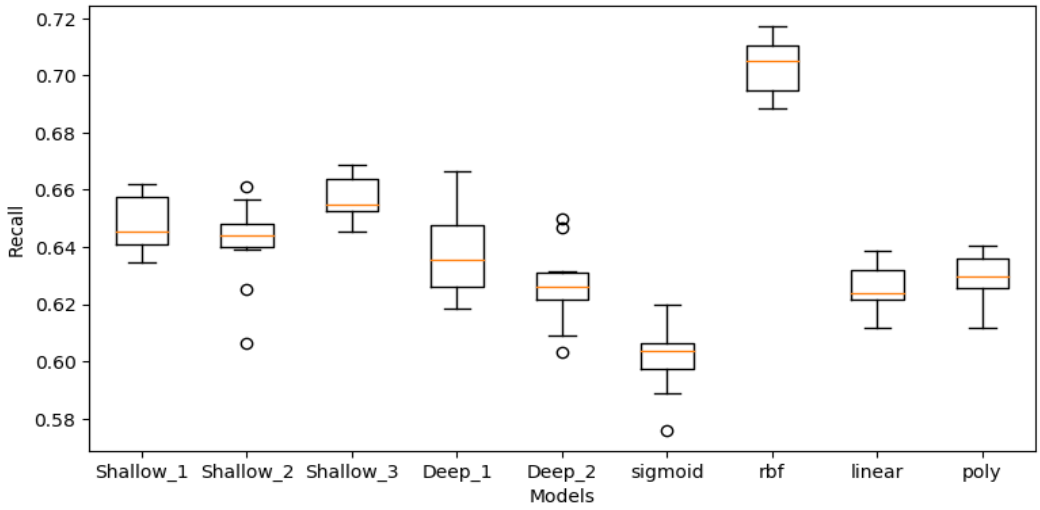

Figure 4.8: Boxplot representation of the weighted recall metric for each SVM and CNN model

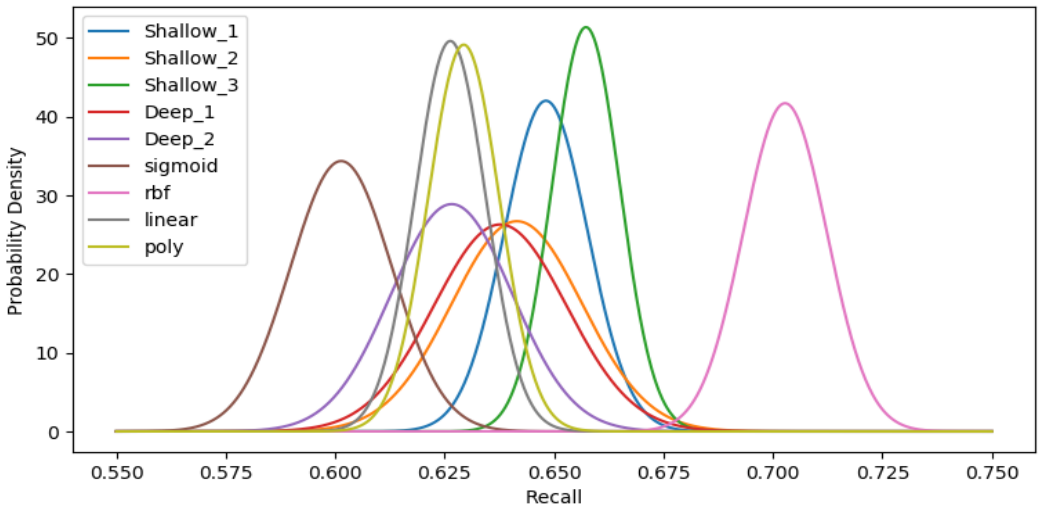

Figure 4.9: Probability Density of the weighted recall metric results for each SVM and CNN model

4.4.3 F1 Scores

The generated results for the F1 score metric for each model are shown in Table 4.5. This illustrates that the SVM model using the RBF kernel attained the highest overall recall scores.

Table 4.5: Weighted f1 metric results for each fold of every SVM and CNN model. The highest f1 score for each fold is highlighted in bold.

Fold	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	Linear	RBF	Poly	Sigmoid
1	0.662	0.646	0.664	0.650	0.623	0.632	**0.717**	0.637	0.612
2	0.658	0.642	0.653	0.625	0.621	0.629	**0.711**	0.633	0.607
3	0.646	0.661	0.645	0.628	0.623	0.611	**0.688**	0.611	0.576
4	0.645	0.648	0.648	0.618	0.609	0.639	**0.707**	0.641	0.605
5	0.634	0.606	0.653	0.633	0.631	0.621	**0.697**	0.626	0.589
6	0.655	0.639	0.664	0.666	0.630	0.624	**0.712**	0.632	0.602
7	0.643	0.657	0.654	0.621	0.603	0.623	**0.705**	0.623	0.603
8	0.640	0.648	0.668	0.637	0.650	0.621	**0.689**	0.626	0.595
9	0.637	0.625	0.669	0.639	0.628	0.624	**0.694**	0.627	0.604
10	0.660	0.643	0.655	0.657	0.647	0.638	**0.706**	0.639	0.620

A box plot showing the range of comparable results for each model can be seen in Figure 4.10. This illustrates that the SVM model using the RBF kernel attained the highest overall F1 scores. A probability density plot of the values is also shown in Figure 4.11, this clearly illustrates that the RBF model results are an improvement over the F1 results from other models.

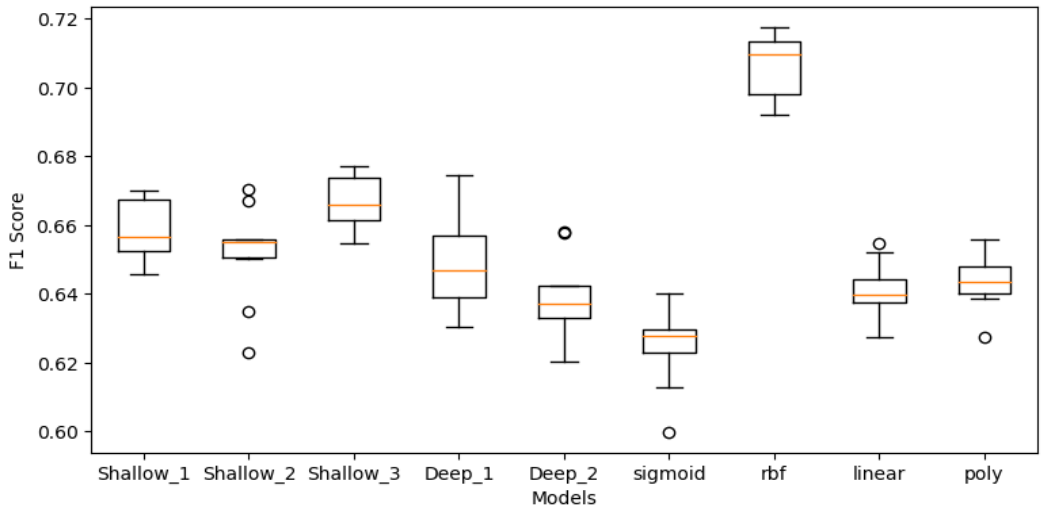

Figure 4.10: Boxplot representation of the weighted f1 metric for each SVM and CNN model

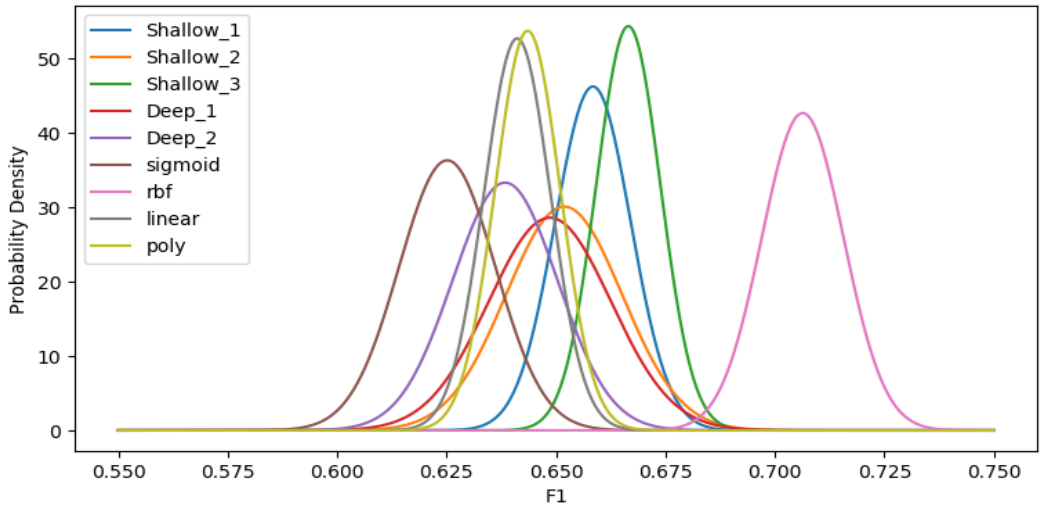

Figure 4.11: Probability Density of the weighted f1 metric results for each SVM and CNN model

Chapter 5

Evaluation and analysis

In this chapter an evaluation will be carried out based on the results from the previous chapter. This evaluation takes the form of a detailed statistical analysis of the result data in order to identify the optimal models for each metric. The results of this evaluation will then be analysed in order to suggest why the experiment generated these results. Finally the hypothesis H1 will be accepted or rejected based on the evaluation results.

5.1 Evaluation

5.1.1 Normality Tests

A Shapiro-Wilk test was run against each set of results for each set of scores to determine if it was normally distributed. This test was implemented using the SciPy packages Shapiro-Wilk test implementation. The null hypothesis for the Shapiro-Wilk test is that the data is normally distributed. A p-value of $<0.05\%$ results in the null hypothesis being rejected.

Precision Metric Normality Test Results

The results of the normality tests for the precision metric can be seen in Table 5.1. The null hypothesis is accepted for each of the models, indicating that they are normally distributed.

Table 5.1: Shapiro-Wilk normality test results for the precision metric

Model	W	p-val	Null-Hypothesis
Shallow_1	0.974	0.929	Accepted
Shallow_2	0.889	0.167	Accepted
Shallow_3	0.937	0.520	Accepted
Deep_1	0.979	0.960	Accepted
Deep_2	0.905	0.250	Accepted
sigmoid	0.866	0.091	Accepted
rbf	0.899	0.214	Accepted
linear	0.882	0.137	Accepted
poly	0.899	0.215	Accepted

Recall Metric Normality Test Results

The results of the normality tests for the recall metric can be seen in Table 5.2. The null hypothesis is accepted for each of the models, indicating that they are normally distributed.

