University of Cape Town

Word Sense Disambiguation in the domain of Sentiment Analysis through Deep Learning

Author:
Vedanth Baiju

Supervisors:
Dr Şebnem Er
Dr Emmanuel Dufourq

Minor dissertation presented in partial fulfilment for the degree of Master of Science in the Department of Statistical Sciences

August 7, 2022

Abstract

Sentiment analysis forms part of a major component of Natural Language Processing (NLP), even though continuous improvements in NLP are being made, word disambiguation remains a complex problem within the domain of sentiment analysis (Navigli, 2009). Word Sense Disambiguation (WSD) is a problem that deals with identifying the correct sense of ambiguous words in a sentence. As such, various words can have multiple meanings depending on the context in which they are used. Although advances in deep learning continue to rise within the NLP domain, WSD is still a task in which deep learning is yet to be fully explored. Whilst there does exist research within WSD as a whole, there is limited research for WSD conducted within the domain of sentiment analysis (Seifollahi and Shajari, 2019). The proposed research explores the task of WSD in the domain of sentiment analysis through recent advances in deep neural networks with a specific focus on 1D Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM) algorithms. Sentiments expressed in text sourced from the Amazon product reviews data were analysed using 1D CNN and LSTM deep learning algorithms. The Amazon product reviews data is segmented according to the type of product category which is essentially a context category. The effectiveness of each algorithm was evaluated from a statistical performance and efficiency perspective. It was found that the inclusion of context as a model input, improves the model out of sample performance as compared to a model without context as an input. In addition to this, it was observed that including more context categories as an input had improved the out of sample performance for both 1D CNN and LSTM algorithms. Furthermore, the 1D CNN exhibited superior performance over the LSTM model from a statistical and efficiency stand-point. Given that there has not been a considerable amount of research which explores the application of deep learning to solving the problem of WSD within sentiment analysis, the findings of this research will aid in providing a base-level of knowledge on future potential exploration and applications for WSD relating to sentiment analysis.

Keywords: Convolutional Neural Networks, Deep Learning, GloVe, Long Short Term Memory, Natural Language Processing, Sentiment Analysis, Word Embeddings, Word Sense Disambiguation

Plagiarism Declaration

I, Vedanth Baiju, declare that this thesis titled, "Word Sense Disambiguation in the domain of Sentiment Analysis through Deep Learning" and the work presented in it are my own. I confirm that:

- It is my understanding and acknowledgement that plagiarism is wrong.
- When I quote another author's work, the source is always cited and referenced according to the UCT-Harvard style.
- Anyone who attempts to pass off my work as their own is not allowed to, nor will I permit anyone to do so going forward.
- By certifying that this is my own work, I acknowledge that it is wrong to copy someone else's research in part or in totality.

Signed:

Signed by candidate

Date:

August 7, 2022

Acknowledgements

I would like to express my sincere and deep gratitude to my supervisors Dr Şebnem Er and Dr Emmanuel Dufourq. The innovative technical ideas, deep level of engagement and prudent advice proved to be an integral part of the research journey. Thank you for the open door policy and invaluable suggestions that widened my thinking both from a research and industry perspective. You have made the journey much more pleasant than I had anticipated.

A special thanks to Dr Lisa October for her level patience and understanding. Thank you for keeping me level-headed, offering a second ear to me and for all the unconditional support. Your guidance has sharpened me and allowed me to prosper on a positive trajectory both from a research and life perspective. I am forever grateful.

A deep level of thanks to my parents, who I don't thank enough. Thank you for always listening, no matter what the topic may be. Thank you for your life guidance from the physical realm as well as the metaphysical world. I am truly grateful and blessed for all the love and immense support that you have bestowed onto me. I will continue to make you proud in this life and the next.

Finally, to my family and friends who exhibited a great deal of understanding with the limited time I could offer to them while pursuing the Masters journey. I am looking forward to narrowing the time lost.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Problem Definition

Customer feedback provides a large level of guidance to organizations as it influences decision making as well as promotes innovations to existing products and service offerings. Collecting and acting upon customer feedback is imperative for any organization wanting to improve client retention as well as holding a competitive advantage within a particular industry.

Due to the rise in digital transformation, sentiment analysis is emerging as a viable tool for any business when it comes to analysing and understanding customer feedback. Sentiment analysis is a text analytics technique that measures the attitude of a customer towards aspects of a product or service offering, which is typically described in text. If customers are able to channel feedback in the form of text, sentiment analysis would potentially be able to significantly categorize responses into related sentiments.

Sentiment analysis forms part of a major component of Natural Language Processing (NLP), even though continuous improvements in NLP are being made, Word Sense Disambiguation (WSD) remains a complex problem in NLP and sentiment analysis. WSD is a problem that deals with identifying the correct sense of ambiguous words in a given sentence. As such, the context in which words are used can significantly influence their meaning.

Although sentiment analysis of textual content is valuable in the domain of NLP, there is a lack of a process for dealing with the ambiguity of words and WSD in the realm of sentiment analysis. As such, this raises questions on whether the sentiment expressed by a given context is correctly identified. Many studies in NLP focus on

WSD in isolation; however there is a weak link between WSD and sentiment analysis (Seifollahi and Shajari, 2019).

A variety of techniques exists that aim to disambiguate the sense of words. These methods include; dictionary-based methods in which lexicons are used, machine learning techniques that use a trained classifier to disambiguate the sense of each word from a dataset, and unsupervised methods that perform clustering on the occurrence of words that cluster.

Even though these methods do prove to have some level of effectiveness, they exhibit two major drawbacks (Da Silva et al., 2014):

- Word ordering is not considered, which is critical in capturing context.
- They rely heavily on language specific hand-crafted features for effective performance.

In contrast to the above mentioned methods, deep learning has allowed for a wide-range of applications to NLP problems and has showcased a level of effectiveness within this domain (Nithyanandan and Raseek, 2019). Although advances in deep learning continue to rise within the NLP domain, WSD is still a task in which deep learning is yet to be fully explored. The aim of this research is therefore to explore the task of WSD in sentiment analysis through recent advances in deep learning on methods such as 1D Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM).

1.2 Research Objectives

The overarching aim of this research can be described as: to explore the task of WSD in sentiment analysis through recent advances in deep neural networks.

The objectives of this research are therefore to:

- Assess the effect of context as input to the deep learning models in terms of WSD and the sentiment classification on text input data.
- Investigate the impact of hyper-parameter tuning on the out of sample performance of the deep learning models.
- Explore the efficiency and statistical performance of the deep learning methods in terms of a cost to benefit ratio.
- Assess the effectiveness of the addition of more context categories on the out of sample performance of the deep learning models.

1.3 Value of Research and Scope

This research will aim to provide concrete findings on the effectiveness of deep learning on WSD for sentiment analysis applied through the usage of deep learning methods such as CNN and LSTM. This will in turn provide a foundational knowledge base for future researchers to use for applications within this domain. This result will aim to meet the objectives of this research and further contribute to the body of knowledge given that there exists limited research within this domain.

From an industry perspective, this research could provide organizations with the ability to enhance the customer relationship management process within a variety of sectors, allowing organizations to better understand a client's attitude towards a particular product or service in a seamless and automated fashion. Figure 1.1 provides a diagrammatic illustration of the focus and fit of this research in sentiment analysis and WSD by using a sentence example from Liu (2012).

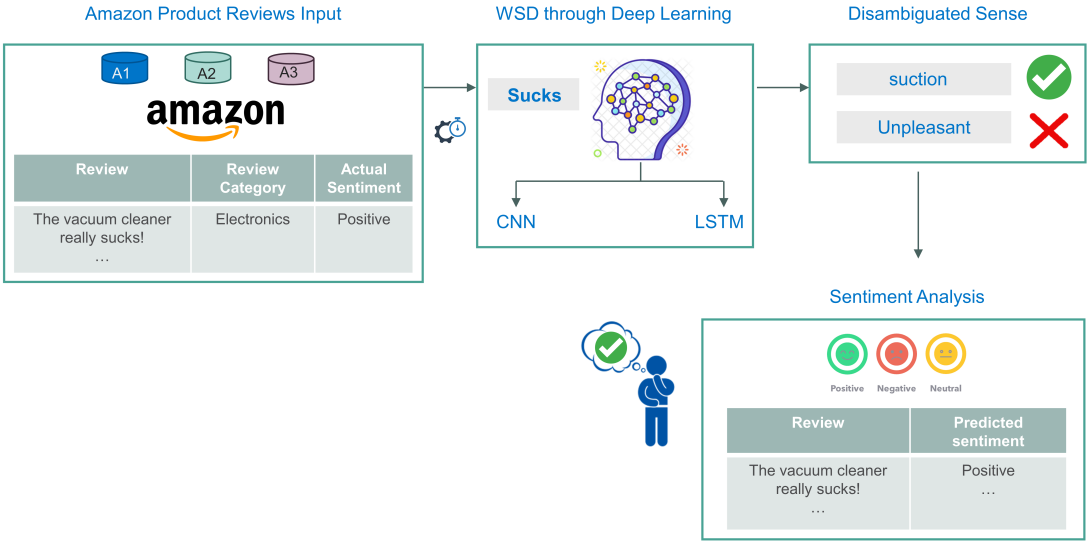

Figure 1.1: Scope of this research

Based on Figure 1.1 a customer review: "The vacuum cleaner sucks!" within the domain of electronics has an associated positive sentiment. This does seem strange at a first glance, however the word 'sucks' poses the ambiguity as it could be referred to as negative or positive. The negative implication would be associated with an unpleasant feeling. Conversely, the positive implication of the word could refer to the suction power of the vacuum cleaner. The aim of the deep learning algorithms is to disambiguate this sentence and predict the correct sentiment.

1.4 Thesis Layout

This section presents the layout of this research and briefly describes the contents of each proceeding chapter.

Chapter 2 - Literature Review

This chapter provides a detailed literature review of both focus areas; sentiment analysis and WSD. This chapter starts with a focus on sentiment analysis as a research area in NLP as well as the approaches and applications associated with it and finally explores WSD broadly and the overall methods applied to WSD as well as the link between sentiment analysis and WSD.

Chapter 3 - Introduction to Deep Learning

This chapter provides a deep dive into the overall dynamics of deep learning by first examining the fundamentals of neural networks and then moving into the broad scale workings of deep learning as well as the underlying CNN, Recurrent Neural Networks (RNN) and LSTM architectures that form the basis of these desired methods applied for solving WSD in the domain of sentiment analysis.

Chapter 4 - Methodology

This chapter explains the methodological components for this research in order to explore and evaluate the effectiveness of deep learning methods on WSD in sentiment analysis. Firstly, the data pre-processing component of the Amazon product review dataset is detailed. The next focus will be the dynamics of the deep learning algorithms that will be used for obtaining the disambiguated sense of words and lastly, the evaluation metrics in order to assess and potentially prove the value of deep learning for WSD in the sentiment analysis domain.

Chapter 5 - Results and Discussion

This chapter presents the results attained and discusses the findings attained from the output of the deep learning methods. This allows for the assessment of the effectiveness of the deep learning methods in disambiguating the correct sense of a word given its context.

Chapter 6 - Conclusion and Recommendations

This chapter summarizes the findings which were observed throughout the experiments conducted in the previous chapters. The research objectives presented in Section 1.2 are highlighted and further discussed how these have been met. Lastly, recommendations are provided for future research.

Chapter 2

Literature Review

This chapter provides a detailed literature review of both of the research focus areas; sentiment analysis and WSD. Section 2.1 focuses on sentiment analysis as a research area in NLP as well as the approaches and applications associated with it. Section 2.2 provides an overview of WSD and the overall link between sentiment analysis and WSD.

2.1 Sentiment Analysis

The purpose of this section is to present a review of literature relating to sentiment analysis and the wide-spread applications of this field of study. Sentiment analysis and opinion mining are the field of studies which focuses on the analysis of the opinions, attitudes, sentiments and overall emotions from language (Liu, 2012). Traditionally, fine-grained sentiment analysis is concerned with the polarity of opinions. This simply translates to understanding whether the opinion expressed is of a positive, negative or neutral attitude towards an entity (Mäntylä et al., 2018). In most research applications the classification of sentiment is binary, such that in which sentiments are classified into either a negative or positive category (Pang et al., 2002). For example, given a customer service review, the sentiment classification system would be able to determine whether the customer review expresses a positive or negative overall opinion about the customer service experienced. The rationale for the exclusion of the neutral category is that the related text is less informative and thus it is filtered out in order to improve the binary classification (Pang et al., 2002).

Although linguistics and NLP do come a long-way in history, prior to the year 2000 a minimal amount of research had been conducted on people's opinions and sentiments (Liu, 2012). Since then, sentiment analysis is one of the most actively researched areas in NLP. There are a number of reasons for this. Firstly, the application of sentiment analysis extends to a wide variety of domains. Secondly, it offers many challenging research problems in the field of NLP. Lastly, with the growth and expansion of social media, large volumes of opinionated data is easily accessible. Hence, the inception of the growth of sentiment analysis research and the growth of social media data coincide with each other. As such, sentiment analysis is not only impactful to NLP but to areas of political science, social sciences, management sciences and economics as these are fields which deal with large levels of opinion from individuals.

2.1.1 Levels of Sentiment Analysis

There exist three levels of granularity in which sentiment analysis can be performed: the document level, sentence level and aspect level (Tang et al., 2015). These are discussed in detail below.

Document level

Document level is focused on determining the sentiment of a complete paragraph or a document. This sentiment analysis level is based on the assumption that a document only contains opinionated text about a single entity. Hence, this level does not support documents in which multiple entities are compared. In addition to binary classification, the classification of a sentiment can be handled as a regression problem. The regression application involves assigning a rating factor in a defined scale, an example of this is the rating scale of 1-5 stars for movie reviews. Alternatively, a multi-class problem can be modelled accordingly.

Sentence level

Sentence level sentiment classification focuses on determining the sentiment of a single sentence. Both sentence and document level analysis share a common main underlying assumption in that each sentence only encompasses a sentiment relating to a single entity. This level of classification determines the sentiment or opinion polarity of each sentence i.e., positive, negative, or neutral. The neutral opinion typically refers to no opinion. Sentence level analysis has a close relation to subjectivity classification (Wiebe, 1990), which differentiates between sentences in which subjective views expressed and opinions from those which express factual information.

Sentiment can be inferred from a sentence using polarity classification and subjectivity classification. Existing deep learning techniques which are discussed in detail in Chapter 3, focus on predicting the polarity of a sentence into positive and negative categories. Given that sentences are shorter than a document, the semantic features obtained using a part of speech tagger and lexicons are useful for the classification of sentence level sentiment.

Aspect level

Document and sentence level sentiment analysis fails to provide the target of an opinion i.e., they do not provide information of what each opinion is about. For example, a positive sentence: "I enjoy Microsoft's Windows 10", only knowing that the sentence is positive is of little value unless we know that the positive opinion relates to Microsoft's Windows 10. It is a misconception that if a sentence is classified to be positive, then all components within that sentence can take on a positive opinion. This is untrue, due to the fact that sentences can have multiple opinions, for example, "Microsoft is doing very well in the poor state of the South African economy". Even though the sentence is positive about Microsoft, it is negative about the South African economy, as such, positive or negative classifications don't seem to make much sense in this sentence. In order to obtain fine-grained results, aspect-level is required.

In this level, sentiments of users expressed towards features (aspects) of entities such as movies and restaurants are extracted. An aspect and polarity pair can be found from a given text using this method. Documents in this level are assumed to be composed of a single entity. Aspect level analysis is key for real-world applications and as a result many sentiment analysis systems are based on this level of analysis from an industry perspective.

Given the levels of sentiment analysis, there are different types of approaches that have been applied in research to sentiment analysis problems. Section 2.1.2 explores the literature associated with the three main approaches to sentiment analysis; lexicon-based, machine learning based and hybrid based approaches.

2.1.2 Approaches to Sentiment Analysis

There are currently three approaches which attempt to address sentiment analysis related problems (Bhavitha et al., 2017): lexicon-based techniques, machine learning based techniques and hybrid approaches. Although machine learning based techniques are covered at a high-level within this section, they will be discussed in detail in Chapter 3 so as to provide insight into the fundamentals of this approach.

Lexicon-based techniques

One of the first traditional methods used for sentiment analysis was lexicon-based techniques. There exists a pre-defined list of words associated with a sentiment which are referred to as sentiment words. These sentiment words and phrases are instrumental to sentiment analysis. Sentiment lexicons encompass a list of sentiment words and phrases and are sometimes also referred to as an opinion lexicon. Dictionary and corpus-based approaches form the basis of lexicon-based techniques (Salas-Zárate et al., 2017). Dictionary-based sentiment classification makes use of a dictionary of terms, such as SentiWordNet (Esuli and Sebastiani, 2006) and WordNet (Miller, 1995) among others.

Corpus-based sentiment analysis focuses on the statistical analysis of the contents of a collection of documents through the usage of k-nearest neighbors based techniques (Huq et al., 2017), hidden Markov models (Soni and Sharaff, 2015) and conditional random field (Pinto et al., 2003) among others. As such, this removes the dependency on a pre-defined dictionary. Over the past several years, researchers have developed numerous algorithms for compiling such lexicons (Tang et al., 2015). Although sentiment words and phrases are important, they are not enough to accurately detect sentiment. The problem is far more complex in nature. The following issues are highlighted (Liu, 2012):

- Different application domains and sentence contexts determine the orientation of sentiment words. A sentiment word that is considered to be positive can have the opposite polarity in a given context. For instance, the word 'suck' is typically associated with a negative sentiment, for example, "This camera sucks", but it can also imply positive sentiment, for example, "This vacuum cleaner really sucks." In this case, the suction power of the vacuum is being referred to.
- It is possible for a sentence to contain sentiment words without expressing any sentiment.
- It's very difficult to deal with sarcastic sentences, whether they contain sentiment words or not. For example, "What a great car! It stopped working in two days." It is evident that there exists a level of mockery about the car, by providing an initial positive compliment, which is the opposite of the intended meaning, given that the car had actually stopped working at an early stage.
- In sentences without sentiment words, the authors' opinions or sentiments can be implied either positively or negatively.

Machine learning based techniques

Techniques for sentiment analysis based on machine learning can be classified into two groups: traditional models and deep learning models. The traditional models encompass classical machine learning techniques most commonly used by researchers, which include; Support Vector Machines (SVM) (Firmino Alves et al., 2014), Naïve Bayes classifiers (Suppala and Rao, 2019), N-gram approaches and maximum entropy classifiers (Mehra et al., 2002).

Lexical features, sentiment lexicon-based features and parts of speech are the main inputs into these algorithms. In addition to this, machine learning methods are able to eliminate overlapping and irrelevant features through the usage of suitable feature selection methods. However, the accuracy of these algorithms have a dependency on the features selected.

Deep learning algorithms have showcased flexibility and excellent performance in NLP applications including sentiment analysis across a wide-spread of datasets (Collobert, 2011). The benefit of these models is that there is no requirement for the input of pre-defined features hand-picked by an engineer, since these models are able to learn sophisticated features from the dataset in an independent fashion. A neural network is used to extract features from a high-dimensional vector space representing words (Mikolov et al., 2013a). As a result, these models have the ability of mapping words with similar semantic and syntactic properties to nearby locations in their coordinate system, which is its own unique way of understanding the meaning of words. These characteristics and capabilities make deep learning methods attractive and an almost natural fit for dealing with sentiment analysis.

Considering that the focus of this research is to utilize deep learning approaches to solve WSD in the domain of sentiment analysis, a review of literature for the application of deep learning methods within the domain of sentiment analysis will be discussed in the next subsection. Furthermore, Chapter 3 provides detailed insight into the concepts and overall components of deep learning.

2.1.3 Applications of Deep Learning for Sentiment Analysis

Sentiment analysis is expected to expand into approaches that can scale, are highly adaptable to source variation, have a high efficiency rate, and are very accurate to maximize the benefits of value mining for users (Sharef et al., 2016).

Recent research has demonstrated that deep learning models (such as CNNs and RNNs) can enhance the efficiency of sentiment analysis problems. This section reviews state-of-the-art approaches to sentiment analysis based on deep learning. Tang et al. (2015) introduced successful deep learning approaches for several sentiment analyses, such as opinion extraction, sentiment classification, and the learning of word embeddings. Zhang and Zheng (2016) conducted a machine learning comparative analysis between the Support Vector Machine (SVM) and Extreme Learning Machine (ELM) approaches. Within this analysis, parts of speech were utilized as a text feature and furthermore Term Frequency-Inverse Document Frequency (TF-IDF) was applied in the weight calculation of words. Both the SVM and ELM with kernels were adopted to analyse the text emotion tendentiousness. The outcome from this analysis was positive in favour of the ELM with kernels as it obtained a better classification result in a relatively short space of time.

Ain et al. (2017) provides a comparative review of recent studies relating to the implementation of deep learning based techniques such as CNN, RNN and LSTM for solving various sentiment analysis tasks such as product review analysis, sentiment classification, text and visual analysis as well as cross lingual tasks among others. It was concluded that sentiment analysis can be conducted in a more efficient and accurate way through the usage of deep learning methods. In addition to this, deep neural networks are better than SVMs and shallow neural networks because they have more hidden layers as compared to shallow neural networks that have one or two hidden layers. In a similar light, Singhal and Bhattacharyya (2016) provides a comparative analysis between deep learning methods (variants of CNN, RNN and LSTM) and traditional machine learning methods such as Naïve Bayes and SVM. This research found that the performance of the deep learning models (especially CNN and LSTM variants) on most of the datasets utilized proved to be of superior strength in terms of the overall classification accuracy.

The application of deep learning based sentiment analysis has been extended to different domains, including areas of finance (Sohangir et al., 2018), recommender systems for cloud services (Preethi et al., 2017), weather related tweets (Qian et al., 2018), trip-advisors (Pham and Le, 2018) and movie reviews (Ain et al., 2017). Jeong et al. (2019) identified product development opportunities through the combination of topic modelling and the results of a sentiment analysis which was conducted using customer related social media dataset. This method has proven to be beneficial in

terms of analysing the existing and changing needs of a customer in evolving product environments. Pham and Le (2018) proposed a novel multi-layer architecture to analyse travel reviews and determine effective sentiments for five travel components, including location, cleanliness, service, value and room. The experimental results generated were relatively good as the proposed model outperformed other popular methods with much improvement.

Gupta et al. (2016) combines sentiment and semantic features in an LSTM model based on the detection of emotions as a multi-class classification problem. The approach presented here outperformed CNN and LSTM baselines, in addition to other machine learning baselines. From a health domain perspective, Salas-Zárate et al. (2017) applied an ontology-based, aspect-level sentiment analysis method to tweets about diabetes.

In the realm of tweets utilization, polarity-based sentiment deep learning analyses were conducted by Abid et al. (2019); Vateekul and Koomsubha (2016); Malik and Kumar (2018); Ramadhani and Goo (2017). Within these research papers, the authors effectively described how deep learning models were used to increase the accuracy and overall efficiency of their respective sentiment analysis.

Most of these applications are centred on the content written in English, but there are a variety of studies that manage tweets in other languages, such as Persian (Roshanfekr et al., 2017), Spanish (Paredes-Valverde et al., 2017) and Thai (Vateekul and Koomsubha, 2016). This further showcases the flexibility of deep learning methods when applied to the problem of sentiment analysis.

As much as sentiment analysis has cemented its place in a wide variety of applications and furthermore research in deep learning has proven to be highly effective in improving accuracy and efficiency in a plethora of analyses, there does still exist the problem of lexical ambiguity, be it syntactic or semantic. The problem is that words are often associated with multiple meanings, these meanings could be fairly similar or sometimes completely different. Only by analysing the context can a word's meaning be determined. The next section describes WSD and focuses on the overall applications and dynamics relating to this problem.

2.2 Word Sense Disambiguation

As discussed previously, sentiment analysis has a variety of applications in NLP and various methods are constantly being applied and improved upon in order to better the outcomes and results within this domain. However, a lack of a process for handling ambiguity within sentiment analysis makes it difficult to ascertain whether the correct sentiment is identified within a given context.

This is where the concept of disambiguation of particular words in a given context becomes key. As much as there are many studies that exist within NLP for the disambiguation of words in isolation, there has been a weak link between WSD and sentiment analysis. The focal point of this section is to describe WSD and its related applications on a general scale as well as a specific angle towards sentiment analysis.

As part of NLP, WSD aims at determining the correct sense of a given word in a given context. An example of an ambiguous case comes from the classical example used in a wide variety of literature: "bank" may have different senses such as "financial institution", "river side" or "reservoir". This indicates that the setting of a word in a particular sentence governs the overall importance of this word.

This problem has been a long-standing one in the field of NLP with research work having started during the late 1940s (Agirre and Edmonds, 2007). In line with Chen et al. (2014), words do not have clearly defined limits between their meanings of sense. As such, there is a major issue that exists with regard to establishing which sense of the word is intended in a given setting. WSD is important for NLP due to richness and ambiguity that exists in natural language itself. Figure 2.1 provides a basic idea of how a potential WSD system should work.