Table 5.2: Shapiro-Wilk normality test results for the recall metric

Model	W	p-val	Null-Hypothesis
Shallow_1	0.921	0.366	Accepted
Shallow_2	0.892	0.177	Accepted
Shallow_3	0.927	0.417	Accepted
Deep_1	0.939	0.540	Accepted
Deep_2	0.950	0.664	Accepted
sigmoid	0.938	0.530	Accepted
rbf	0.937	0.518	Accepted
linear	0.939	0.546	Accepted
poly	0.944	0.599	Accepted

F1 Metric Normality Test Results

The results of the normality tests for the F1 metric can be seen in Table 5.3. The null hypothesis is accepted for each of the models, indicating that they are normally distributed.

Table 5.3: Shapiro-Wilk normality test results for the f1 metric

Model	W	p-val	Null-Hypothesis
Shallow_1	0.912	0.297	Accepted
Shallow_2	0.893	0.184	Accepted
Shallow_3	0.948	0.648	Accepted
Deep_1	0.939	0.538	Accepted
Deep_2	0.924	0.389	Accepted
sigmoid	0.896	0.200	Accepted
rbf	0.883	0.142	Accepted
linear	0.963	0.818	Accepted
poly	0.961	0.801	Accepted

5.1.2 Statistical Analysis

As the Shapiro-Wilk normality test indicates that all results are normally distributed, dependent T-Tests were executed for the model results. The test was run to compare the three metric results for each model to the same metric generated by every other model.

Analysis of Precision Results

The significance results for the precision metric can be seen in Table 5.4. Models with a significance of <0.05 are considered to be significant at the confidence level of 95%. There are significant differences indicated between many of the results. However, there is no significant difference indicated between; Shallow_1 and linear, Shallow_1 and poly, Shallow_2 and linear, Shallow_2 and poly, Shallow_3 and linear, Shallow_3 and poly, Deep_2 and linear. This indicates that the CNN Shallow_1, Shallow_2, Shallow_3, and

Deep_2 models do not offer significantly improved results when compared to some of the SVM models for the precision metric.

Table 5.4: Matrix of the Dependent T-Test P-Vals for the precision metric for each model (DF=9). Significant results (<0.05%) are highlighted in bold.

Model	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	sigmoid	rbf	linear	poly
Shallow_1	N/A	0.338	0.395	**0.016**	0.116	**0.001**	**<.001**	0.944	0.843
Shallow_2	0.338	N/A	0.97	0.096	**0.047**	**<.001**	**<.001**	0.533	0.620
Shallow_3	0.395	0.97	N/A	**0.044**	**0.046**	**<.001**	**<.001**	0.439	0.488
Deep_1	**0.016**	0.096	**0.044**	N/A	**0.009**	**<.001**	**<.001**	**0.032**	**0.035**
Deep_2	0.116	**0.047**	**0.046**	**0.009**	N/A	**<.001**	**<.001**	0.059	**0.044**
sigmoid	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	N/A	**0.007**	**<.001**	**<.001**
rbf	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.007**	N/A	**<.001**	**<.001**
linear	0.944	0.533	0.439	**0.032**	0.059	**<.001**	**<.001**	N/A	0.328
poly	0.843	0.620	0.488	**0.035**	**0.044**	**<.001**	**<.001**	0.328	N/A

The t-statistic values for the Dependent T-Test of the precision metric can be seen in Table 5.5. The Dependent T-Tests are carried out in pairs, a negative number indicates that the results for the first model in the comparison are not as good as the results for the second model. Only the SVM RBF model results indicate that its precision results are significantly better than the precision results of both the CNN models and the other SVM models. The performance is also represented in Figure 5.1 where it can be seen that the RBF kernel generates better results than the other models.

When comparing the CNN model results to those of the SVM models, it is evident that the Deep_2 model generated significantly better results than the SVM poly model for the precision statistic. No other CNN model generated significantly better results than the SVM models.

The CNN model comparisons indicate that Shallow_1 generated significantly better results than the Deep_1 model. The Shallow_3 model also generated significantly better results than the Deep_1 model. The Deep_2 model generated significantly better results than the Shallow_2, Shallow_3, and Deep_1 models, making it the most effective CNN

model in relation to the precision statistic. A ranking of the models based on the precision statistic results can be seen in Table 5.6. This ranking is based on the number of models where the model being evaluated generates a statistically improved result.

Table 5.5: Matrix of the Dependent T-Test T-statistic for the precision metric for each model (DF=9). Significant results (p-val <0.05%) are highlighted in bold. A result of >0 indicates that a model generated better results, with results of <0 indicating poorer results. Reading from the left take the left value, reading from the top take the right value.

Model	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	sigmoid	rbf	linear	poly
Shallow_1	N/A	1.01/-1.01	0.89/-0.89	**2.93/-2.93**	-1.74/1.74	**-4.73/4.73**	**-10.12/10.12**	0.07/-0.07	0.20/-0.20
Shallow_2	-1.01/1.01	N/A	-0.03/0.03	1.86/-1.86	**-2.30/2.30**	**-4.85/4.85**	**-8.93/8.93**	-0.64/0.64	-0.51/0.51
Shallow_3	-0.89/0.89	0.03/-0.03	N/A	**2.33/-2.33**	**-2.31/2.31**	**-5.59/5.59**	**-9.34/9.34**	-0.80/0.80	-0.72/0.72
Deep_1	**-2.93/2.93**	-1.86/1.86	**-2.33/2.33**	N/A	**-3.34/3.34**	**-6.56/6.56**	**-9.28/9.28**	**-2.52/2.52**	**-2.47/2.47**
Deep_2	1.74/-1.74	**2.30/-2.30**	**2.31/-2.31**	**3.34/-3.34**	N/A	**-7.05/7.05**	**-8.41/8.41**	2.15/-2.15	**2.33/-2.33**
sigmoid	**4.73/-4.73**	**4.85/-4.85**	**5.59/-5.59**	**6.56/-6.56**	**7.05/-7.05**	N/A	**-3.42/3.42**	**7.98/-7.98**	**7.87/-7.87**
rbf	**10.12/-10.12**	**8.93/-8.93**	**9.34/-9.34**	**9.28/-9.28**	**8.41/-8.41**	**3.42/-3.42**	N/A	**11.83/-11.83**	**11.74/-11.74**
linear	-0.07/0.07	0.64/-0.64	0.80/-0.80	**2.52/-2.52**	-2.15/2.15	**-7.98/7.98**	**-11.83/11.83**	N/A	1.03/-1.03
poly	-0.20/0.20	0.51/-0.51	0.72/-0.72	**2.47/-2.47**	**-2.33/2.33**	**-7.87/7.87**	**-11.74/11.74**	-1.03/1.03	N/A

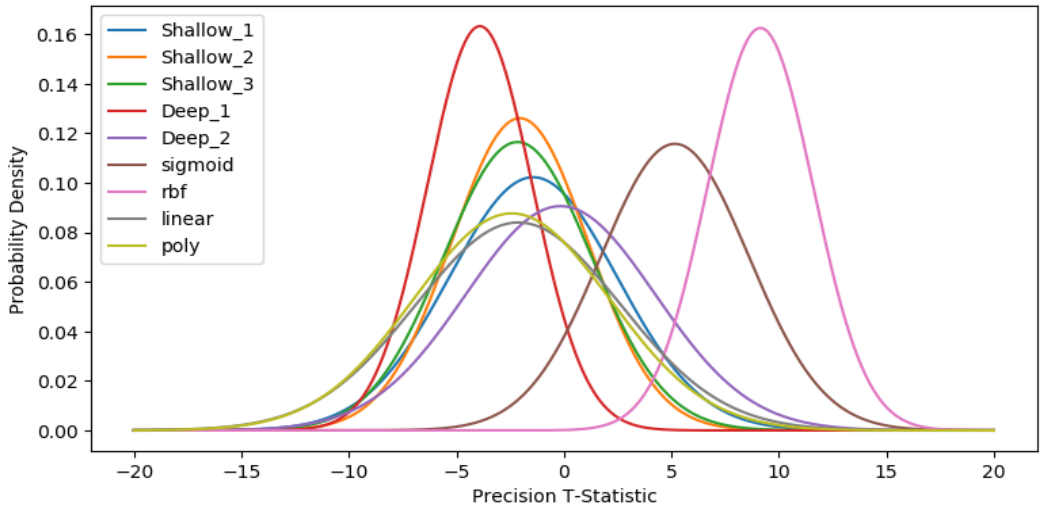

Figure 5.1: Probability density of Dependent T-Test T-statistics for the precision metric

Table 5.6: Ranking of models by significantly improved (p-val <0.05) precision results as indicated by the Dependent T-Test

Rank	Model	Type	No. Sig. Worse Models
1	rbf	SVM	8
2	sigmoid	SVM	7
3	Deep_2	CNN	4
4	Shallow_3	CNN	1
4	Shallow_1	CNN	1
4	poly	SVM	1
4	linear	SVM	1
5	Deep_1	CNN	0
5	Shallow_2	CNN	0

Analysis of Recall Results

The significance results for the recall metric can be seen in Table 5.7. Models with a significance of <0.05 are considered to be significant at the confidence level of 95%. There are significant differences indicated between many of the results. However, there is no significant difference indicated between; Shallow_2 and poly, Deep_1 and poly, Deep_1 and linear, Deep_2 and poly, Deep_2 and linear. This indicates that the CNN Shallow_2, Deep_1, and Deep_2 models do not offer significantly improved results when compared to some of the SVM models for the precision metric.

When the precision results are compared with 5.1 it does not appear that the RBF model is significantly better than the sigmoid models, as the statistical tests indicate. The Shapiro-Wilks test for normality has a 5% chance of indicating that the data is normal, even if it is not (this is regardless of the sample size). If some of the data is not normally distributed, it may result in an invalid result for the dependent t-test which assumes normality of the supplied data. In this case, two SVM models are of concern, so the ultimate results of this experiment are not significantly impacted.

Table 5.7: Matrix of the Dependent T-Test P-Vals for the recall metric for each model (DF=9). Significant results (<0.05%) are highlighted in bold.

Model	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	sigmoid	rbf	linear	poly
Shallow_1	N/A	0.198	0.053	**0.043**	**0.003**	**0.001**	**<.001**	**<.001**	**<.001**
Shallow_2	0.198	N/A	**0.031**	0.636	0.085	**<.001**	**<.001**	**0.026**	0.076
Shallow_3	0.053	**0.031**	N/A	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
Deep_1	**0.043**	0.636	**0.001**	N/A	**0.03**	**<.001**	**<.001**	0.06	0.128
Deep_2	**0.003**	0.085	**<.001**	**0.03**	N/A	**0.002**	**<.001**	0.954	0.614
sigmoid	**0.001**	**<.001**	**<.001**	**<.001**	**0.002**	N/A	**<.001**	**<.001**	**<.001**
rbf	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	N/A	**<.001**	**<.001**
linear	**<.001**	**0.026**	**<.001**	0.06	0.954	**<.001**	**<.001**	N/A	**0.009**
poly	**<.001**	0.076	**<.001**	0.128	0.614	**<.001**	**<.001**	**0.009**	N/A

The t-statistic values for the Dependent T-Test of the recall metric can be seen in Table 5.8. The Dependent T-Tests are carried out in pairs, a negative number indicates that the results for the first model in the comparison are not as good as the results for the second model. Only the SVM RBF model results indicate that its precision results are significantly better than the precision results of both the CNN models and the other SVM models. The performance is also represented in Figure 5.2 where it can be seen that the RBF kernel generates better results than the other models.