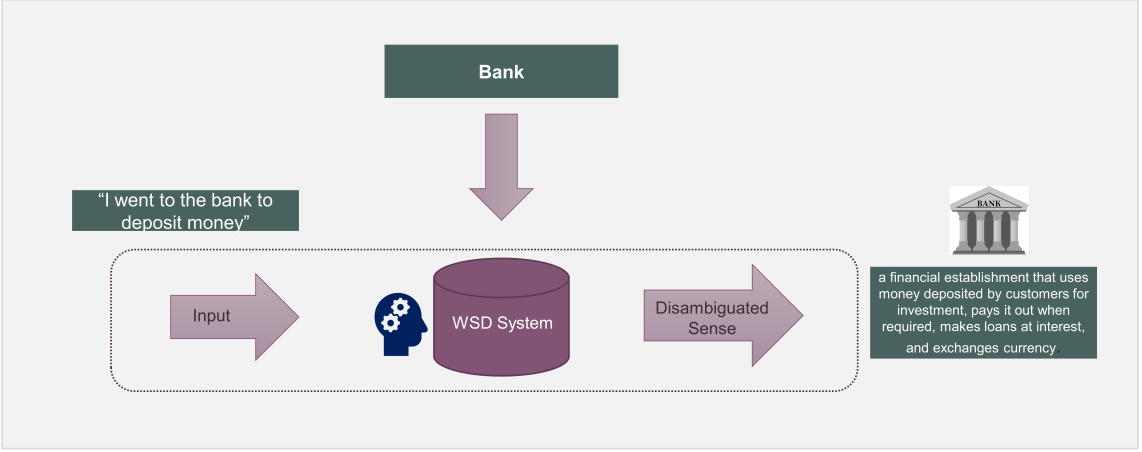

Figure 2.1: Basic idea of a WSD system - (adapted from Nithyanandan and Raseek (2019))

As discussed earlier the word "bank" can offer a level of ambiguity without a certain context or domain. Based on Figure 2.1 the task is for the WSD system to be able to correctly associate the word to the correct domain. In this case scenario the word bank should be classified as a financial institution as opposed to a river bank.

2.2.1 Applications of WSD

The main field for the application of WSD is machine translation, but it is utilized in a wide variety of linguistic NLP applications. This section looks at a high-level overview of various application fields of WSD. However, given that the aim of this research is to apply WSD to sentiment analysis, a deeper dive into this specific application will be the main focus of this section.

Machine translation:

WSD is required for lexical choice in machine translation (Carpuat and Wu, 2005), (Chan et al., 2007) for words that have diverse interpretations for distinctive senses which have potential ambiguity in a given domain. For example, consider the following English sentences: "I was attacked by a criminal and as a result injured my arm", "It is best practice to arm yourself with a solid educational foundation" – the word "arm" carries different meanings which is a major hurdle in the process of language translation.

Information retrieval:

One of the most vital issues in information retrieval is to resolve ambiguity in a query (Sanderson, 1994; Stokoe et al., 2003). An example of this would be the word "depression" in a particular query that could vary in terms of its meaning such as; an illness or economic well-being. As such, the essential component is to obtain the sense of an ambiguous word in a particular question, before finding the answer to the question. WSD has showcased benefits in improving cross-lingual information retrieval and document classification (Vossen et al., 2006; Bloehdorn and Hotho, 2004; Clough and Stevenson, 2004).

Information extraction and text mining:

WSD is used in information extraction and text mining as it enables a more accurate analysis to be conducted on text data. On a general scale, senses and synonyms play an integral role in semantic analysis and as result semantic analysis can prove to be useful in information extraction (Pal and Saha, 2015).

Speech Recognition:

Computers with sound input sources, such as a microphone, will perceive human speech using speech recognition technologies. It is the method of translating an acoustic signal to a series of words using a microphone or a telephone. To obtain correct outcomes, WSD is used in speech recognition.

Sentiment Analysis:

As highlighted earlier, one of the issues experienced in sentiment analysis is WSD. Generally, WSD has been defined to be used when identifying the intended meaning of a word in a sentence. However, within the sentiment analysis domain, WSD refers to the process of determining the sentiment orientation sense of a word in a sentence in a given context. The three main senses that a word can take on are; positive, negative or neutral. The context refers to the text in which a word appears and which helps ascertain its meaning. So essentially the sense of a word depends on the other words around a particular word within a sentence or passage. As such, there exist varying words according to the context. Consider the following example:

- Positive sentiment – "My purse is small enough that it fits easily in my jeans pocket."
- Positive sentiment - "I have a very small mobile phone which I can easily carry around."
- Negative sentiment - "The hotel room is very small."

From the above example, it is evident that the context defines the type of sentiment sense that the word 'small' relates to. In the purse and mobile domain, the word 'small' is expressed in a positive sense, conversely in the hotel domain 'small' expresses negative opinion. There are many descriptive words that experience a similar trend. This further validates the fact that most of the errors within sentiment analysis are due to sense in a given context. This effectively demonstrates the importance of WSD is in accurately determining the polarity of a sentence.

Research that handles the ambiguity problems that exist in order to improve the performance of sentiment analysis is still very limited. Akkaya et al. (2009) showed that contextual opinion analysis can be improved through Subjectivity Word Sense Disambiguation (SWSD). The application of SWSD to contextual polarity classification resulted in an improvement in accuracy of 3% over the benchmark classifier calculated on the SenMPQA dataset (Wilson et al., 2005). Rentoumi et al. (2009) demonstrated that WSD provides added value in polarity classification for sentences that encompass figurative expressions.

Khan et al. (2013) focused on applying a WSD method to sentence level sentiment analysis on different domain datasets (Twitter, Pakistan election comments, airline reviews). This method involved the creation of a matrix map by blending the sentiment scores extracted from SentiWordNet (Baccianella et al., 2010) for desired target words. The method applied in this research attained an average out of sample accuracy of 90.71% across all the input domain datasets.

Sumanth and Inkpen (2015) showed that the use of WSD alone has improved results in sentiment analysis that outperform results without incorporating WSD. Babelfy was used for WSD in the system and a random forest decision tree for the sentiment polarity classification. Furthermore, this study utilized both tweets from Twitter and text from Short Messaging Services obtained via the Semantic Evaluation Exercises (Nakov et al., 2013) and yielded average F1 scores that exceeded all baseline comparable datasets by 10%.

Jose and Chooralil (2015) considered predicting the outcome of the election through the analysis of Twitter data. SentiWordNet was utilized and WSD was further added to select the most appropriate synset (groupings of synonymous words that express the same concept). More specifically SentiWordNet was used as a classifier to determine the polarity of a word.

Pamungkas and Putri (2017) explored several path-based methodologies to perform the WSD for lexicon-based sentiment analysis. WordNet was utilized to extract the sense candidate of each word whilst the path-based methodology aided in determining the relatedness between words. A comparison was made using the F1 scores between a system that did not leverage of WSD and one that did, and the results were indicative that WSD was able to improve the performance of the sentiment analysis process.

Seifollahi and Shajari (2019) proposed a system for FOREX market prediction, in which the directional movement of currency pairs are predicted. This system exploits WSD in sentiment analysis of news headlines in order to make the predictions more accurate. WordNet was utilized as part of the parts of speech annotation relating to particular news headline words. Tests were carried out with the same dataset utilized by Nassirtoussi et al. (2014) and ultimately proved that the proposed system outperforms one of the leading systems proposed for market prediction. This was due to the fact that the proposed method had attained an out of sample accuracy of 91.67% which was an improvement over the benchmark model which produced an accuracy of 83.33%.

Given the limited research explorations within WSD specifically for sentiment analysis, this provides a motivating factor for the aim of this research i.e., an investigation for solving WSD in the domain of sentiment analysis. The next section explores the variety of approaches to WSD. This will provide a detailed view and guided approach on the success of various approaches to WSD.

2.2.2 Approaches to WSD

This section aims to provide an overview of the varying methods applied to WSD as well as a comparative contrast between these methods. Although this section provides an overview of different types of approaches to WSD, the main focus of the review would be approaches in deep learning as this feeds into the overall focus of this research and further provides guidance in the selection of the appropriate algorithms and architectures to leverage of in the application to WSD in sentiment analysis. WSD is classified into the following families of methods:

- Knowledge-based approaches.
- Traditional machine learning based approaches.
- Deep learning approaches.

The concept of knowledge-based approaches is to extensively utilize knowledge sources to help determine the senses of words in a particular context. One could think of this as tapping into a repository of learning assets which include word references, thesauri, ontologies and collocations; to name a few. According to (Miller et al., 2012), knowledge-based methods may utilize sentence structure rules for the disambiguation component. WordNet (Miller, 1995) is the most widely used machine readable dictionaries within this field of research. The attractive aspect of knowledge-based approaches is that it is simple in terms of the required input, in that it only requires a lexical resource, such as a dictionary or a computational lexicon. Some of the main knowledge-based methodologies include; Lesk's Algorithm (Lesk, 1986), WSD using conceptual density (Agirre and Rigau, 1995) and Walker's algorithm (Kaur, 2014).

With the widespread growth of big data and enhancements in technologies, the NLP community has experienced a shift from manually deployed systems to more automated classification algorithms. This dramatic increase of interest towards machine learning or deep learning is evident in the large amount of supervised learning approaches applied to the problem of WSD. Traditional machine learning based approaches relate to algorithms that are not independent learners which is in contrast to deep learning that learns features independently. A classifier forms the core of machine learning techniques in that it learns features and senses are assigned to unseen data. The word to be disambiguated forms the initial input and is referred to as the target word. The text in which the target is embedded is known as the context. The feature value is essentially the frequency of the occurrence of a word in the surrounding region of the target word. The surrounding region is usually a fixed window, and has the target word at its centre. Machine learning based approaches can be broken down into: supervised learning techniques, unsupervised learning techniques and semi-supervised learning techniques.

Supervised learning techniques utilizes artificial intelligence for inducing a word expert called the classifier, which has manually sense-annotated data as the input. The classifier focuses on a single word and performs the task of classification with the aim of assigning the appropriate sense to each instance of that word. The training dataset provides the basis of learning for the classifier. As the classifier learns from a training dataset containing datapoints relating to a manually tagged target word, that has a sense tagged to it from a particular sense database or sense reference dictionary. To better illustrate this, consider the learning process of a child who has not been exposed to reading and writing. The child is being taught by parents at home and the teacher at school. Their each and every action is being supervised by the teacher. To test the child's learning capability an exam is given which is unseen to the child prior to the examination session. Similarly, a set of labelled instances sampled from the same distribution as the test dataset is used to train the WSD system (Dixit et al., 2015).

Supervised methods have achieved superior results in WSD related tasks, as they perform better than knowledge-based methods, however require significant amounts of data. There exists a wide variety of algorithms within supervised learning. Naïve Bayesian classifiers and decision trees have yielded positive results for WSD applications. Naïve Bayesian classifiers are associated with good results and performance of WSD. This algorithm is based on the application of Bayes theorem (Triola, 2010). It is based on the rationale that choosing the best meaning for the input text vector results in the selection of the most likely meaning of the word in its context (Ting et al., 2011). The main benefit of this method is that it provides simplicity from an implementation perspective, requires less training data and can output probabilistic predictions. As much as these benefits seem attractive, a major drawback of this method relates to the paucity of data. The potential value for a particular feature needs to be estimated through an iterative method.

Decision trees are applied to WSD through the selection of a desired concept using a Yes-No tree. A decision tree can be thought of as a binary tree in which every internal node is categorized by an attribute, as a result each branch is classified with either a 1 or 0. The highest-gain feature is used to divide the training examples, and the process is repeated with the aim of inducing good decision trees. Lee and Ng (2002) applied decision trees and other approaches using SENSEVAL-2 and SENSEVAL-1 data on the English language. It was found that SVMs performed the best without feature selection. Albayaty and Joshi (2014) applied decision trees using SENSEVAL-3 data on English language. It was found that accurate results were only attained by using a few words. As such, this lead to a low overall accuracy (45.14%). Although decision trees are easier to understand and interpret if the tree is small, it is associated with instability. The optimal decision tree structure can

be significantly changed, by adjusting the underlying data. Even though supervised learning methods have been widely used within WSD, the major drawback lies in their requirement or dependency on manually annotated input data. However, the creation of manually annotated data is expensive and time-consuming. As such, this is not feasible for applications on a large scale (Artstein and Poesio, 2008).

In contrast to supervised approaches, the unsupervised approach does not require initial knowledge of sense information in huge scale assets for the disambiguation. In addition to this, long computing time and heavy computational power are not required (Zhou and Han, 2005). It is based on the fact that the sense of a particular word depends on its neighbouring words. Word senses are derived through the formation of clusters of the occurrences of words and the task is to classify the new occurrence to the derived clusters. This approach detects clusters as opposed to having sense labels tagged to text. Although this seems attractive, the unsupervised approach has its overall performance less than that of other methods (Chen et al., 2009). Similarly, Fulmari and Chandak (2013) indicate that the unsupervised approach has a worse-off performance than the supervised approach due to the dependency on less knowledge as well as due to the fact that the input data is unknown. Various methods utilized in unsupervised approaches are: context clustering, word clustering and co-occurrence graphs (Yarowsky, 1995) and (Sinha and Mihalcea, 2007).