When comparing the CNN model results to those of the SVM models, it is evident that the Shallow_3 model generated significantly better results than all other models except Shallow_1 and the SVM RBF model.

The CNN model comparisons indicate that Shallow_1 generated significantly better results than the Deep_1 and Deep_2 models. The Shallow_3 model also generated significantly better results than the Shallow_2, Deep_1, and Deep_2 models. The Deep_2 model generated significantly better results than the Deep_1 model. The results indicate that the Shallow_3 model generated the most significantly better results of the CNN models. A ranking of the models based on the recall statistic results can be seen in Table 5.9. This ranking is based on the number of models where the model being

evaluated generates a statistically improved result.

Table 5.8: Matrix of the Dependent T-Test T-statistic for the recall metric for each model (DF=9). Significant results (p-val <0.05%) are highlighted in bold. A result of >0 indicates that a model generated better results, with results of <0 indicating poorer results. Reading from the left take the left value, reading from the top take the right value.

Model	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	sigmoid	rbf	linear	poly
Shallow_1	N/A	1.39/-1.39	-2.23/2.23	**2.34/-2.34**	**3.98/-3.98**	**14.23/-14.23**	**-23.97/23.97**	**7.52/-7.52**	**6.33/-6.33**
Shallow_2	-1.39/1.39	N/A	**-2.56/2.56**	0.49/-0.49	1.93/-1.93	**6.31/-6.31**	**-10.54/10.54**	**2.66/-2.66**	1.99/-1.99
Shallow_3	2.23/-2.23	**2.56/-2.56**	N/A	**4.70/-4.70**	**7.52/-7.52**	**14.65/-14.65**	**-11.22/11.22**	**8.34/-8.34**	**8.35/-8.35**
Deep_1	**-2.34/2.34**	-0.49/0.49	**-4.70/4.70**	N/A	**2.56/-2.56**	**7.18/-7.18**	**-12.80/12.80**	**2.14/-2.14**	**1.67/-1.67**
Deep_2	**-3.98/3.98**	-1.93/1.93	**-7.52/7.52**	**-2.56/2.56**	N/A	**4.35/-4.35**	**-11.89/11.89**	0.05/-0.05	-0.52/0.52
sigmoid	**-14.23/14.23**	**-6.31/6.31**	**-14.65/14.65**	**-7.18/7.18**	**-4.35/4.35**	N/A	**-35.77/35.77**	**-12.12/12.12**	**-12.71/12.71**
rbf	**23.97/-23.97**	**10.54/-10.54**	**11.22/11.22**	**12.80/-12.80**	**11.89/-11.89**	**35.77/-35.77**	N/A	**31.6/-31.6**	**33.99/-33.99**
linear	**-7.52/7.52**	**-2.66/2.66**	**-8.34/8.34**	-2.14/2.14	-0.05/0.05	**12.12/-12.12**	**-31.66/31.66**	N/A	**3.29/-3.29**
poly	**-6.33/6.33**	-1.99/1.99	**-8.35/8.35**	-1.67/1.67	0.52/-0.52	**12.71/-12.71**	**-33.99/33.99**	**-3.29/3.29**	N/A

Reading from left take the left number, reading from top take the right number - Significant results in bold

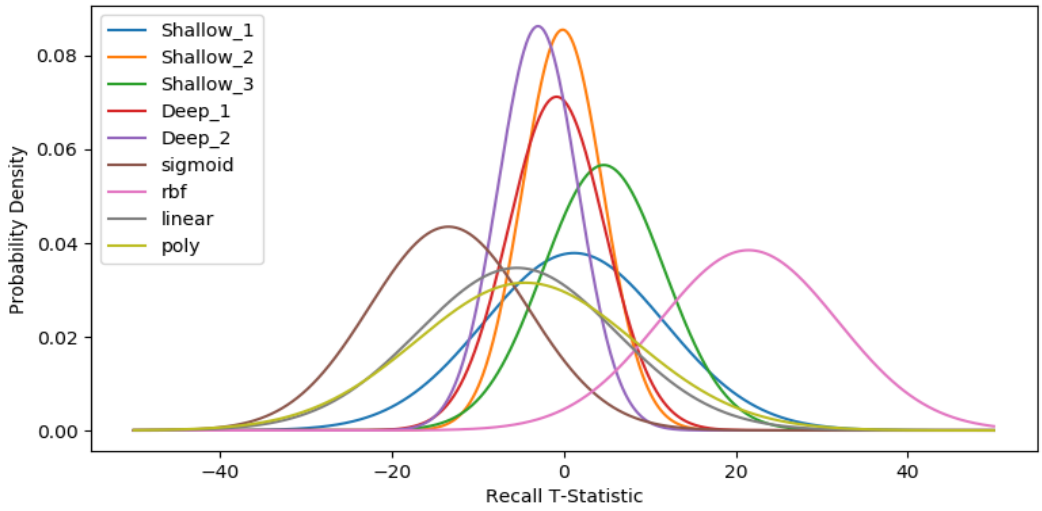

Figure 5.2: Probability density of Dependent T-Test T-statistics for the recall metric

Table 5.9: Ranking of models by significantly improved (p-val <0.05) recall results as indicated by the Dependent T-Test

Rank	Model	Type	No. Sig. Worse Models
1	rbf	SVM	8
2	Shallow_3	CNN	6
3	Shallow_1	CNN	4
4	linear	SVM	2
4	Deep_1	CNN	2
4	Shallow_2	CNN	2
5	Deep_2	CNN	1
5	poly	SVM	1
6	sigmoid	SVM	0

Analysis of F1 Results

The significance results for the F1 metric can be seen in Table 5.10. Models with a significance of <0.05 are considered to be significant at the confidence level of 95%. There are significant differences indicated between many of the results. However, there is no significant difference indicated between; Shallow_2 and linear, Shallow_2 and poly, Deep_1 and linear, Deep_1 and poly, Deep_2 and linear, Deep_2 and poly. This indicates that the CNN Shallow_2, Deep_1, and Deep_2 models do not offer significantly improved results when compared to the SVM linear and poly models for the F1 metric.

Table 5.10: Matrix of the Dependent T-Test P-Vals for the f1 metric for each model (DF=9). Significant results (<0.05%) are highlighted in bold.

Model	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	sigmoid	rbf	linear	poly
Shallow_1	N/A	0.145	**0.043**	**0.034**	**0.002**	**<.001**	**<.001**	**<.001**	**<.001**
Shallow_2	0.145	N/A	**0.032**	0.65	0.076	**0.001**	**<.001**	0.067	0.164
Shallow_3	**0.043**	**0.032**	N/A	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
Deep_1	**0.034**	0.65	**<.001**	N/A	**0.032**	**<.001**	**<.001**	0.182	0.324
Deep_2	**0.002**	0.076	**<.001**	**0.032**	N/A	**0.032**	**<.001**	0.591	0.311
sigmoid	**<.001**	**0.001**	**<.001**	**<.001**	**0.032**	N/A	**<.001**	**<.001**	**<.001**
rbf	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	N/A	**<.001**	**<.001**
linear	**<.001**	**0.067**	**<.001**	0.182	0.591	**<.001**	**<.001**	N/A	**0.013**
poly	**<.001**	0.164	**<.001**	0.324	0.311	**<.001**	**<.001**	**0.013**	N/A

The t-statistic values for the Dependent T-Test of the F1 metric can be seen in Table 5.11. The Dependent T-Tests are carried out in pairs, a negative number indicates that the results for the first model in the comparison are not as good as the results for the second model. Only the SVM RBF model results indicate that its F1 results are significantly better than the F1 results of both the CNN models and the other SVM models.

The performance is also represented in Figure 5.3 where it can be seen that the RBF kernel generates better results than the other models. When comparing the CNN model results to those of the SVM models, it is evident that Shallow_1, Shallow_2, Shallow_3, and Deep_1 models generated significantly better results than the SVM sigmoid, linear, and poly models for the F1 statistic. The Deep_2 model only generated better results than the SVM sigmoid model. The CNN model comparisons indicate that Shallow_3 offered the best performance, with a significantly better result than all other models except the SVM RBF model. A ranking of the models based on the F1 statistic results can be seen in Table 5.12. This ranking is based on the number of models where the model being evaluated generates a statistically improved result.

Table 5.11: Matrix of the Dependent T-Test T-statistic for the f1 metric for each model (DF=9). Significant results (p-val <0.05%) are highlighted in bold. A result of >0 indicates that a model generated better results, with results of <0 indicating poorer results. Reading from the left take the left value, reading from the top take the right value.

Model	Shallow_1	Shallow_2	Shallow_3	Deep_1	Deep_2	sigmoid	rbf	linear	poly
Shallow_1	N/A	1.59/-1.59	**-2.35/2.35**	**2.48/-2.48**	**4.21/-4.21**	**9.86/-9.86**	**-20.42/20.42**	**5.43/-5.43**	**4.80/-4.80**
Shallow_2	-1.59/1.59	N/A	**-2.53/2.53**	0.46/-0.46	2.00/-2.00	**4.63/-4.63**	**-10.38/10.38**	**2.08/-2.08**	1.51/-1.51
Shallow_3	**2.35/-2.35**	**2.53/-2.53**	N/A	**4.95/-4.95**	**8.24/-8.24**	**12.52/-12.52**	**-10.52/10.52**	**7.29/-7.29**	**7.44/-7.44**
Deep_1	**-2.48/2.48**	-0.46/0.46	**-4.95/4.95**	N/A	**2.53/-2.53**	**5.04/-5.04**	**-12.04/12.04**	1.44/-1.44	1.04/-1.04
Deep_2	**-4.21/4.21**	-2.00/2.00	**-8.24/8.24**	**-2.53/2.53**	N/A	**2.53/-2.53**	**-11.63/11.63**	-0.55/0.55	-1.07/1.07
sigmoid	**-9.86/9.86**	**-4.63/4.63**	**-12.52/12.52**	**-5.04/5.04**	**-2.53/2.53**	N/A	**-31.10/31.10**	**-7.39/7.39**	**-8.79/8.79**
rbf	**20.42/-20.42**	**10.38/-10.38**	**10.52/-10.52**	**12.04/-12.04**	**11.63/-11.63**	**31.10/-31.10**	N/A	**28.45/-28.45**	**30.87/-30.87**
linear	**-5.43/5.43**	-2.08/2.08	**-7.29/7.29**	-1.44/1.44	0.55/-0.55	**7.39/-7.39**	**-28.45/28.45**	N/A	**-3.07/3.07**
poly	**-4.80/4.80**	-1.51/1.51	**-7.44/7.44**	-1.04/1.04	1.07/-1.07	**8.79/-8.79**	**-30.87/30.87**	**3.07/-3.07**	N/A

Reading from left take the left number, reading from top take the right number - Significant results in bold

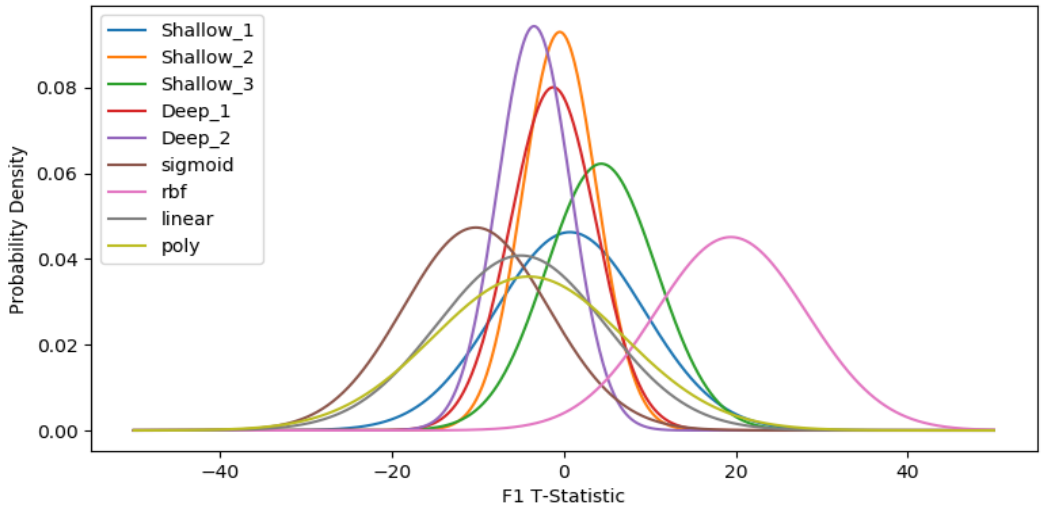

Figure 5.3: Probability density of Dependent T-Test T-statistics for the f1 metric

Table 5.12: Ranking of models by significantly improved (p-val <0.05) f1 results as indicated by the Dependent T-Test

Rank	Model	Type	No. Sig. Worse Models
1	rbf	SVM	8
2	Shallow_3	CNN	7
3	Shallow_1	CNN	5
4	Deep_1	CNN	2
5	Shallow_2	CNN	1
5	Deep_2	CNN	1
5	poly	SVM	1
6	sigmoid	SVM	0
6	linear	SVM	0

Overall Results

An overall ranking combining the ranking results for all three metrics is presented in Table 5.13. The SVM RBF model provided the most consistently better results of all the models evaluated. This is followed by the Shallow_3, and Shallow_1 CNN models.

Table 5.13: Ranking of models by significantly improved (p-val <0.05) results for all metrics as indicated by the Dependent T-Test

Rank	Model	Type	No. Sig. Worse Models
1	rbf	SVM	24
2	Shallow_3	CNN	14
3	Shallow_1	CNN	10
4	sigmoid	SVM	7
5	Deep_2	CNN	6
6	Deep_1	CNN	4
7	Shallow_2	CNN	3
7	poly	SVM	3
7	linear	SVM	3

5.1.3 Analysis

For this experiment the same data input was employed for all nine models, using Chi-squared test and PCA for dimensionality reduction, and TF-IDF to vectorise the text input. With this process, the input is converted into vectors with two-hundred dimensions, these dimensions represent the ones which are the most significantly tied to the intent for each record. As seen in Table 5.13, Shallow_3 is the best performing CNN model. In this case, Shallow_3 makes use of a CN layer with 150 one-dimension filters, instead of the 100 one-dimension filters employed by the other CNN models. By making use of more of the significant dimensions than the other models, this may explain why Shallow_3 model generated better results in this experiment.

Of the SVM models evaluated, the linear kernel provided amongst the poorest results, this would indicate that the data is not linearly separable to a great degree. However, the non-linear polynomial kernel tied with the linear kernel regarding general performance. The sigmoid kernel generated better results, with the RBF kernel generating the best results. The high performance of some of the non-linear kernels

would suggest that the kernel trick of mapping the data into a higher dimensional plane proves effective at transforming that data into a linearly separable set.

Figure 5.4[1] illustrates the decision boundaries in a 2-dimensional plane, for the four kernels, on three small sample datasets.

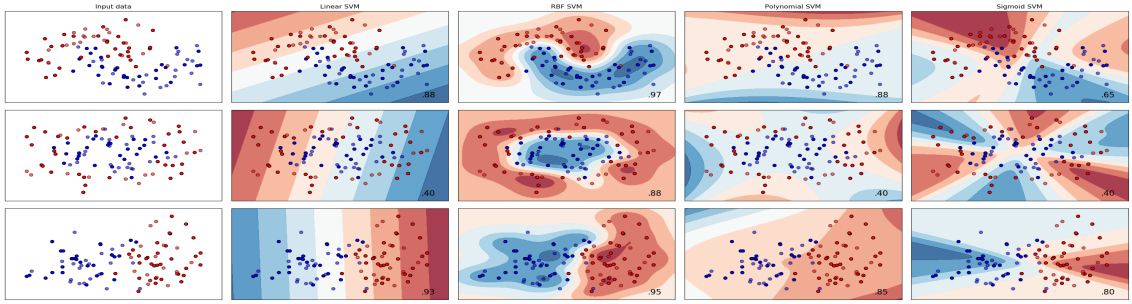

Figure 5.4: Representation of SVM kernel hyperplane separation on three sample datasets based on: [1]

The first two datasets are not linearly separable but the final dataset is. This representation in Figure 5.4 demonstrates the differences between the three nonlinear models which were utilised for the evaluation. The linear and poly models generated the worst results of those evaluated. The linear model decision boundary is a very rigid line separating the two pools of data, and polynomial is somewhat similar but less rigid in its separation. The sigmoid model generated the second best results and the representation illustrates that it is capable of forming multiple highly curved decision boundaries as needed. The RBF kernel forms the least rigid decision boundaries of those evaluated, with the boundaries able to flow around the data to form the best fit. As the model's transition from less to more flexible, there is, however, a concern that the models generated would over-fit the data. In this experiment, based on the data available and the NLP pre-processing steps carried out on the data, the RBF model was able to generate the most accurate model. This would indicate that the decision boundaries between the data points benefit from the most flexibility in terms of the

[1] http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

models' ability to fit the boundary or boundaries.

An experiment conducted by Min and Lee (2005) in the area of financial prediction where PCA was implemented for dimensionality reduction, and an SVM model for each of the four kernels (linear, RBF, sigmoid, polynomial) was trained and evaluated. The experiment suggested that either the RBF or polynomial kernels provided the best results, dependent on their hyper-parameters. The experiment ultimately found that the RBF kernel generated the best results. Although the experiment was not conducted in an NLP context, the results show similarities. This raises the question as to the impact of conducting a PCA on the SVM kernels. It may be that some kernels i.e. RBF and polynomial, are better at modelling the data generated by a PCA.

When considering linear kernels, W. Zhang, Yoshida, and Tang (2008) conducted an experiment to compare the classification results of a linear and non-linear SVM (RBF kernel) for an NLP classification task. NLP pre-processing was carried out and TF-IDF was employed to vectorise the text. No dimensionality reduction was carried out in this experiment. The experiment determines that the linear SVM model generated better results than the non-linear model. This agrees with the results seen in this experiment and the one carried out by Min and Lee (2005) in that the steps carried out to process the data, namely any dimensionality reduction may have an impact on which kernel proves most effective. In this experiment the dataset contained over a thousand dimensions after TF-IDF was applied. Another experiment carried out by Gamon (2004) conducted an NLP classification exercise using a linear SVM with good accuracy and F1 score. Again no PCA was carried out but some feature reduction steps were carried out to form datasets ranging in size between 1,000 and 4,000 dimensions. This could suggest that linear SVM models perform best in datasets with large dimensions without a transformation like PCA.

The experiments identified in the literature review generally reported better results for the CNN models when compared to other models, including SVMs. However, in this experiment, the CNN models did not generate the best results when compared to the SVM models evaluated. Two of the CNN models did generate results which were

often better than SVM models. The question is why did some models generate better results than others?

When comparing the results of this experiment with others where SVM and CNN models were evaluated, it would suggest that the model effectiveness can be dependent on the pre-processing steps carried out. This agrees with (Gaikwad & Joshi, 2016) when they state that: "The accuracy of the model is highly dependent on the way pre-processing is done". By altering the pre-processing steps carried out in this experiment, the effectiveness of any of the models could be increased or decrease.

Many of the NLP experiments identified, which make use of CNN models utilised a word embedding system like Word2Vec in order to generate the input vectors. These experiments generally report better performance for the CNN models than the other models evaluated. This may explain the difference between this experiment and the others identified. When CNN is combined with word embedding it may generate improved results. In addition, the proximity of words in a word embedding output is an important element of that output. When a typical CNN convolutional layer is considered, where the layer contains multiple filters of varying length, there is potential for the proximity-based output of a word embedding step to improve performance. Filters of length greater than a single dimension mean that several closely placed dimensions may be considered as input. Whereas, there is no such significance to the placement of dimensions in the output of the TF-IDF process employed in this experiment.

In addition, the application of PCA may also impact the performance with respect to CNN's. A typical CNN model is designed to utilise multiple filters of varying sizes in order to map the input. In this case, the initial experimentation of CNN models indicated that a single input dimension provided good results with no appreciable improvement for wider filters. This may be as a result of the PCA input, where the individual dimensions generated by the PCA are unrelated. By using wider sliding filters the CNN models are typically able to consider multiple combinations of input dimensions. Initial experiments indicated that single dimension filters offered good performance, possibly because the PCA generated dimensions are unrelated. When

this is considered in relation to text inputs, it could allow the CNN model to consider combinations of single and multiple words which would not normally be considered as part of an individual input. The SVM models, on the other hand, consider all two-hundred dimensions when training and evaluating the data. However, initial experiments did not show an appreciable gain in the CNN models when more than 150 dimensions were included.

5.1.4 Accepting/Rejecting the Hypothesis H1

The hypothesis H1 states that CNN models result in better precision, recall, and F1 scores, compared to comparable SVM models when generated and evaluated using a conversational dataset. The results of the Dependent T-Tests indicate that all of the CNN models evaluated, did not generate statistically significant, improved results, with 95% confidence, when comparing the precision, recall, and F1 metrics to the same metrics generated by all of the SVM models. As a result, the H1 Hypothesis is rejected.