Despite the advantages that unsupervised approaches have, such as not needing a sense inventory or sense annotated corpa, they pose challenges from an implementation perspective. Furthermore, the out of sample performance has shown to be inferior in comparison to traditional supervised and deep learning methods (Nithyanandan and Raseek, 2019).

As shown earlier, traditional machine learning algorithms are not independent learners. Although deep learning is a branch of machine learning (detailed in Chapter 3), it does showcase superiority in the fact that it overcomes the learning dependency issues exhibited by machine learning algorithms. This is achieved through deep neural networks. Deep learning has been applied on a variety of NLP tasks and has proven to be very effective. WSD is still a task in which deep learning is yet to be fully explored. Deep learning can potentially be effective because:

- It uses sequence processing and considers the word order which is essential for understanding the context.
- The underlying models are independent in the sense that it does not require hand crafted features to be given as input since the features are learned automatically.

The performance of neural network models has been improved by increasing the number of layers of the network (Collobert et al., 2011). Deep learning techniques have been successfully applied on NLP tasks which use sparse data, due to its ability in handling high dimensional sparse data and due to the significant advancements in the computational power of machines (Young et al., 2018).

Wiriyathammabhum et al. (2012) have applied a novel technique for WSD, which involves deep belief networks. This technique has shown positive results for smaller datasets when compared to the various state-of-the-art shallow learning algorithms. This further confirms the impact that deep learning has in WSD. The LSTM method has successfully mitigated the problem of a vanishing gradient using the memory cell. LSTMs allow for the linear structure of the language to be captured sequentially.

A variety of NLP tasks has seen performance improvements as a result of LSTMs. These tasks include: named entity tagging, tweet encoding and question answering (Young et al., 2018). However, the training of LSTM algorithms require large amounts of data.

Zvonarev and Bilyi (2019) evaluated the performance of three models; logistic regression, XGBoost classifier and CNNs to solve a binary classification problem within the stream of sentiment analysis relating to the Russian language Twitter data. Based on the study, it was found that CNN showed the best results among chosen models in terms of the predictive accuracy.

Kansara and Sawant (2020) discussed two paradigms; traditional classification approaches, which have been in use for the past few decades and the more recent approaches which leverage of deep learning algorithms. The traditional classification approaches consisted of logistic regression, Naïve Bayes and random forests whilst LSTM, CNN and a combination of CNN and LSTM were implemented as part of the deep learning suite. The implementation was conducted on multiple datasets and based on the results, deep learning algorithms showcased a level of superiority over traditional machine learning algorithms in terms of predictive accuracy. These studies demonstrate the superiority of deep learning over classical supervised machine learning techniques and cements the motivation for the utilization of deep learning techniques as part of this research.

Within deep learning, the depth of actual learning is dependent on the number of hidden layers between the input and output layers. The abstract features are extracted by the initial layers, and as the learning progresses the specific and meaningful information is extracted by the deeper layers. Word embeddings serve as input into deep learning algorithms. Pre-trained word vectors such as Word2Vec (Rong, 2014) and GloVe are used to convert the input text into embeddings. Gupta et al. (2016) have demonstrated that deep learning outperformed traditional supervised

machine learning methods in terms of predictive accuracy through the implementation of LSTM and window probability maximization methods which leverage off word embeddings.

Le et al. (2018a) explored the LSTM approach to WSD on the GigaWord, Sem-Cor and OMSTI open source datasets. The exploration of this approach showcased similar performance to Yuan et al. (2016), however with a differentiating factor in that less data was utilized. It was also shown that the addition of more unannotated training data does add value however it does suffer from diminishing returns. Furthermore, the LSTM model showed the ability to correctly identify meanings that were both unpopular and popular.

Pesaranghader et al. (2018) used deep bi-directional LSTM to learn the context from the text. Deep bi-directional LSTM is used in order to resolve the issue of one-word per LSTM approach having to create a single LSTM for each word. This research focused on the creation of deep networks of LSTMs in order to consider senses and context sequences. Wiriyathammabhum et al. (2012) use a Deep Belief Network (DBN) for disambiguating the sense. DBN uses Restricted Boltzmann Machine as a pre-training method to greedily train layer by layer. After which, a fine-tuning step was employed with the intention of improving discriminative power. The system was evaluated on SEMEVAL-1 dataset.

Pesaranghader et al. (2019) have proposed a deepBioWSD model which leverages off a single bidirectional LSTM network in which a sense prediction is made for any input ambiguous medical term. Their proposed model first focused on the computation of the Unified Medical Language System sense embeddings through the usage of their text definitions. This was done to initialize the network. After the network initialization process, the model was trained on all the available training data. This application was conducted on the MSH (Management Sciences for Health) dataset and compared across WSD algorithms. Macro and micro accuracies were used as evaluation metrics. The deepBioWSD achieved a state-of-the-art performance of 96.82% in terms of macro accuracy, thereby outperforming existing models for WSD within the biomedical domain.

This further showcases the flexibility of deep learning as part of a way to address the WSD problem within the biomedical space where large amounts of data is present in the form of unstructured narratives such as clinical reports and research articles. Furthermore, the deepBioWSD has less manual dependency by requiring less manually annotated data as the target and terms are jointly learned. This is favourable from a practical perspective as it allows for the deepBioWSD to be seamlessly deployable in real-time biomedical applications.

Kai and Wen (2016), proposed a CNN to tackle the WSD problem within the biomedical space. It was shown that CNN is an efficient method for WSD in the biomedical domain. This study utilized commonly ambiguous words from MSH-WSD corpus. This study demonstrated that the average accuracy achieved for the proposed method was 94.65%, which was a significant improvement over methods that were implemented previously. This result provides an indication on the superior efficiency of CNN for WSD within the biomedical domain.

Festag and Spreckelsen (2017) investigated WSD based on word embedding and recurrent convolutional neural networks. This study focused on terms mapped to multiple concepts of the Unified Medical Language System. The MSH WSD was used and multiple recurrent convolutional neural network pipelines were created. The results achieved were deemed to be excellent as 70% of the WSD pipelines achieved at least 90% accuracy, 40% achieved at least 98% accuracy. In addition to the successful results attained, there exists motivational ground for up-scaling the problem with improved availability of training data.

Joopudi et al. (2018) focused on a special case of WSD known as abbreviation sense disambiguation using clinical narratives from Cleveland Clinic as training data. SVMs and CNNs were applied to this niche problem and an effective comparison in terms of accuracy was made. It was found that the CNN models showcase superior performance compared to the SVMs in disambiguating abbreviations in clinical narratives. From a values perspective, the CNN yielded an improvement in accuracy by 1%-4% on all the input datasets compared to SVM. The CNN used within this research, consisted of a single convolutional layer, a max-pooling layer, a fully connected feed forward neural network layer followed by a fully connected softmax classifier.

Munot and Govilkar (2015) use RNNs to learn the context embeddings. Using this context embedding the sense of the word is identified. One-word per LSTM model is employed. For every word to be disambiguated an LSTM and classifier is trained. The LSTM learns the context of the word from the training data and using a softmax activation function the classifier predicts the probability distribution of the senses of that particular word.

Deep learning has the ability to extract features independently as opposed to traditional machine learning algorithms in which the applied features need to be identified by a domain expert first. Deep learning has the ability to learn and make decisions in an independent fashion as opposed to traditional machine learning algorithms which learn from input data and makes informed decisions based on the learnings it has made. In addition to this, Nithyanandan and Raseek (2019), explored a variety of deep learning models for the WSD problem. The models that were analysed were: BiLSTM, BiGRU, LSTM. A comparison was made between these models as well as across other approaches. Table 2.1 showcases a comparative view between approaches for WSD.

Table 2.1: Comparison of WSD approaches (adapted from Nithyanandan and Raseek (2019))

Approaches	**Advantages**	**Disadvantages**
Knowledge-Based	Higher precision is exhibited by these algorithms.	Given that these algorithms are overlap based. As such, overlap sparsity is a major drawback. Furthermore, performance is dependent on dictionary definitions.
Supervised	Are more superior than knowledge-based and unsupervised methods from an implementation perspective.	Performance is compromised for applications on languages which are resource scarce.
Deep Neural Networks	These are the best algorithms which gives the maximum accuracy.	These algorithms require large training data which may not be easily available.
Unsupervised	This approach is independent of sense annotated corpora and sense inventory as aiding inputs.	Poses challenges from an implementation perspective and exhibits inferior performance compared to supervised methods and deep neural networks.

Based on the above, it is evident that deep neural networks, in particular CNN and RNN based methods have previously shown success in NLP and thus this provides the motivation for the exploration of these architectures. The following chapter will discuss deep learning and its underlying mechanisms in detail.

Chapter 3

Introduction to Deep Learning

The aim of this research is to apply deep learning methods for solving WSD in the realm of sentiment analysis and as demonstrated in Chapter 2, deep learning has showcased relatively strong performance and wide-spread usage for the WSD and sentiment analysis. As such Section 3.1 provides a detailed insight into the fundamentals of neural networks. While Section 3.2 discusses the broad scale workings of deep learning as well as the underlying CNN, RNN and LSTM architectures that form the basis of these desired methods applied for solving WSD in the domain of sentiment analysis.

Before delving into the fundamentals of neural networks and deep learning it might be beneficial to illustrate the linkage between artificial intelligence, machine learning, deep learning and neural networks. Figure 3.1 illustrates the linkage in form of Russian nesting dolls. Each element is a component of the prior element.

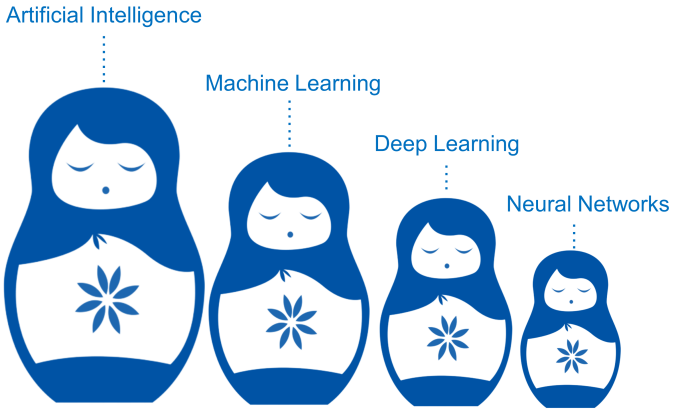

Figure 3.1: Artificial intelligence hierarchy

Based on Figure 3.1, we observe that machine learning is a sub-field of artificial intelligence. Deep learning is a sub-field of machine learning, with neural networks forming the nucleus of deep learning.

Hence before delving into the realm of deep learning, one must first touch on the fundamentals of neural networks as this provides the basis for the underlying architectures of deep learning algorithms. Although discussed very briefly in Chapter 2, the proceeding Section 3.1, focuses on a detailed view of the components and overall workings of neural networks.

3.1 Neural Networks

Neural networks, and more specifically, artificial neural networks share a level of similarity with the functioning of the brain in that both acquire the skills in processing data and finding solutions through training (Haykin, 2010). The processing of different pieces of information are handled by different parts of the human brain. It can be thought of that these parts of the brain are arranged in a hierarchical form or in layers. Based on this analogy, information enters the brain and each level of neurons process this input information and this processed information is then passed through to the next and more senior layer in a hierarchical fashion. For example, your brain may process the appetizing smell of French fries wafting from a local diner in multiple stages:

- Data Input - "I smell French fries."
- Thought - "I love French fries."
- Decision Making – "I am going to get some French Fries."
- Memory - "Oh, but I promised to eat healthier."
- Reasoning - "Surely, a small portion would not hurt."
- Action - "I am getting French fries."

Artificial neural networks try to simulate this layered approach to information processing and decision making. In its simplest form artificial neural networks consist of 3 main layers: the input layer (entry of data into the system), the hidden layer (processing of information occurs here) and the output layer (where the neural network decides what to do based on the data). The list below will provide detailed insight into the components of an artificial neural network.

3.1.1 The Perceptron

The neuron is most commonly referred to as the building block or the most basic unit of the neural network. Weights are assigned to a neuron based on its relative importance against other inputs.

The perceptron is a single layer network, which can be thought of as a linear binary classifier which maps some input vector x to a single output value (either 0 or 1). In order to make this mapping, a function $f(x)$ is used. The function $f(x)$ is given by equation 3.1. The dot product is computed between the input vector (x_i denotes the i^{th} input) and the weights, $\mathbf{w}$ (w_i denotes the i^{th} weight). The ultimate goal is to optimise the input weights such that the network is able to learn which are the important input features.

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=1}^{p} w_i x_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Figure 3.2 showcases the structure of a perceptron with one input x_1 and a bias term.