5.2 Summary of evaluation

The first step in the evaluation process was to run Shapiro-Wilk test for normality in order to determine the appropriate statistical tests to be carried out on the result data. The Shapiro-Wilk test indicated that the results for all models and metrics are normally distributed.

As the results are normally distributed, dependent t-tests were run comparing the output of each model to every other model, for each of the three metrics; precision, recall, and F1 score. For each of the three metrics, the dependent t-test indicates that the SVM RBF model generated statistically better results than the other models which were evaluated, with 95% confidence. The Shallow 3 model generated the best results of out the five CNN models evaluated. However, the results for Shallow 3 model are not as good as the SVM RBF model.

The hypothesis H1 states that the CNN models will generate better results for the

three metrics when compared to the CNN models. As the results of the CNN models are not as good as the SVM models, the hypothesis H1 is ultimately rejected.

5.2.1 Discussion

A summary of the results based on the precision metric can be seen in Figure 5.5. This illustrates that the SVM RBF model generated statistically better results when compared to the results of every other model. The sigmoid model generated better results than every other model excluding RBF for this metric. The CNN Deep_2 model generated the next best results. Therefore, two SVM models generated better results than the best CNN model.

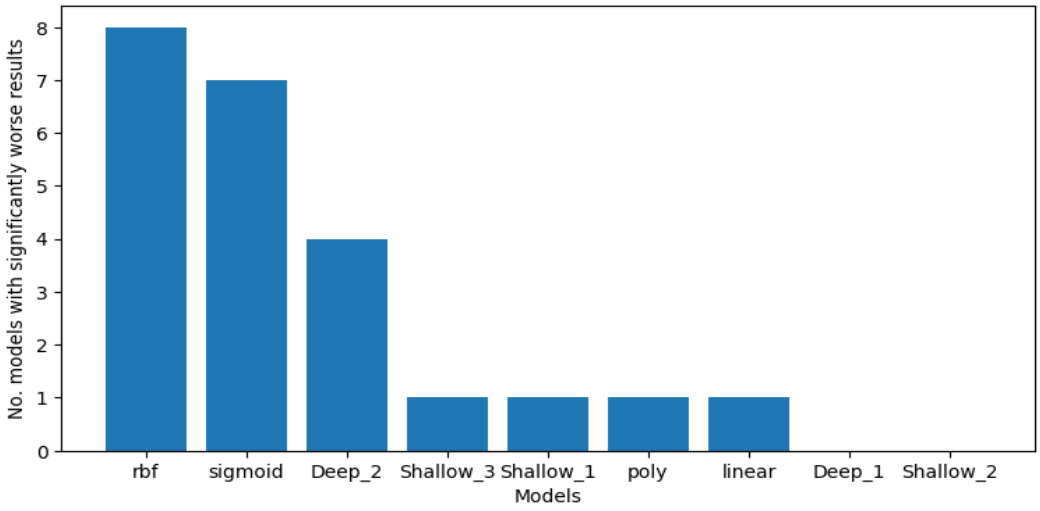

Figure 5.5: Representation of models by significantly improved (p-val <0.05) results for the precision metric as indicated by the Dependent T-Test

A summary of the results based on the recall metric can be seen in Figure 5.6. This again illustrates that the SVM RBF model generated statistically better results when compared to the results of every other model. Unlike the precision results, the CNN Shallow_3 and Shallow_1 models generated the next best results for this metric. The CNN models which generated the most significant results for the recall metric are different from those identified for the precision metric. The Shallow_3 model comes closest to the RBF model results, but ultimately only significantly improves on six of

the other models.

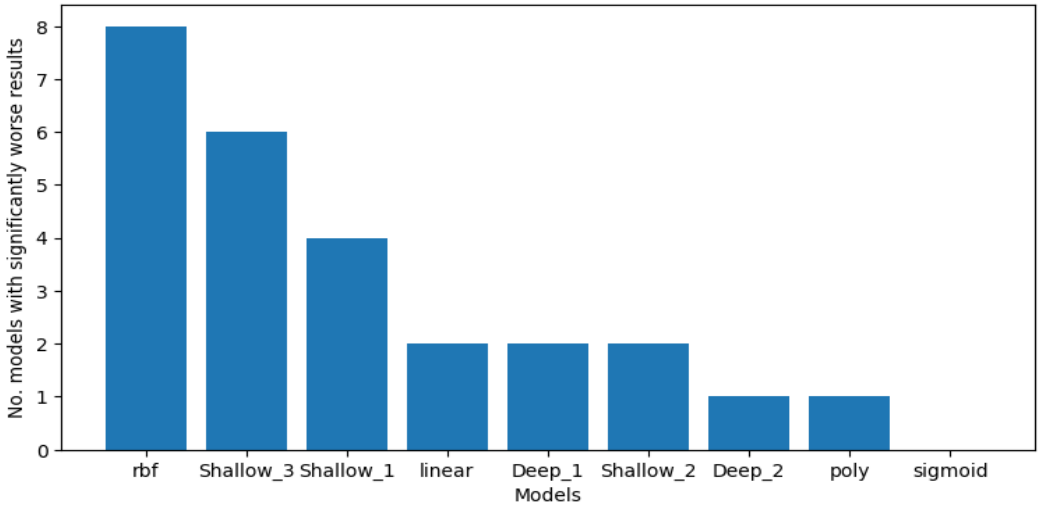

Figure 5.6: Representation of models by significantly improved (p-val <0.05) results for the recall metric as indicated by the Dependent T-Test

A summary of the results based on the f1 metric can be seen in Figure 5.7. Once again this illustrates that the SVM RBF model generated statistically better results when compared to the results of every other model. Like the recall results, the CNN Shallow_3 and Shallow_1 models generated the next best results for this metric. The results seen here are very similar to the results of the recall metric. The Shallow_3 model comes closest to the RBF model results, but ultimately only significantly improves on seven of the other models.

A summary of the results based on a combination of all metrics can be seen in Figure 5.8. As clearly illustrated here, the model using the SVM RBF kernel generated far more statistically better results than all of the other models. The RBF model was, in fact, the only model which generated statistically better results when compared to every over model over all three metrics. The next best models were the CNN models titled Shallow_3 and Shallow_1. These models performed better than the other model but overall are not close to matching the results of the RBF model.

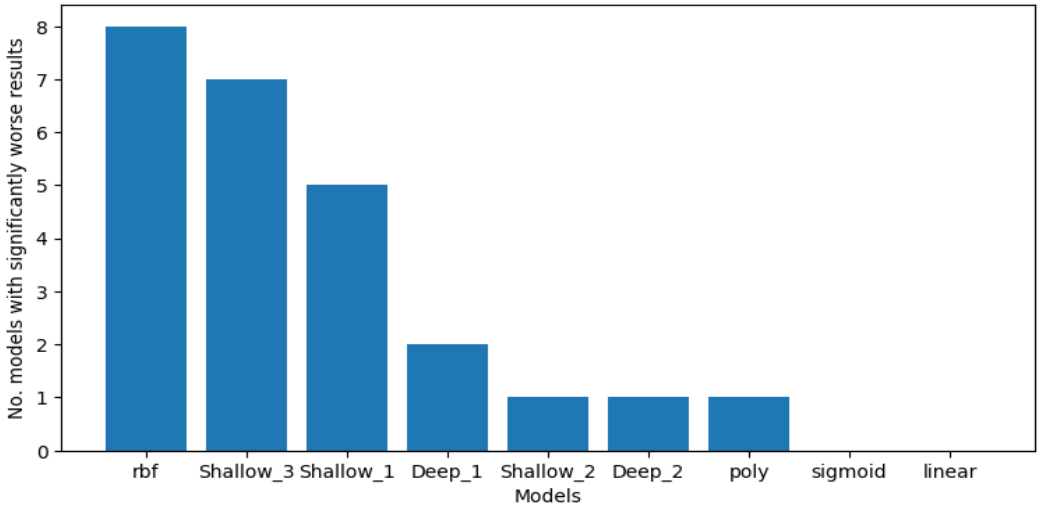

Figure 5.7: Representation of models by significantly improved (p-val <0.05) results for the f1 metric as indicated by the Dependent T-Test

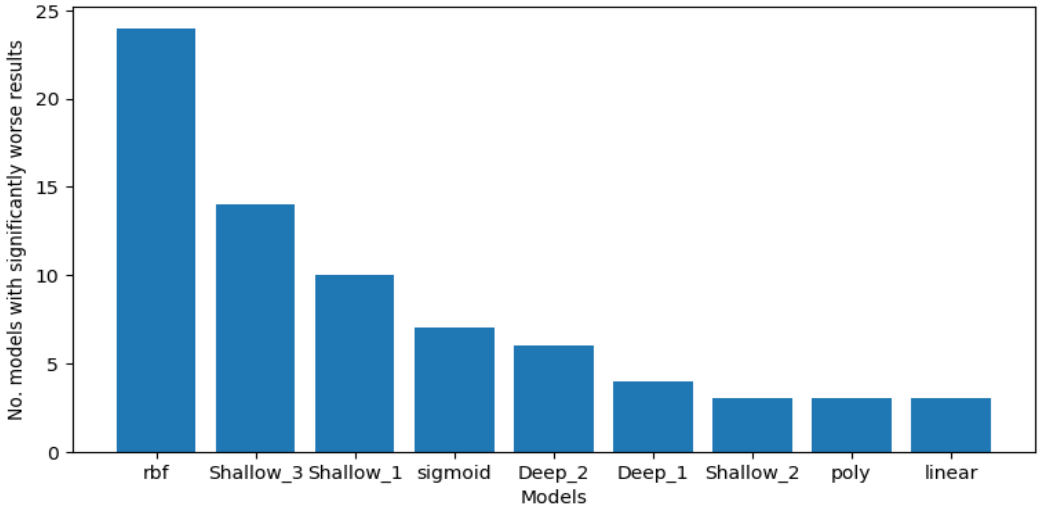

Figure 5.8: Representation of models by significantly improved (p-val <0.05) results for all metrics as indicated by the Dependent T-Test

As the CNN models do not generate statistically better results than the SVM models for the considered metrics, the hypothesis H1 is rejected. As the experiment results in the hypothesis being rejected, this allows an answer to the research question defined in the literature review chapter to be determined: "CNN models are not necessarily more effective than SVM models at determining intent, for use in conversational

agents".

Most literature identified in the review suggested that CNN models generated better results for text classification tasks when compared to SVM models. This experiment, however, resulted in an SVM model (RBF) generating better results than all of the evaluated CNN models. There are several factors which may explain the difference between this experiment and the others identified.