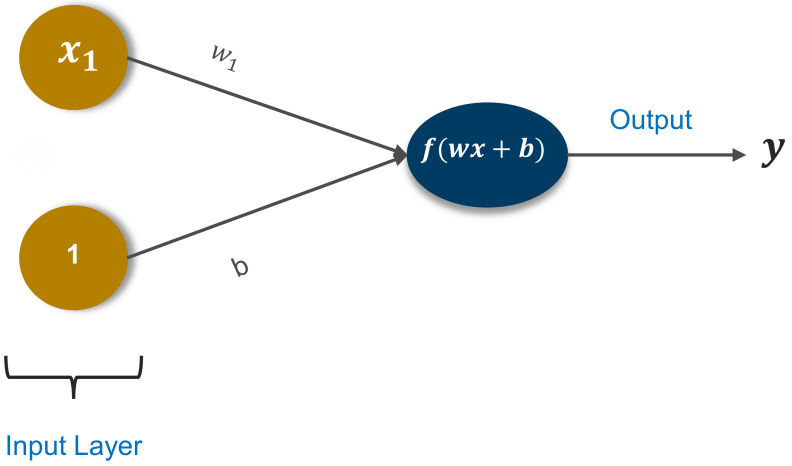

Figure 3.2: Perceptron single layer network

Equation 3.2 presents the modified function of equation 3.1 $f(x, b)$ to incorporate the bias term.

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=1}^{p} w_i x_i + b \ > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Based on Figure 3.2 the input to the network is x_1 and has a weight of w_1 assigned to it. In the domain of sentiment analysis x_1 corresponds to an input vector of text. The weight is essentially an intrinsic parameter. This implies that it is a parameter that the model has control over in order to obtain a better fit for the output. The second input as part of the input layer is known as the bias. The bias is determined solely by the value b, since the weight is set to a constant value of 1. The function of the bias term is to add some level of unpredictability to the model. This helps provide the model with flexibility to adapt to different unseen inputs when using testing data. Given that sentiment analysis is the domain focus of this research, the output y of the network will be a predicted probability. This predicted probability refers to the likelihood of the input text from either being positive or negative.

3.1.2 Layers

Neurons are organized into layers by neural networks. A fully connected layer refers to a layer in which every neuron is connected to every other neuron in its proceeding layer. Section 3.1.1 discussed a single layer network which is the perceptron. However, more than one layer is required to enable the network to approximate more complex problems. This subsection provides a high-level overview of multilayer perceptrons.

The level of complexity relates to the fact that several functions can be connected in a chain. A chain is described as the most commonly used structure of neural networks (Goodfellow et al., 2016). Consider functions f_1 and f_2, which can be connected in the following way: $f(x) = f_2(f_1(x))$. The function $f(x)$ has a higher complexity than the individual functions that this function is comprised of i.e., f_1 and f_2. Functions f_1 and f_2, are referred to as layer 1 and layer 2 respectively.

Based on this, there can be any number of layers which are added to a chain. Given that the length of the network can be large, this gives rise to the term “deep neural networks”. The depth of the neural network relates to the number of functions chained together. Figure 3.3 diagrammatically illustrates a multilayer neural network having two layers.

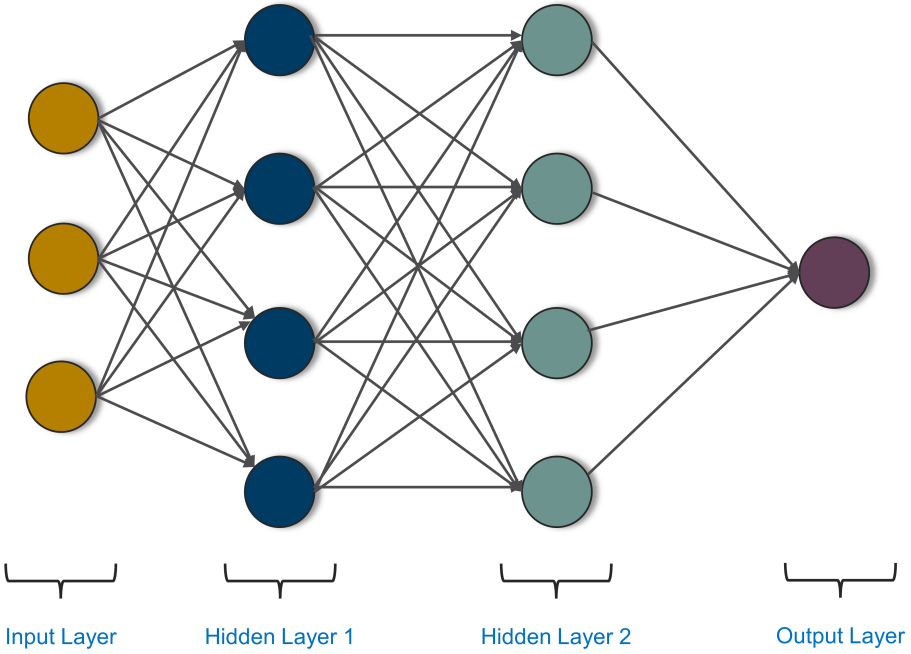

Figure 3.3: Multilayer neural network

In Figure 3.3, the input layer is depicted as the outermost yellow layer. The value for the first hidden layer (*Hidden Layer 1*) is generated once all the input layer node values are multiplied with their corresponding weight and summarized. The first hidden layer has a specified activation function which decides upon which node will be "activated" as well as its level of "activeness".

3.1.3 Activation Function

The activation function adds a level of complexity to the neural network. An activation function is a function that transforms our input data using either a linear or non-linear method. The most commonly used activation functions are sigmoid (Turian et al., 2009), ReLU (Nair and Hinton, 2010) and softmax (Bishop, 2006). Equation 3.3 presents the sigmoid activation function. This function maps the input values to a bound, between 0 and 1.

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{3.3}$$

The ReLU activation is given by equation 3.4. A value of 0 is output by this function if the input value is less than 0, whilst yielding an output of x when the input value is greater than 0.

$$f(x) = max\{x, 0\} \tag{3.4}$$

The softmax activation is presented by equation 3.5. The softmax function is typically used for multiclass classification problems. The output values from the softmax activation function correspond to the probability associated with each class and sum to 1.

$$f(x) = \frac{e^{x_j}}{\sum_{j=1}^{D} e^{x_j}} \tag{3.5}$$

where

D = dimension of x (in most instances the number of classes)

$j = 1, ..., D$

Having non-linear activation functions incorporated to the neural network model allows for an element of adaptability in the case of a classification problem the model will be able to predict classes that do not have approximate non-linear functions or linear decision boundaries. In essence without the presence of an activation function neural networks would only be able to learn linear relationships.

3.1.4 Propagation Function

The function responsible for the transportation of values through the neurons of a neural network's layers is known as a propagation function. There are two main propagation functions that work in a neural network: forward propagation which delivers the predicted value and backward propagation by which the error value is delivered. The error value is calculated between the expected outputs and the outputs forward propagated from the network.

Forward propagation refers to the process of calculating and storing intermediate variables (which includes outputs) for a neural network, from the input layer through to the output layer.

Backpropagation is the heart of a neural network and a clear distinction needs to be made between backpropagation and optimizers. Backpropagation refers to a process of efficiently calculating the gradients of the parameters of a neural network.

Whereas an optimizer is actually used for the training of a neural network, using the gradients computed with backpropagation.

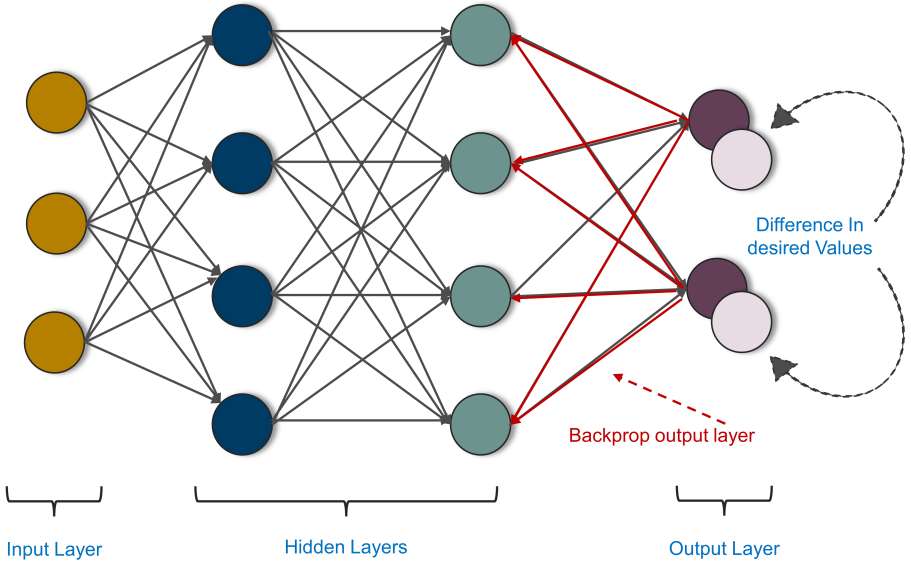

Figure 3.4: Diagrammatic illustration of backpropagation

In brief, backpropagation navigates the neural network in reverse order, from output to input layer based on the calculus chain rule. The algorithm results in the storage of any required intermediate variables (partial derivatives) while the gradient is calculated with respect to parameters of the network such as w_{ij}. Listed below are the steps that better articulate the back propagation method in Figure 3.4:

- The input layer is used to enter data into the network.
- The input data is processed by each neuron in the network, with the resulting values steadily percolating, from layer to layer, the output layer generates a result.
- A comparison is made between the actual and expected outputs of the network for a given input. The resultant output is an error value which is the difference between the given input and predicted output.
- An adjustment is made based on the resultant error value on the networks connection weights. This adjustment takes place backwards from the output layer, through the hidden layers, and to the input layer, until the correct output is produced. The fine-tuning of the weights, ensures that the correct output is produced for a given input.

3.1.5 Weight Optimization

During the initialization of the neural network, a random assignment of weights takes place. The neural network showcases its power through the large level of control it has over the data through the adjustment of these weights. Through an iterative process, the neural network adjusts its weights and measures the related performance. This process is continued until some stopping criterion is met.

The overall accuracy of the predictions is determined by some loss or cost function. This function compares the model's output with the actual output in order to determine how well the model is estimating the input dataset. In summary a loss function is provided to the neural network model in order to minimize, which is done through the incremental tweaking of the weights. To complete this loss function, a loss metric would need to be established in order to bring about level of tangibility of measuring the overall accuracy of the model. Given that predicting sentiments are classification in nature and based on the literature surveyed, the two main sentiment categories are positive and negative. As such, this presents a binary classification problem. Binary crossentropy is a loss function that is used in binary classification tasks. Binary crossentropy is the negative average of the log of corrected predicted probabilities. Each of the predicted probabilities are compared to the actual class output, which can either be 0 or 1 (Ruby and Yendapalli, 2020). After which, a score is calculated which penalizes the probabilities based on the distance from the expected value. The binary crossentropy loss function is given in equation 3.6.

$$Loss = \frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{3.6}$$

Where y is the actual value and $\hat{y}$ being the estimated output from a neural network, whilst n refers to the sample size of the input data. Based on equation 3.6, binary crossentropy needs to compute the logarithms of $\hat{y}_i$ and $1 - \hat{y}_i$, which only exist if $\hat{y}_i$ lies between 0 and 1. Hence, the sigmoid function is applied to $\hat{y}_i$ to ensure that this condition is satisfied.

Lastly, the optimizer provides a meaningful position in finding the best set of weights. In neural networks, the optimisation method used is stochastic gradient descent (Ketkar, 2017). Gradient descent is a method of minimizing an objective function $L(\theta)$, which is parameterized by a model's parameters θ. It performs this through the update of the parameters in the opposite direction of objective functions' gradient $\nabla_\theta L(\theta)$ w.r.t to the θ parameters. A learning rate η governs the rate at which the parameter estimates are updated by an algorithm. The stochastic gradient descent algorithm will repeat the following steps after every time period (epoch):

- As a starting point, the weights need to be initialized by setting them to some values.
- The weights will need to be updated such that the cost function is reduced.
- The algorithm will come to a stop when the minimum error on the data is reached or until the number of iterations has been completed.

Equation 3.7 showcases the parameter updates performed during a single step in the gradient descent algorithm.

$$\theta := \theta - \eta \times \nabla_\theta L(\theta) \tag{3.7}$$

One major drawback of gradient descent is that it can be very slow for very large datasets. This is due to the fact that parameters are updated only once for each epoch, which implies that a large number of epochs is required to have a substantial number of updates. As such, over the years there have been many proposed and implemented modifications to gradient descent to enhance the overall performance. These include stochastic gradient descent with momentum (Polyak, 1964), RMSProp (Hinton et al., 2012) and Adaptive Moment Estimation (Adam) (Kingma and Ba, 2014).

The Adam optimizer can be viewed as a combination of RMSprop and stochastic gradient descent with momentum. This method involves an adaptive learning rate approach, which simply means that the individual learning rates are computed for different parameters. Furthermore, this method scales the learning rate through the usage of the squared gradients and exploits momentum by leveraging off the moving average of the gradient as opposed to gradient itself. This makes this method superior to that of stochastic gradient descent, hence showing huge improvements in the overall training time of algorithms (Kingma and Ba, 2014).

It is important to be able to measure the accuracy of the neural network in terms of predicting the correct sentiment. In order to do this, evaluation metrics need to be defined and these will be introduced in the next section.

3.1.6 Performance Evaluation Metrics

The performance measures listed below will form a wide-spread view on the ability to effectively critique and compare the various neural network algorithms for WSD in sentiment classification (either positive or negative). At the outset it would be best to define a confusion matrix as described in Table 3.1, in order to articulate and expand on the definition of the various performance metrics.

Table 3.1: Confusion matrix for sentiment classification

		Actual Labels	
		Positive	Negative
Predicted Labels	Positive	*True Positive* (**TP**)	*False Positive* (**FP**)
	Negative	*False Negative* (**FN**)	*True Negative* (**TN**)

A confusion matrix provides a performance evaluation of a classification model by comparing the predicted and observed categories of a target variable. This in turn provides a holistic view of the overall performance of the classification model.

Consider the example where the input is a text review given a particular product category like fashion needs to be classified into a sentiment polarity of either "positive" or "negative" using a predictive model. The confusion matrix values or components would be as follows:

- **True positive (TP)** - A "positive" review correctly predicted as a "positive" sentiment by the predictive model.
- **False negative (FN)** - A "positive" review predicted as a "negative" sentiment by the predictive model.
- **True negative (TN)** - A "negative" review correctly predicted as a "negative" sentiment by the predictive model.
- **False positive (FP)** – A "negative" review predicted as a "positive" sentiment by the predictive model.

The actual labels are the pre-defined labels for the classification and the predicted labels are the labels predicted by the predictive model. As part of the performance evaluation of the deep learning models the following metrics were utilized and will be unpacked as the section unfolds: Accuracy, Precision, Recall and F1 score.

Accuracy

Accuracy is one of the main performance metrics for evaluating classification models. Accuracy can be viewed as the fraction of predictions that the predictive model got correct. Formally, accuracy is given by equation 3.8.

$$Accuracy = \frac{Total\ Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{3.8}$$

Given that this research focuses on a binary classification problem i.e., classifying whether a review is positive or negative given the context, equation 3.8 can be translated into equation 3.9 by drawing from the definition of the confusion matrix.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

Accuracy is a widely used metric for the evaluation of classification models, however accuracy alone does not provide a detailed picture when working with a class-imbalanced data set, in which there exists a difference between the number of positive and negative classes (Branco et al., 2015). Therefore, other parameters need to be looked at when evaluating the performance of a model. This is where information must be drawn from the confusion matrix in order to consider precision, recall and F1 score as alternative metrics for model performance evaluation.

Precision

This "quantifies the number of positive class predictions that actually belong to the positive class" (Brownlee, 2020). Precision aims to answer the question: "What proportion of positive identifications was actually correct?". The mathematical definition of precision is given by equation 3.10.

$$Precision = \frac{TP}{TP + FP} \tag{3.10}$$

Recall

Recall "quantifies the number of positive class predictions made out of all positive examples in the dataset" (Brownlee, 2020). This measure attempts to answer the following question: "What proportion of actual positives was identified correctly?". Recall is given by equation 3.11.

$$Recall = \frac{TP}{TP + FN} \tag{3.11}$$

Recall is commonly used when positive scenarios are needed to be correctly classified.

F-Measure

The F1 score balances both Precision and Recall in one metric (Brownlee, 2020). F1 score is the weighted average of Precision and Recall. This implies that false positives and false negatives are taken into account as part of this metric. The F1 score proves to be more useful than accuracy when working with class-imbalanced datasets. The F1 score is given by equation 3.12.

$$F1\ score = \frac{2(Recall \times Precision)}{(Recall + Precision)} \tag{3.12}$$

For the comparison of any two models, it is important to utilize the F1 score. There is a challenge that exists when comparing two models with low precision and high recall or vice versa. The F1 score aids in the ability to have Precision and Recall measured at the same time. The harmonic mean is used instead of arithmetic mean as it penalizes the extreme values more.

Confidence Interval Spread

The confidence interval provides value in its ability to quantify the uncertainty of the estimate. More precise estimates are often obtained from larger samples and the confidence intervals are typically smaller (better) (Cumming and Calin-Jageman, 2016). The confidence interval formula assuming a z-score statistic is given by equation 3.13. Single iteration estimates can be biased as this does not take into account the spread of the performance metric at hand, as such confidence intervals aim to reduce the level of bias by providing a spread to add more credibility to the performance metric estimate reported on.

$$95\%\ Confidence\ Interval = \bar{x} \pm\ z\frac{s}{\sqrt{n}} \qquad df = n \tag{3.13}$$

where:

$\bar{x}$ = sample mean
s = sample standard deviation
z = confidence level value
n = sample size
df = degrees of freedom

Based on the fundamentals of the artificial neural networks discussed above, it is evident that a single neural network differs from a deep learning algorithm in terms of the number of node layers or the depth of the overall network. Section 3.2 aims to deep dive into deep learning to better articulate the differences and overall dynamics of these algorithms and architectures.

3.2 Deep Learning

Deep learning is a branch of machine learning, that encompasses a suite of algorithms in which high-level abstractions in the data is modelled through the usage of multiple processing layer structures, or otherwise composed of multiple non-linear transformations (Bengio et al., 2013; Schmidhuber, 2015; LeCun et al., 2015; Arel et al., 2010).

Deep learning methods focus on learning representations of data. Alternatively, deep learning has been characterized as a re-branding of neural networks (Gomes, 2014; Collobert et al., 2011). This is largely due to fact that neural network architectures form the basis of most deep learning architectures. As a result of this, deep learning models are often referred to as deep neural networks.

The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks typically contain 2-3 hidden layers, while deep networks can have as many as 150 (Hornik, 1991). Deep learning algorithms are based on distributed representations. Distributed representations assume that observed data is generated by factors organized in layers. Deep learning further assumes that the layers of factors correspond to levels of abstraction. Different amounts of abstraction can be provided depending on the number of layers and layer sizes at hand (Bengio et al., 2013). Deep learning goes under the premise that the learning of abstract concepts occurs in a hierarchical fashion i.e., more abstract concepts are learned from lower level ones. Deep learning aids in separating these abstractions and effectively selects the most useful features for learning (Bengio et al., 2013).

One of the potentials of deep learning is replacing handcrafted features with efficient algorithms for hierarchical feature extraction, semi-supervised and unsupervised feature learning (Song and Lee, 2013; Xiong et al., 2015). This implies that the learning of the features can occur directly from the data without the need for manual feature extraction (LeCun et al., 2015). This benefit is an important one, due to unlabelled data being more readily available than labelled data.

To better understand the concept of deep learning, consider an example of a toddler who utters his/her first word and that word being "cat". The toddler starts to learn what a cat is and what is not, by pointing at objects and saying the word cat. The parent then says, "Yes, that is a cat." or, "No, that is not a cat." As the toddler continues to point at a variety of objects, an awareness is established relating to the features that cats possess. What the toddler is doing unknowingly is to clarify a complex abstraction i.e., the concept of a cat. Using a hierarchical approach, each level of abstraction in the hierarchy was constructed based on knowledge gathered from the previous hierarchical layer.

This is in contrast to traditional (i.e., shallow) machine learning algorithms which are linear whereas deep learning algorithms are in the form of a hierarchical stack of increasing complexity and abstraction. In the context of image recognition, the machine learning workflow starts with the manual extraction of relevant features from images. These relevant features are then used to develop a model that categorizes the distinct objects in an image. This is in contrast to a deep learning workflow, in which relevant features are automatically extracted from images (LeCun et al., 2015). In addition to this, "end-to-end learning" is performed by deep learning. This concept refers to an instance in which raw data and the mandated task is given to the network, and the network learns how to do this automatically.

Figure 3.5 provides an illustrative view of the comparison between machine learning and deep learning approaches when categorizing vehicles.

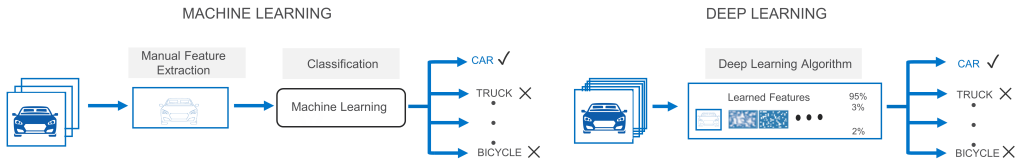

Figure 3.5: Comparing machine learning and deep learning in terms of vehicle image recognition
(adapted from Mathworks (2021))

Another differentiating factor is that deep learning algorithms scale with data, whereas shallow learning exhibits convergence. Traditional machine learning methods that plateau at a certain level of performance as more training data is incorporated into the network is referred to as shallow learning. As such, deep learning has been applied to many areas of research apart from NLP. These include; automatic speech recognition (Sak et al., 2015; Deng and Li, 2013), image recognition (Shen et al., 2017; Krizhevsky et al., 2012), recommendation systems (Van Den Oord et al., 2013; Elkahky et al., 2015). Furthermore, for a wide-spread of applications there exists a wide variety of deep learning architectures, however most of these architectures are branches of some original parent architecture.

Considering the promising results generated from various authors (detailed in Chapter 2) through the application of CNN, RNN and LSTM algorithms to the WSD and sentiment analysis research areas, the next section will provide a deep dive into the overall dynamics of the underlying architectures for these desired deep learning algorithms as these would form the basis of the methods applied within this thesis.

3.3 Deep Neural Network Architectures

3.3.1 Convolutional Neural Networks

CNN is a feed forward network primarily used in image recognition and various tasks within NLP (LeCun et al., 1998). The number of hidden layers between the input and output layers contributes to the overall strength of the CNN. Within each layer, a set of features are extracted. Feature maps are then generated through the application of a series of filters over the input. The entire input is scanned by the filter and subsequently multiplies its weights by the input values. A rectifier function is applied to the resultant output so as to increase non-linearity in the CNN. A loss function estimates the error loss of a CNN, so that the weights can be updated in an attempt to reduce the loss on the next evaluation. Different parts of the input are highlighted by the filters generated by the feature maps. In the NLP setting, a pre-processing step is conducted in which the input text is converted to a matrix representation. Sentence characters are utilized as rows and letters as columns. A filter slides over the words of the matrix. Hence, words are recognized by the implementation of the sliding window technique.

CNN has three main layers which are (Aghdam and Heravi, 2017):

- Convolution layer
- Pooling layer
- Fully connected layer

Figure 3.6 provides a diagrammatic view of the architecture of a CNN model

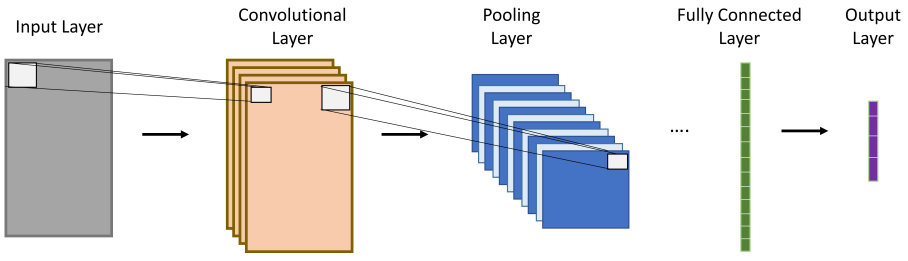

Figure 3.6: CNN model with various layers - (adapted from Prabha and Srikanth (2019))

3.3.1.1 Word Embeddings

Before discussing the concept of convolutions and convolutional layers, it is important to first introduce the concept of word embeddings. The main challenge in NLP and modelling language is due to the fact that the language construct is in the form of text, i.e., words and sentences. As such identifying the context of a word in text is paramount in the case of sentiment analysis and text analytics. Hence, text needs to be converted into an interpretable form that can be understood by NLP text-based systems. This interpretable form is usually a mathematical representation, which represents the input text as numeric vectors. Words are often mapped to vectors in a vector-space with the intention to reflect the meaning of the words relative to other words. This mathematical representation encompasses the following components:

- The word's actual meaning.
- The context of the word.

Mathematical representations of words which models the actual and semantic meaning of words are known as word embeddings. The philosophy is that words that occur in similar contexts tend to have similar meanings. This implies that a words' meaning is given by the surrounding words that it appears frequently with. The approach of word embeddings is considered to be a key contribution to deep learning on difficult NLP applications (Dessì et al., 2021).

There are several methodologies in order to generate word embeddings, these include dimensionality reduction (Raunak et al., 2019), neural network language-based models and co-occurrence matrices. Dimensionality reduction methods have two major drawbacks; they are computationally expensive and most importantly they do not consider the global context in which the same word occurs. Neural network based language models like Word2Vec (Mikolov et al., 2013b) on the other hand are able to capture statistical information at a local level, however do not consider the count or co-occurrence of words in a global context.