Comparing this experiment to the others identified, it indicates that the pre-processing steps carried out on the NLP data can have a significant impact on the performance of the different models. For example, the inclusion of PCA as a dimensionality reduction step may explain why the Shallow 3 model generates better results than the other models, as it considers more of the important dimensions than the other CNN models.

Existing literature suggests that using a higher dimensionality dataset with the CNN models may lead to better results for those models. In particular, several papers suggest using word embedding models with CNN results in better results than other feature extraction techniques. However, changing the input dataset in this way would require the CNN model architectures and hyper-parameters to be re-evaluated. In particular, word embedding would require two-dimensional convolutional layers instead of the one-dimensional layers employed in this experiment. The pre-processing steps may also impact the performance of the SVM models. The result indicates the potential that RBF and sigmoid models generate the best results when PCA is utilised in the pre-processing phase. Research indicates that linear models may generate better results for high dimensionality datasets which do not make use of PCA dimensionality reduction.

5.2.2 Strengths and limitations of findings

This experiment was based on the theory that CNN models would offer better performance for intent detection in conversational agents. Firstly, the use of a conversational agent based dataset is a strength in terms of the stated objective of the experiment. Very few experiments in this area have been built on a dataset which is particularly

relevant to this domain.

As stated previously, optimum configurations for each of the four SVM kernels and five CNN kernels were generated using a grid search mechanism. As such, this can be described as a comprehensive review where multiple optimised models for both algorithms are compared. In addition, ten independent models are trained and evaluated for each model using k-fold stratified sampling. This results in ninety models being created and assessed. For each of the models, three metrics are considered; precision, recall, and f1 score, resulting in two-hundred and seventy data points for statistical analysis, the volume of data points being a strength of this experiment.

A robust NLP pipeline was researched and defined based on the particular dataset and the problem under review. This pipeline took into account the specific requirements of intent detection for conversational interfaces. In addition, cutting-edge NLP tools like lemmatisation and TF-IDF were used to form the NLP pipeline. A thorough statistical analysis of the results is another strength of this experiment. This analysis consisted of seventy-two dependent t-tests being executed to compare the results for each metric, and two-hundred and sixteen overall statistical tests for significance. This results in a comprehensive analysis of the effectiveness of the models under review.

One drawback of this experiment is that the dataset has a significant imbalance in the data between the twenty classes. This imbalance could potentially cause issues with any of the models evaluated. The experiment attempted to mitigate the class imbalance problems by introducing class weights in the models and by using weighted measures of the precision, recall, and F1 metrics. However, further research would need to be carried out in order to evaluate the full impact of the imbalanced classes. Additionally, four SVM kernels were explored in this experiment, while over twenty individual kernels are possible, and combinations of kernels can also be explored (J. Zhang, 2015). The four kernels employed are some of the most popular available, and the only ones currently implemented in the scikit-learn package which was utilised to conduct this experiment. There is the potential that other kernels may generate even better results than the top performing RBF kernel identified in this experiment.

There are also a large number of possible CNN models which could be applied

to text classification problems. The basic architectures explored here are variations of a general purpose NLP architecture (Collobert & Weston, 2008). There are many variations of this architecture which have the potential to offer better results than the ones evaluated. The research also indicates that the pre-processing steps carried out may be having an impact on the CNN results. By exploring alternate pre-processing steps, especially around vectorisation, it may be possible to obtain better CNN model results.

Another potential drawback is the general NLP pipeline which was defined for the experiment. A single pipeline was defined which could work with both SVM and CNN algorithms, but there is the potential that a specific pipeline could be tuned for each algorithm, resulting in better results for that model.

Chapter 6

Conclusion

This chapter provides a brief overview of the research carried out and the problem under investigation. In addition the experimental design, evaluation, and results are summarised. This is followed by a discussion of the contributions and impact of the experiment and lastly an overview of future work which could expand on the work carried out in this experiment.

6.1 Research Overview

The development and use of conversational agents has grown over the last 50 years. From simple chatbots like ELIZA which could carry on limited conversations with users, to modern complex bot applications which provide a conversational interface for performing both simple and complex tasks. In the last number of years, the fields of natural language processing and machine learning have made vast improvements in the ability of systems to handle complex natural language interaction. These advancements can be applied to conversational agents in many areas, but a primary use case is in natural language understanding, and more specifically, intent detection. However, few of these advancements have been directly evaluated in a conversational agent context.

6.2 Problem Definition

Modern conversational agents provide a new form of user interface for traditional software systems. With the development of modern NLP practices and tools, along with new machine learning algorithms, the question for the construction of an agent becomes, what practices, tools, and algorithms are best to utilisation in this context.

This research has identified a good representation of a conversational agent dataset in the form of the Frames dataset (El Asri et al., 2017). Modern NLP pre-processing steps have been identified to prepare the data for classification. Two of the most commonly identified high performing machine learning algorithms for NLP classification tasks have been identified in the form of SVM and CNN. CNN is typically regarded as generating better results than SVM models, so an experiment was designed to explore multiple models for each algorithm and to scientifically compare the results.

6.3 Design/Experimentation, Evaluation & Results

The CRISP-DM life-cycle was employed as the basis for the experimental design. The first step (data understanding) is gathering a detailed understanding of the data in the dataset under consideration. Next comes data preparation which consists of several steps: missing data removal, Lemmatisation, noise reduction through pattern matching, feature extraction in the form of TF-IDF, feature selection using Chi-squared tests and PCA, determining of class weights for imbalanced data, and finally ten-fold stratified cross-validation split.

A modelling step follows where the machine learning models are built and trained. Four SVM kernels were identified with a range of possible parameters for each kernel. A grid search was carried out to identify the optimal parameters for each model. A similar process was carried out for five CNN architectures which were defined based on the existing literature. In this case, a grid search was utilised to identify the activation, pooling type, and optimiser for each architecture. Each of the models was trained and evaluated against all ten folds of the data, and the following metrics were recorded;

precision, recall, F1. Every metric was calculated as a weighted average due to an imbalance in the dataset.

In conclusion, the evaluation was carried out on the recorded metrics. Each collection of data was tested for normality using Shapiro-Wilk tests, where each dataset reported normality. Next, a series of Dependent T-Tests were carried out for each metric, where the results from the metric for each model were compared to all of the other results for that metric. These tests indicated that in this experiment the CNN models did not generate significantly better results than the SVM models. In fact, the SVM model using the RBF kernel generated significantly better results than all of the other models, over all three metrics. As a result, the hypothesis H1 is ultimately rejected, as the CNN models did not generate better results than the SVM models in this experiment.

6.4 Contributions and impact

The purpose of this paper is to compare the use of SVM and CNN algorithms for use in intent detection for conversational interfaces. As published experiments in the NLP space typically indicate that CNN models offer better performance than SVM models, the research question which drives the experiment is: "Are CNN more effective than SVM at determining intent, for use in conversational agents?"

To provide a comprehensive answer to this question, several models were optimised, trained, and evaluated for both algorithms, using ten-fold stratified cross-validation in order to repeat the experiment multiple times utilising all available data. Then a detailed statistical analysis was carried out on three metrics to compare the results of all models. Through the utilisation of these rigorous methods, it is proposed that this experiment makes a significant contribution in the area of NLP and ML for use as intent detection in conversational agent systems.

The results of this experiment indicate that CNN models are not necessarily better than SVM models for intent detection. In fact, it is indicated that an SVM model using an RBF kernel has the potential to provide better results than CNN models.

The answer to the question posed earlier is, therefore: "No, CNN are not necessarily more effective than SVM at determining intent, for use in conversational agents". By indicating that SVM models can potentially achieve better results than SVM models, this experiment suggests that SVM models should be considered for an intent detection component of a conversational agent system. However, the experimental results in the context of existing literature do suggest that changes to the pre-processing pipeline like the utilisation of word embedding and the removal of PCA may alter the results of the models. This would indicate that further research in this area is of value.

6.5 Future Work & recommendations

The primary recommendation from this experiment is to extend the comparison between the SVM and CNN models to identify the optimal pre-processing step for each model so that each model generates the best possible results before a statistical analysis is carried out. Of particular interest are dimensionality reduction steps taken and their impact on the models. For example, Li and Shen (2017) suggested LSA as a viable dimensionality reduction step for NLP tasks. Additionally, there are numerous SVM kernels, and alternate CNN architectures which can be implemented, which have the potential to provide better results than the results generated in this experiment. The inclusion of other parameters in the CNN GridSearch may also improve performance, e.g. the number of filters. In conclusion, due to the imbalanced nature of the dataset, additional work to evaluate the impact, and possibly address the imbalance in the dataset.

Bibliography

Acharya, S., & Parija, S. (2010). The process of information extraction through natural language processing. *International Journal of Logic and Computation (IJLP)*, *1*(1), 40–51.

Al-Dmour, H., & Al-Ani, A. (2016, Nov). Mr brain image segmentation based on unsupervised and semi-supervised fuzzy clustering methods. In *2016 international conference on digital image computing: Techniques and applications (dicta)* (p. 1-7). doi: 10.1109/DICTA.2016.7797066

Ayat, N., Cheriet, M., & Suen, C. (2005). Automatic model selection for the optimization of svm kernels. *Pattern Recognition*, *38*(10), 1733 - 1745. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0031320305001433` doi: https://doi.org/10.1016/j.patcog.2005.03.011

Bhargava, A., Celikyilmaz, A., Hakkani-Tr, D., & Sarikaya, R. (2013, May). Easy contextual intent prediction and slot detection. In *2013 ieee international conference on acoustics, speech and signal processing* (p. 8337-8341). doi: 10.1109/ICASSP.2013.6639291

Bieliauskas, S., & Schreiber, A. (2017, Sept). A conversational user interface for software visualization. In *2017 ieee working conference on software visualization (vissoft)* (p. 139-143). doi: 10.1109/VISSOFT.2017.21

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: Analyzing text with the natural language toolkit*. O'Reilly Media. Retrieved from `https://books.google.ie/books?id=KGIbfiiP1i4C`

Bonaccorso, G. (2017). *Machine learning algorithms: A reference guide to popular algorithms for data science and machine learning*. Packt Publishing.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167. Retrieved from https://doi.org/10.1023/a:1009715923555 doi: 10.1023/a:1009715923555

Byun, H., & Lee, S.-W. (2002). Applications of support vector machines for pattern recognition: A survey. In *Proceedings of the first international workshop on pattern recognition with support vector machines* (pp. 213–236). London, UK, UK: Springer-Verlag. Retrieved from http://dl.acm.org/citation.cfm?id=647230.719394

Cambria, E., & White, B. (2014, May). Jumping nlp curves: A review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, *9*(2), 48-57. doi: 10.1109/MCI.2014.2307227

Chen, L., Zhang, D., & Mark, L. (2012). Understanding user intent in community question answering. In *Proceedings of the 21st international conference on world wide web* (pp. 823–828). New York, NY, USA: ACM. doi: 10.1145/2187980.2188206

Cheung, E., & Li, Y. (2017, May). Self-training with adaptive regularization for s3vm. In *2017 international joint conference on neural networks (ijcnn)* (p. 3633-3640). doi: 10.1109/IJCNN.2017.7966313

Colby, K. M. (1981, dec). Modeling a paranoid mind. *Behavioral and Brain Sciences*, *4*(04), 515. Retrieved from https://doi.org/10.1017/s0140525x00000030 doi: 10.1017/s0140525x00000030

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1390156.1390177 doi: 10.1145/1390156.1390177

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011, November). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, *12*, 2493–2537. Retrieved from http://dl.acm.org/citation.cfm?id=1953048.2078186

Cox, K., Grinter, R. E., & Mantilla, D. (2000). Using dialog and context in a speech-based interface for an information visualization environment. In *Proceedings of the working conference on advanced visual interfaces* (pp. 274–275). New York, NY, USA: ACM. doi: 10.1145/345513.345342

Culp, M., & Michailidis, G. (2008, Jan). Graph-based semisupervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(1), 174-179. doi: 10.1109/TPAMI.2007.70765

Ding, X., Liu, T., Duan, J., & Nie, J.-Y. (2015). Mining user consumption intention from social media using domain adaptive convolutional neural network. In *Proceedings of the twenty-ninth aaai conference on artificial intelligence* (pp. 2389–2395). AAAI Press. Retrieved from `http://dl.acm.org/citation.cfm?id=2886521.2886653`

Draskovic, D., Gencel, V., Zitnik, S., Bajec, M., & Nikolic, B. (2016, Nov). A software agent for social networks using natural language processing techniques. In *2016 24th telecommunications forum (telfor)* (p. 1-4). doi: 10.1109/TELFOR.2016.7818921

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on information and knowledge management* (pp. 148–155). New York, NY, USA: ACM. doi: 10.1145/288627.288651

El Asri, L., Schulz, H., Sharma, S., Zumer, J., Harris, J., Fine, E., ... Suleman, K. (2017, August). Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th annual sigdial meeting on discourse and dialogue* (pp. 207–219). Saarbrucken, Germany: Association for Computational Linguistics. Retrieved from `http://aclweb.org/anthology/W17-5526`

Feldman, S. (1999). Nlp meets the jabberwocky: Natural language processing in information retrieval, online , 23:3. url: http://www.onlineinc.com/onlinemag/ol1999/feldman5.html feldman, s.(jan 2000), the answer machine, searcher , 8:1. url: http://www.infotoday.com/search. In *Proceedings of the fifth hong kong*

web symposium.

Firmino Alves, A. L., Baptista, C. d. S., Firmino, A. A., Oliveira, M. G. a. d., & Paiva, A. C. d. (2014). A comparison of svm versus naive-bayes techniques for sentiment analysis in tweets: A case study with the 2013 fifa confederations cup. In *Proceedings of the 20th brazilian symposium on multimedia and the web* (pp. 123–130). New York, NY, USA: ACM. doi: 10.1145/2664551.2664561

Forman, G. (2008). Bns feature scaling: An improved representation over tf-idf for svm text classification. In *Proceedings of the 17th acm conference on information and knowledge management* (pp. 263–270). New York, NY, USA: ACM. doi: 10.1145/1458082.1458119

Furlan, B., Batanovic, V., & Nikolic, B. (2013). Semantic similarity of short texts in languages with a deficient natural language processing support. *Decision Support Systems*, *55*, 710-719.

Gaikwad, G., & Joshi, D. J. (2016, Aug). Multiclass mood classification on twitter using lexicon dictionary and machine learning algorithms. In *2016 international conference on inventive computation technologies (icict)* (Vol. 1, p. 1-6). doi: 10.1109/INVENTIVE.2016.7823247

Gamon, M. (2004). Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on computational linguistics.* Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from `https://doi.org/10.3115/1220355.1220476` doi: 10.3115/1220355.1220476

Gharatkar, S., Ingle, A., Naik, T., & Save, A. (2017, March). Review preprocessing using data cleaning and stemming technique. In *2017 international conference on innovations in information, embedded and communication systems (iciiecs)* (p. 1-4). doi: 10.1109/ICIIECS.2017.8276011

Graf, B., Krüger, M., Müller, F., Ruhland, A., & Zech, A. (2015). Nombot: Simplify food tracking. In *Proceedings of the 14th international conference on mobile and ubiquitous multimedia* (pp. 360–363). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2836041.2841208` doi: 10.1145/2836041

.2841208

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, *28*(1), 100. Retrieved from https://doi.org/10.2307/2346830 doi: 10.2307/2346830

Hassan, A., & Mahmood, A. (2017, April). Deep learning approach for sentiment analysis of short texts. In *2017 3rd international conference on control, automation and robotics (iccar)* (p. 705-710). doi: 10.1109/ICCAR.2017.7942788

Hayashida, Y., Uetsuji, T., Ebara, Y., & Koyamada, K. (2017, jun). Category classification of text data with machine learning technique for visualizing flow of conversation in counseling. In *2017 nicograph international (NicoInt).* IEEE. doi: 10.1109/nicoint.2017.35

Hijjawi, M., Bandar, Z., & Crockett, K. (2013, mar). User's utterance classification using machine learning for arabic conversational agents. In *2013 5th international conference on computer science and information technology.* IEEE. doi: 10.1109/csit.2013.6588784

Hipp, J., Güntzer, U., & Nakhaeizadeh, G. (2000, June). Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explor. Newsl.*, *2*(1), 58–64. Retrieved from http://doi.acm.org/10.1145/360402.360421 doi: 10.1145/360402.360421

Hollerit, B., Kröll, M., & Strohmaier, M. (2013). Towards linking buyers and sellers: Detecting commercial intent on twitter. In *Proceedings of the 22nd international conference on world wide web* (pp. 629–632). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2487788.2488009 doi: 10.1145/2487788.2488009

Ignatov, D., & Ignatov, A. (2017, Nov). Decision stream: Cultivating deep decision trees. In *2017 ieee 29th international conference on tools with artificial intelligence (ictai)* (p. 905-912). doi: 10.1109/ICTAI.2017.00140

Imane, A., & Mohamed, B. A. (2017). Multi-label categorization of french death certificates using nlp and machine learning. In *Proceedings of the 2nd international conference on big data, cloud and applications* (pp. 29:1–29:4). New York, NY,

USA: ACM. Retrieved from http://doi.acm.org/10.1145/3090354.3090384 doi: 10.1145/3090354.3090384

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the sixteenth international conference on machine learning* (pp. 200–209). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645528.657646

Jurafsky, D., & Martin, J. H. (2009). Speech and language processing (2nd edition). In (2nd ed., p. 418-440). Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Klopfenstein, L. C., Delpriori, S., Malatini, S., & Bogliolo, A. (2017). The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 conference on designing interactive systems* (pp. 555–565). New York, NY, USA: ACM. doi: 10.1145/3064663.3064672

Korenius, T., Laurikkala, J., Järvelin, K., & Juhola, M. (2004). Stemming and lemmatization in the clustering of finnish text documents. In *Proceedings of the thirteenth acm international conference on information and knowledge management* (pp. 625–633). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1031171.1031285 doi: 10.1145/1031171.1031285

Lau, T., Cerruti, J., Manzato, G., Bengualid, M., Bigham, J. P., & Nichols, J. (2010). A conversational interface to web automation. In *Proceedings of the 23nd annual acm symposium on user interface software and technology* (pp. 229–238). New York, NY, USA: ACM. doi: 10.1145/1866029.1866067

Lee, J. Y., & Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies.* Association for Computational Linguistics. doi: 10.18653/v1/n16-1062

Li, Y., & Shen, B. (2017, Dec). Research on sentiment analysis of microblogging based on lsa and tf-idf. In *2017 3rd ieee international conference on computer and communications (iccc)* (p. 2584-2588). doi: 10.1109/CompComm.2017.8323002

Lilleberg, J., Zhu, Y., & Zhang, Y. (2015, July). Support vector machines and

word2vec for text classification with semantic features. In *2015 ieee 14th international conference on cognitive informatics cognitive computing (icci*cc)* (p. 136-140). doi: 10.1109/ICCI-CC.2015.7259377

Ma, Q. (2002, Dec). Natural language processing with neural networks. In *Language engineering conference, 2002. proceedings* (p. 45-56). doi: 10.1109/LEC.2002.1182290

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*.

Min, J. H., & Lee, Y.-C. (2005, May). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Syst. Appl.*, *28*(4), 603–614. Retrieved from `http://dx.doi.org/10.1016/j.eswa.2004.12.008` doi: 10.1016/j.eswa.2004.12.008

Mishra, A., & Vishwakarma, S. (2015, Dec). Analysis of tf-idf model and its variant for document retrieval. In *2015 international conference on computational intelligence and communication networks (cicn)* (p. 772-776). doi: 10.1109/CICN.2015.157

Mitchell, T. M. (1997). *Machine learning (mcgraw-hill international editions computer science series)*. McGraw-Hill.

Nazzal, J., M. El-Emary, I., & A. Najim, S. (2008, 01). Multilayer perceptron neural network (mlps) for analyzing the properties of jordan oil shale. , *5*.

Niculescu, A. I., Yeo, K. H., Haro, L. F. D., Kim, S., Jiang, R., & Banchs, R. E. (2014, dec). Design and evaluation of a conversational agent for the touristic domain. In *Signal and information processing association annual summit and conference (APSIPA), 2014 asia-pacific*. IEEE. doi: 10.1109/apsipa.2014.7041744

Nii, M., Tsuchida, Y., Kato, Y., Uchinuno, A., & Sakashita, R. (2017, jun). Nursing-care text classification using word vector representation and convolutional neural networks. In *2017 joint 17th world congress of international fuzzy systems association and 9th international conference on soft computing and intelligent systems (IFSA-SCIS)*. IEEE. doi: 10.1109/ifsa-scis.2017.8023240

Ortiz, C. L. (2014, mar). The road to natural conversational speech interfaces. *IEEE*

Internet Computing, *18*(2), 74–78. doi: 10.1109/mic.2014.36

Purohit, H., Dong, G., Shalin, V., Thirunarayan, K., & Sheth, A. (2015, Dec). Intent classification of short-text on social media. In *2015 ieee international conference on smart city/socialcom/sustaincom (smartcity)* (p. 222-228). doi: 10.1109/SmartCity.2015.75

Rahman, Y. A., Sohan, M. A., Zinnah, K. I., & Hoque, M. M. (2017, Feb). A framework for building a natural language interface for bangla. In *2017 international conference on electrical, computer and communication engineering (ecce)* (p. 935-940). doi: 10.1109/ECACE.2017.7913037

Ryoo, J., Arunachalam, M., Khanna, R., & Kandemir, M. T. (2018, March). Efficient k nearest neighbor algorithm implementations for throughput-oriented architectures. In *2018 19th international symposium on quality electronic design (isqed)* (p. 144-150). doi: 10.1109/ISQED.2018.8357279

Sarikaya, R., Crook, P. A., Marin, A., Jeong, M., Robichaud, J. P., Celikyilmaz, A., ... Radostev, V. (2016, Dec). An overview of end-to-end language understanding and dialog management for personal digital assistants. In *2016 ieee spoken language technology workshop (slt)* (p. 391-397). doi: 10.1109/SLT.2016.7846294

Sarker, A., & Gonzalez, G. (2015, feb). Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *Journal of Biomedical Informatics*, *53*, 196–207. Retrieved from `https://doi.org/10.1016/j.jbi.2014.11.002` doi: 10.1016/j.jbi.2014.11.002

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd acm conference on electronic commerce* (pp. 158–167). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/352871.352887` doi: 10.1145/352871.352887

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/371920.372071` doi: 10.1145/371920.372071

Sebastiani, F. (2002, March). Machine learning in automated text categorization. *ACM Comput. Surv.*, *34*(1), 1–47. doi: 10.1145/505282.505283

Sharma, Y., Agrawal, G., Jain, P., & Kumar, T. (2017, Dec). Vector representation of words for sentiment analysis using glove. In *2017 international conference on intelligent communication and computational techniques (icct)* (p. 279-284). doi: 10.1109/INTELCCT.2017.8324059

Shearer, C. (2000, 01). The crisp-dm model: the new blueprint for data mining. *Journal of Data Warehouse*, *5*(4), 13-22.