Co-occurrence matrix methods overcome the above-mentioned drawbacks in that they capture the global context of a word and most importantly how frequently two words appear together. As such, it is for this reason that pre-trained GloVe (Pennington et al., 2014) embeddings were utilized as part of this research to create the associated embedding matrices and embedding layers within the implementation of the deep learning algorithms (discussed in Section 4.3.2).

Given that the model captures statistics directly at a global level, it is named as 'Global Vectors' model. GloVe is trained on global word-word co-occurrence counts.

The intuition behind GloVe is based on the fact that the most important statistical information that is available for the model to 'learn' a word representation is word co-occurrence.

A co-occurrence matrix for a given sentence provides an indication of how frequent a given pair of words appear together. The elements in the matrix represent the count of the word pairs that occur together. The occurrence relates to immediately before and immediately after a specific word.

Consider a corpus containing V words. The co-occurrence matrix be defined as $\mathbf{X}$ which will be a $V \times V$ matrix. $\mathbf{X}$ will then have an i^{th} row and j^{th} column. X_{ij} denotes the number of times that word i has co-occurred with word j. Let $X_i = \sum_k X_{ik}$ denote the frequency of any word appearing in the context of word i.

The important question is to understand how to establish a metric that measures semantic similarity between words from the co-occurrence matrix. In order to ascertain this, the following conditional probability must be defined:

- Let $P(j|i)$ be the probability that the word j appears in the context of word i.
- $P(j|i) = \frac{X_{ij}}{X_i}$

In order to demonstrate this practically, consider the example from Pennington et al. (2014) which is based on the words "ice" and "steam". As such Figure 3.7 provides an excerpt of the table taken from Pennington et al. (2014). This provides the co-occurrence probabilities for target words ice and steam with selected context words from the input corpus.

Probability and Ratio	k = *solid*	k = *gas*	k = *water*	k = *fashion*
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 3.7: Co-occurrence probabilities for target words ice and steam in relation to selected context words (Pennington et al., 2014)

Based on Figure 3.7, it is evident that ice is more similar to solid than steam, due to the fact that $P(solid|ice)$ has a larger value than $P(solid|steam)$. In addition to this, both ice and steam are not related to fashion as the probability ratios $P(fashion|ice)$ and $P(fashion|steam)$ are small. It is important to note that by examining the stand-alone probabilities; $P(fashion|ice)$ or $P(fashion|steam)$, it will not be sufficient enough to infer the lack of relationship of ice and steam with fashion.

As such the idea behind GloVe is to consider co-occurrence probabilities as a ratio of the conditional probabilities and use it as a word representation. Based on the ratio of the conditional probabilities: $P(fashion|ice)/P(fashion|steam)$ this yields a value of 0.96. Hence, the probability ratios showcase a better approach for distinguishing relevant words (solid and gas) from irrelevant words like fashion.

3.3.1.2 Convolutional Layer

CNNs use the concept of weight sharing, this implies that fewer parameters are required than the conventional artificial neural networks, which enables CNNs to converge faster than conventional neural networks. The convolution layers apply a convolutional operation to extract features from the input text or images. There are two main parameters associated with the sliding window; the stride size and the number of filters. The stride refers to the size of the step filter that moves every instance of time. The filter count refers to the desired number of filters to be applied to the input text.

Figure 3.8 provides an example of the calculations involved in a 1D discrete convolution operation. This example uses a kernel size of 3, which means that the weights (w_1, w_2, w_3) are shared by each stride of the input layer $(i_1, i_2, ..., i_6)$. The output values are $o_1, o_2, ..., o_6$. The inputs and weights in the kernel window are multiplied together. Feature map values are based on the summarized resultant outputs of the input and weight calculation. Consider output o_3, the value is obtained as: $o_3 = w_1 \times i_2 + w_2 \times i_3 + w_3 \times i_4$.

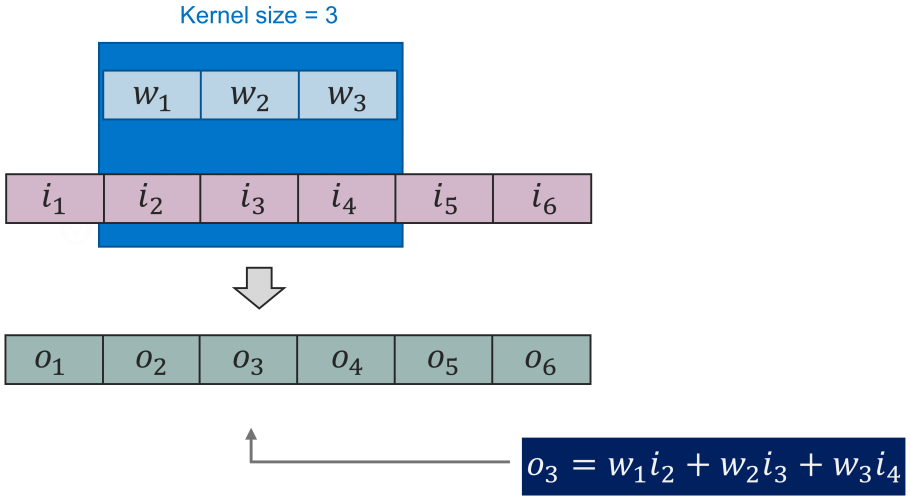

Figure 3.8: 1D convolution operation calculations

3.3.1.3 Pooling Layer

The overall complexity is reduced by the pooling layer. This is achieved through the reduction in size of the output from one layer to the proceeding layer, in which the most important features are retained so as to enhance the computational efficiency to the next calculation step. There are different pooling techniques that are utilized in reducing outputs while still preserving important features (Scherer et al., 2010). Max pooling is a widely used pooling method, which involves the selection of the maximum element within the pooling window.

Figure 3.9 illustrates the 1D max pooling operation. This example considers a feature map length of 6 before pooling is applied. In addition to this the pooling size considered is 2. Hence after applying max pooling the output length is 3 and the maximum values are selected as feature values for the next layer.

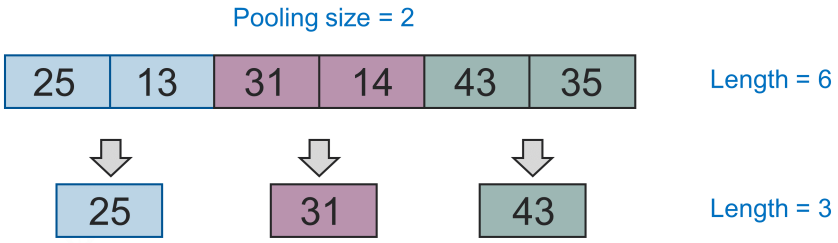

Figure 3.9: 1D max pooling

3.3.1.4 Fully Connected Layer

In order to feed the pooled output from stacked featured maps into the next layer, feature maps are flattened into one column. The final layers in a CNN are typically fully connected. The fully connected layer is essentially where all the inputs from one layer are connected to every activation unit of the next layer.

3.3.1.5 Output layer

The output of the CNN is a predicted probability. Considering the sentiment analysis application of this research, in which the focus is to predict whether a review text is positive or negative. This problem is classification in its nature and as such the predicted probability relates to the likelihood of a text being either negative or positive. In order to ascertain the prediction class, a threshold on the predicted probability needs to be specified.

This threshold governs the choice to turn the predicted probability into a class label. For instance, if the threshold was set to 0.5, then every predicted probability greater than 0.5 will be classified as a positive sentiment and every predicted probability less than 0.5 will be classified as negative.

3.3.1.6 Dropout

Due to the overfitting problem that exists during training, the concept of regularization is key in order to reduce the impact of overfitting. Dropout is a regularization technique that assists in preventing overfitting (Nair and Hinton, 2010). Figure 3.10 provides a diagrammatic illustration of two multilayered feed forward networks. The network on the left contains no dropout, whilst the network on the right does contain dropout.

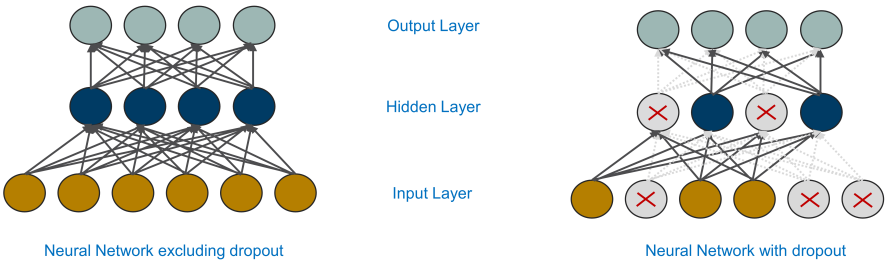

Figure 3.10: Illustration of dropout

Dropout involves the process of randomly selecting neurons, with the aim of disabling the selected neurons during the training process. This results in the output values of the disabled neurons to be 0. This is demonstrated on the right-hand side of Figure 3.10, in which the first layer has three neurons which are disabled, and the second layer has two disabled neurons. The grey neurons and dashed lines indicate this practically. Even though this involves a simple random selection of neurons which results in neurons being temporarily disabled, the neural network can still work well (Kuo and Huang, 2018).

3.3.2 Recurrent Neural Networks

The RNN forms the foundational basis in terms of network architectures from which other deep learning architectures are derived and built (Aggarwal et al., 2018). The main difference between a traditional multilayer network and a recurrent network is that there exist connections that feed back into prior layers or the same layer. This concept is known as back propagation through time. This differs from CNN in which a feed forward network is utilized.

Back propagation through time, allows RNNs to maintain memory of past inputs and model problems in time. The design is based on the defining principle that humans do not think from scratch at every time point. In order to predict the proceeding word in a given context, RNNs make use of the previous word. RNNs have a widespread usage in a variety of NLP text applications, voice recognition and translation applications. Figure 3.11 provides a diagrammatic view of the architecture of an Elman RNN model (Toha and Tokhi, 2008).

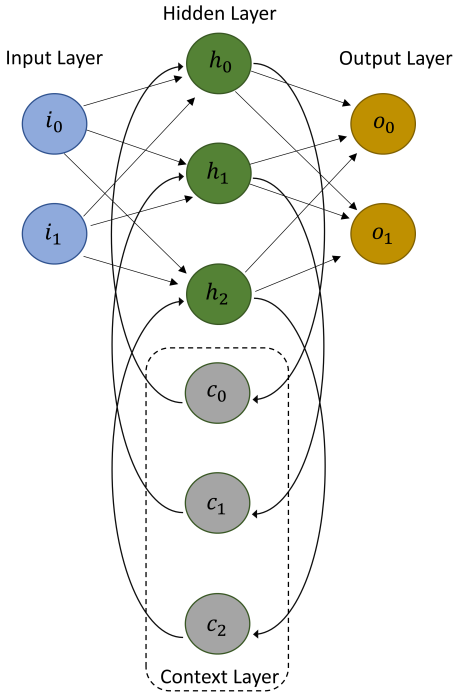

Figure 3.11: Elman RNN architecture - (adapted from Madhavan (2017))

3.3.3 Long Short Term Memory

LSTM (Hochreiter and Schmidhuber, 1997) is an extension of RNN, in that inputs can be remembered over a long period of time. LSTMs possess an advanced memory, in contrast to RNNs which have a basic internal memory. The LSTM method introduced the concept of a memory cell, thereby moving away from typical neuron-based neural network architectures. The memory cell has the ability of value retention for both a short or long period of time as a function of its inputs. This allows for important information to be remembered by the cell, as opposed to just its last computed value. This addresses the drawback of RNN which suffered from vanishing gradients. In the LSTM approach, memory can be gated. In order to control the flow of information into or out of the memory cell, there exist three control gates within the architecture.

The input gate provides control over when new information is allowed to flow into the memory. The forget gate provides control over when existing or current information can be forgotten. This allows new important features to be remembered by the memory cell. Lastly, the output gate decides upon when the information contained in the memory cell is used in the output from the cell. Figure 3.12 provides a diagrammatic view of the architecture of a LSTM model.