Sokolova, M., & Lapalme, G. (2009, jul). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. Retrieved from `https://doi.org/10.1016/j.ipm.2009.03.002` doi: 10.1016/j.ipm.2009.03.002

Tang, B., Cao, H., Wu, Y., Jiang, M., & Xu, H. (2012). Clinical entity recognition using structural support vector machines with rich features. In *Proceedings of the acm sixth international workshop on data and text mining in biomedical informatics* (pp. 13–20). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2390068.2390073` doi: 10.1145/2390068.2390073

Turing, A. M. (1950). I.—Computing Machinery And Intelligence. *Mind*, *LIX*(236), 433–460. Retrieved from `https://doi.org/10.1093/mind/lix.236.433` doi: 10.1093/mind/lix.236.433

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley-Interscience. Retrieved from `https://www.amazon.com/Statistical-Learning-Theory-Vladimir-Vapnik/dp/0471030031?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0471030031`

Wallace, R. S. (2009). The anatomy of a.l.i.c.e. In R. Epstein, G. Roberts, & G. Beber (Eds.), *Parsing the turing test: Philosophical and methodological issues in the quest for the thinking computer* (pp. 181–210). Dordrecht: Springer Netherlands. doi: 10.1007/978-1-4020-6710-5_13

Wallach, H. M. (2006). Topic modeling: Beyond bag-of-words. In *Proceedings of the*

23rd international conference on machine learning (pp. 977–984). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1143844.1143967 doi: 10.1145/1143844.1143967

Waltz, D. L. (1977, February). Natural language interfaces. *SIGART Bull.*(61), 16–17. doi: 10.1145/1045283.1045285

Wang, X., & Yuan, C. (2016, oct). Recent advances on human-computer dialogue. *CAAI Transactions on Intelligence Technology*, *1*(4), 303–312. doi: 10.1016/j.trit.2016.12.004

Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in nlp. In *Proceedings of the 14th conference on computational linguistics - volume 4* (pp. 1106–1110). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/992424.992434 doi: 10.3115/992424.992434

Weizenbaum, J. (1966, January). Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, *9*(1), 36–45. Retrieved from http://doi.acm.org/10.1145/365153.365168 doi: 10.1145/365153.365168

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques (morgan kaufmann series in data management systems)*. Morgan Kaufmann.

Xu, P., & Sarikaya, R. (2013, Dec). Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding* (p. 78-83). doi: 10.1109/ASRU.2013.6707709

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international acm sigir conference on research and development in information retrieval* (pp. 42–49). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/312624.312647 doi: 10.1145/312624.312647

Zeroual, I., & Lakhouaja, A. (2017, April). Arabic information retrieval: Stemming or lemmatization? In *2017 intelligent systems and computer vision (iscv)* (p. 1-6).

doi: 10.1109/ISACV.2017.8054932

Zhang, J. (2015). A complete list of kernels used in support vector machines. *Biochemistry & Pharmacology: Open Access*, *04*(05). Retrieved from https://doi.org/10.4172/2167-0501.1000195 doi: 10.4172/2167-0501.1000195

Zhang, R., Ma, S., Shanahan, L., Munroe, J., Horn, S., & Speedie, S. (2017, Nov). Automatic methods to extract new york heart association classification from clinical notes. In *2017 ieee international conference on bioinformatics and biomedicine (bibm)* (p. 1296-1299). doi: 10.1109/BIBM.2017.8217848

Zhang, W., Yoshida, T., & Tang, X. (2008). Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, *21*(8), 879 - 886. Retrieved from http://www.sciencedirect.com/science/article/pii/S0950705108000968 doi: https://doi.org/10.1016/j.knosys.2008.03.044

Zhang, Y., & Wallace, B. (2017). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the eighth international joint conference on natural language processing (volume 1: Long papers)* (pp. 253–263). Asian Federation of Natural Language Processing. Retrieved from http://aclweb.org/anthology/I17-1026

Zhao, R., & Mao, K. (2018, April). Fuzzy bag-of-words model for document representation. *IEEE Transactions on Fuzzy Systems*, *26*(2), 794-804. doi: 10.1109/TFUZZ.2017.2690222

Zue, V., & Glass, J. (2000, aug). Conversational interfaces: advances and challenges. *Proceedings of the IEEE*, *88*(8), 1166–1180. doi: 10.1109/5.880078

Appendix A

Additional content

Table A.1: Words with greater then 500 occurrences in the un-processed dataset

Ordinal	Word	No.	Ordinal	Word	No.	Ordinal	Word	No.	Ordinal	Word	No.
1	to	8550	**21**	it	1889	**41**	august	1099	**61**	leave	676
2	the	7820	**22**	on	1792	**42**	go	1019	**62**	4	673
3	i	6896	**23**	we	1777	**43**	trip	901	**63**	need	661
4	you	6829	**24**	like	1765	**44**	about	894	**64**	has	643
5	a	6764	**25**	do	1753	**45**	near	890	**65**	only	617
6	and	5756	**26**	at	1747	**46**	no	868	**66**	want	613
7	for	4782	**27**	free	1638	**47**	days	858	**67**	leaving	600
8	is	4331	**28**	with	1586	**48**	but	833	**68**	flights	599
9	have	3968	**29**	me	1470	**49**	or	822	**69**	just	591
10	in	3163	**30**	package	1430	**50**	available	811	**70**	5	590
11	from	2991	**31**	your	1410	**51**	not	807	**71**	i'm	580
12	would	2729	**32**	september	1399	**52**	if	806	**72**	stay	577
13	of	2536	**33**	what	1390	**53**	guest	789	**73**	dates	536
14	hotel	2491	**34**	book	1366	**54**	any	763	**74**	budget	535
15	can	2334	**35**	will	1340	**55**	get	760	**75**	as	533
16	that	2028	**36**	how	1208	**56**	one	753	**76**	sept	530
17	star	1979	**37**	there	1205	**57**	3	708	**77**	also	516
18	are	1957	**38**	my	1202	**58**	economy	693	**78**	rating	502
19	this	1904	**39**	day	1103	**59**	where	683			
20	be	1903	**40**	business	1100	**60**	class	682			

Table A.2: Words with greater then 500 occurrences in the processed dataset

Ordinal	**Word**	**No.**	**Ordinal**	**Word**	**No.**	**Ordinal**	**Word**	**No.**
1	[gpe]	7502	**21**	breakfast	1005	**41**	ok	643
2	[datetime]	4692	**22**	date	975	**42**	only	629
3	hotel	3598	**23**	available	964	**43**	leaving	605
4	from	3310	**24**	class	939	**44**	please	585
5	[currency]	3233	**25**	one	929	**45**	5	581
6	star	2467	**26**	about	901	**46**	all	521
7	day	2412	**27**	near	897	**47**	adult	518
8	package	2296	**28**	guest	893	**48**	so	517
9	free	1611	**29**	leave	787	**49**	anything	514
10	trip	1576	**30**	economy	778	**50**	out	508
11	what	1418	**31**	stay	761	**51**	more	502
12	book	1416	**32**	yes	755			
13	go	1262	**33**	where	731			
14	flight	1235	**34**	parking	710			
15	wifi	1214	**35**	cost	710			
16	how	1205	**36**	need	708			
17	business	1166	**37**	3	697			
18	budget	1149	**38**	4	674			
19	no	1097	**39**	offer	668			
20	rating	1087	**40**	ha	648			

Table A.3: List of Regex strings used for pattern matching and replacement during the processing phase

Description	Regex

Identify Currency
Matching: '$2500',
'2747.8 usd',
'3258.15usd',
'usd 2386.2'
'23341.86',
Not Matching: 2018,
7.15
	`(((\$)|(usd)|(usd))()?`
	`(\d\{3,7\}(\,\d\{3\})*|(\d\{3,7))`
	`(\.\d\{1,2\})?)|((\d\{3,7\}(\,\d\{3\})*|`
	`(\d\{3,7))(\.\d\{1,2\})?((usd)|( usd)))|`
	`((\d\{3,7\})(\.\d\{2\}))|(\d\{4,10\})`

Identify Full Stop Matching: 'Thanks.', 'Hi...'. Not matching: 7.15	`((?<=\.)|(?<=[a-z]))(\.)`

Table A.4: List of intents and their relative weights based on the number of occurrences in the dataset

Intent	Weight
reject	76.33846153846154
hearmore	58.37647058823529
moreinfo	23.62857142857143
canthelpt	17.110344827586207
sorry	12.102439024390243
you_are_welcome	11.67529411764706
goodbye	7.297058823529412
request_alts	5.83764705882353
request_compare	5.393478260869565
confirm	3.816923076923077
negate	2.811331444759207
greeting	1.9420743639921723
affirm	1.9345029239766083
suggest	1.4279136690647483
thankyou	1.3338709677419356
no_result	0.8644599303135888
switch_frame	0.521218487394958
offer	0.42739018087855296
request	0.41838111298482294
inform	0.12045151110571671

Table A.5: List of primary package versions used when conducting the experiment

Name	Version
Python	3.6.4
NLTK	3.2.5
matplotlib	2.2.2
python-dateutil	2.7.0
scikit-learn	0.19.1
scipy	1.0.0
imbalanced-learn	0.3.3
keras	2.1.5
tensorflow	1.7.0
pandas_ml	0.5.0
scipy	1.0.0