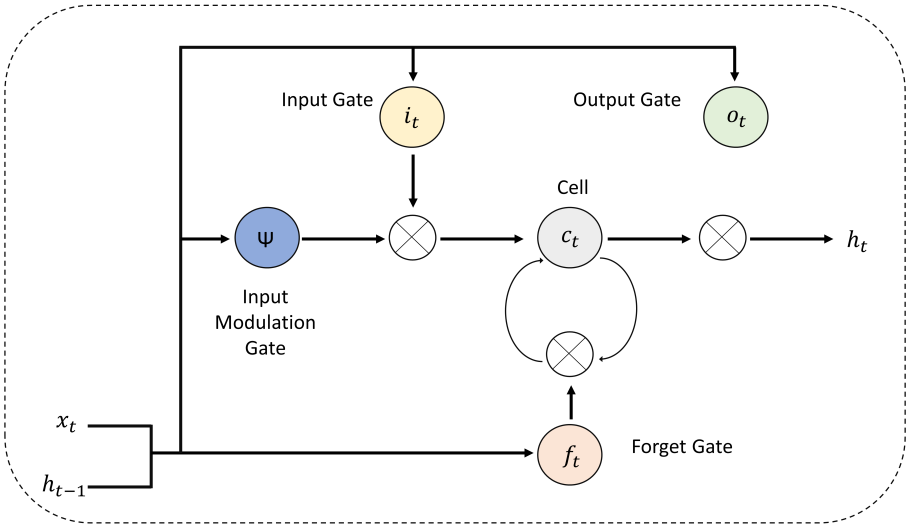

Figure 3.12: LSTM model architecture (adapted from Kang (2017))

The following components that relate to the LSTM architecture in Figure 3.12 are articulated below:

The cell state

There are two inputs for the RNN cell, this includes the last hidden states' output (h_{t-1}) and the observation at time = t given by x_t. The cell state (c_t) is referred to as the long term memory. The recursive nature of the cell is shown by the looping arrows. This allows for the LSTM cell to store information from previous intervals. The forget gate (f_t) modifies the cell state and is further adjusted by the input modulation gate ($\tilde{c_t}$). The equation for the cell state is given by equation 3.14.

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{3.14}$$

From equation 3.14, the previous cell state c_{t-1} forgets by being multiplied with the forget gate (f_t) and as a result, new information is added through the output of the input gates (i_t).

The forget gate

The output of the forget gate (f_t) informs the cell state (c_t) about which information to forget or keep. Information from the previous hidden state and information from the current input is passed through a sigmoid function. As such the output values range between 0 and 1. The cell state stores the information if forget gates' output is closer to 1. Conversely, the information is disregarded if the output of the forget gate is closer to 0. The mathematical representation of the forget gate is given by equation 3.15.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.15}$$

From equation 3.15, a sigmoid function, σ is applied to the weighted input/observation and previous hidden state.

The input gate

The input gate (i_t) is usually referred to as the save vector and governs the information entering the cell state. The activation functions for each gate are important components for each gate. The input gate is a sigmoid function and has a range of [0,1]. Equation 3.16 provides a mathematical description for the input gate.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.16}$$

Given that the mathematical equation for the cell state is a function of the previous cell state, a sigmoid function alone can only include memory and not have the ability to remove/forget memory. As such, this is the rationale for the tanh activation function of the input modulation gate ($\tilde{c}_t$). The tanh function effectively allows the cell state to forget memory due to its function range of [-1, 1]. Equation 3.17 provides a mathematical description of the input modulation gate.

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{3.17}$$

The output gate

The output gate (o_t) is usually referred to as the focus vector. It provides an indication of which values should move forward to the hidden state. Equation 3.18 provides a mathematical description for the output gate.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.18}$$

The hidden state

The hidden state is essentially the working memory. This decides which information should be taken through to the next sequence and is presented by equation 3.19.

$$h_t = o_t \times \tanh(c_t) \tag{3.19}$$

The research objectives highlighted in Section 1.2 will feed as input into the proceeding chapter so as to understand how these objectives will be met in order to satisfy the overarching aim of this research.

Chapter 4

Methodology

This chapter presents the methodology used to apply deep learning to WSD and sentiment analysis. The proposed methodology will be applied and tested using sentiment text related data on the CNN and LSTM deep learning algorithms.

The performance will be measured, and hyper-parameter tuning will be conducted on the deep learning algorithms in order to yield improved results. This will be repeated on several iterations until optimal, or near-optimal results are obtained.

The various optimal models for each of the algorithm categories will then be compared in terms of the overall accuracy, precision, recall and F-measure. This will in effect allow for the assessment of the effectiveness of the CNN and LSTM algorithms in WSD within the sentiment analysis domain and thus aiming to solve the research objectives outlined in Section 1.2.

Figure 4.1 provides a diagrammatic illustration of the overall methodology process at a high-level.

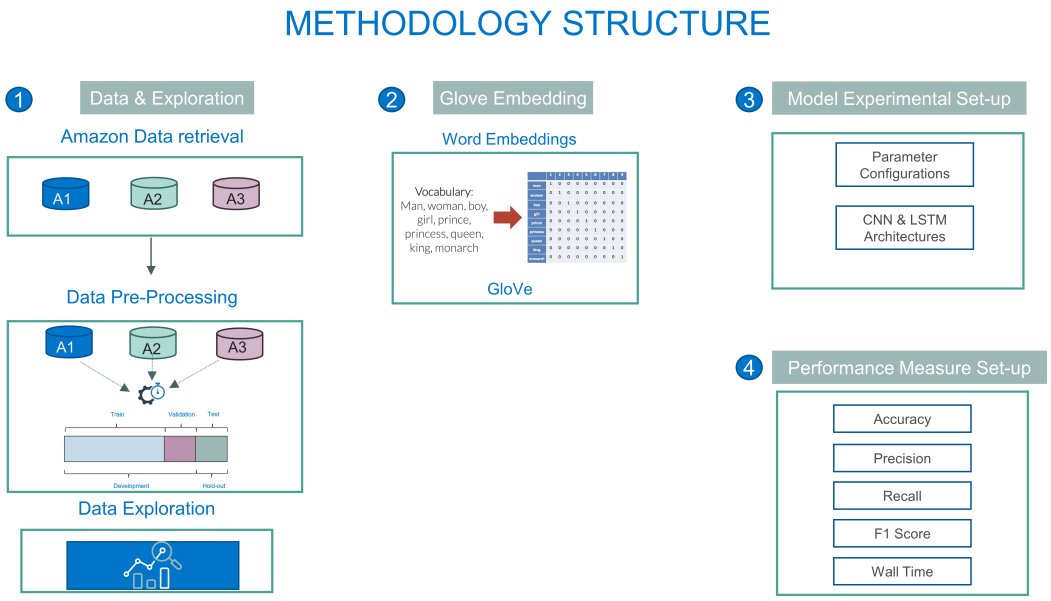

Figure 4.1: High-level methodology structure

This chapter will comprise of the following sections:

- **Data and Exploration** – This section is aimed at describing the input data, the data pre-processing techniques as well as an analysis of the data.
- **GloVe Embedding Application** – This section showcases the practical implementation of GloVe embeddings on the input dataset.
- **Model Experimental Set-up** – This section will focus on the experimental set up in order to fulfil the aims of this research.
- **Performance Measure Analysis Set-up** – This section provides insight into the metrics used to evaluate the implemented deep learning algorithms in order to draw effective conclusions.

The remainder of this chapter will discuss the steps highlighted in Figure 4.1.

4.1 Data and Exploration

4.1.1 Input Data Overview

The aim of this research is to explore WSD in sentiment analysis. Based on the literature surveyed, deep learning methods are recognized as state-of-the-art for this task. This research focuses on customer review data. The Amazon product review dataset was selected as it is the most commonly used dataset for NLP tasks. This section provides a summary of the Amazon product reviews data and serves as a preamble to the data pre-processing component of this section.

Amazon is one of the largest online retailers and a prominent figure in the E-commerce industry. As such, this comes with a large customer base and consequently a considerable amount of customer reviews. This aligns with the domain focal points of this research which are the opinions and overall sentiments of customers. The Amazon product data was approved and provided by researchers He and McAuley (2016) and is accessible at: `https://nijianmo.github.io/amazon/index.html`. The Amazon dataset contains product reviews from Amazon spanning from May 1996 – July 2014. The product metadata relating to the Amazon product reviews are tabulated in Table A.1 of Appendix A.

The Amazon product reviews dataset is comprised of multiple product categories, however for the implementation of this research three categories were selected (as shown in Table 4.1). The reason for the choice of these specific product categories are due to the level of difference in the overall context and topic between these categories, enabling WSD to be tested effectively. This relates to the fact that the meaning of words differ from context to context. The number of observations for each related selected product category is showcased in Table 4.1.

Table 4.1: Selected Amazon product categories & number of reviews

Amazon product category	**Number of reviews**
Amazon Fashion	3176
Software	12805
Grocery and Gourmet Food	57193

The input data described in Table 4.1 will be split into the following two sub datasets that will be used to investigate the key objectives of this research (as detailed in Section 1.2):

- **2 Categories:** This encompasses the Amazon fashion and Software datasets.
- **3 Categories:** This encompasses all 3 datasets described in Table 4.1.

Figure 4.2 provides an excerpt of the fashion dataset in raw format.

overall	reviewTime	reviewerID	reviewerName	reviewText
5.0	09 4, 2015	ALJ66O1Y6SLHA	Tonya B.	Great product and price!
3.0	04 18, 2017	A3T8NHQYBQSV1X	Mark	The waist string broke first time trying them ...
5.0	04 13, 2017	A2NEJIX5NIH2ZZ	jami	Fit perfectly. I bought dark grey, and they di...
3.0	03 24, 2017	A1G0HYMR02WM2W	Cici Ciconia	Average product. As described.
1.0	02 8, 2017	A21HH0VIBKK80J	Sondra McDonald	Was terribly disappointed, the pants were way ...
5.0	07 22, 2015	A3CNRM11BHAR1A	DD	Havaiana's are THE BEST!!!

Figure 4.2: Amazon fashion data excerpt

In addition to the Amazon dataset, the Wikipedia 2014 + Gigaword 5 dataset (Pennington et al., 2014) was used as part of the pre-trained GloVe embeddings. This dataset encompasses pre-trained word vectors from the combined Wikipedia 2014 + Gigaword 5th Edition corpora. The dataset that supports this is publicly available at: `https://nlp.stanford.edu/projects/glove/`

4.1.2 Data Pre-processing

The data management process is focused on the pre-processing of the dataset, which enables the data to be cleaned and normalized for ease of computation by machine learning algorithms within the NLP domain. The data management was conducted within Python 3.7 and interfaced using Python Jupyter notebooks. The main pre-processing components that were conducted as part of this research were there the following:

- Data ingestion and file conversion
- Context labelling and encoding
- Sentiment category classification and target variable creation
- Text pre-processing
- Synset enrichment
- Overlap selection

4.1.2.1 Data ingestion and file conversion

The Amazon reviews datasets were extracted in individual product categories which were embedded within .json file format inputs. In order for these files to be usable they were converted from a .json format into a data frame structure within the Python environment as shown in Figure 4.3.

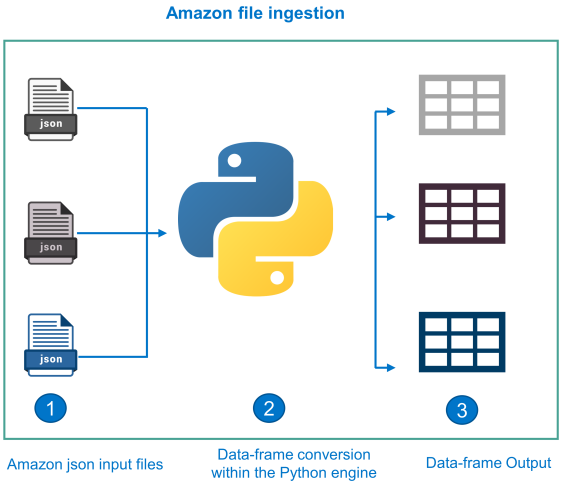

Figure 4.3: Amazon input data file ingestion process

4.1.2.2 Context Labelling and Encoding

Within the online retail sector, it is clear that every retail product is grouped into a specific retail category. For instance, consider designer shoes, these would be listed under the fashion product category listing for a particular retailer. This allows the customer a much easier interaction when searching for a desired product hence improving the online shopping experience.

Online retailers have a review section, allowing customers to express their opinions and experiences with the products purchased from a retailer. As such, a product category will be prompted to the reviewing customer, in order for the retailer to better manage reviews on a product category level and make adjustments to their product offering accordingly.

Amazon retailing follows suit with the above mentioned and it is for this reason that it was decided upon to include a context category within the dataset and make the reasonable assumption that a product category context is known. As such, each of the Amazon reviews datasets were enriched to include a feature known as *category_context* which is a representation of the product category a review relates to.

As it stands the *category_context* feature is essentially a categorical feature as it contains label values rather than numeric values. This poses a challenge as most machine learning algorithms are unable to deal directly with categorical data. They require all input variables and output variables to be numeric. Label encoding exhibits an additional problem in that higher categorical values are assumed to be a 'better' category.

To overcome this, one-hot encoding is utilized in which "binarization" is performed on the category and included as a feature in the training of the model. One-hot encoding involves the conversion of categorical variables into a usable form that can be input into machine learning algorithms so as to make better predictions.

Figure 4.4 showcases the *category_context* feature enrichment and one hot encoding process.

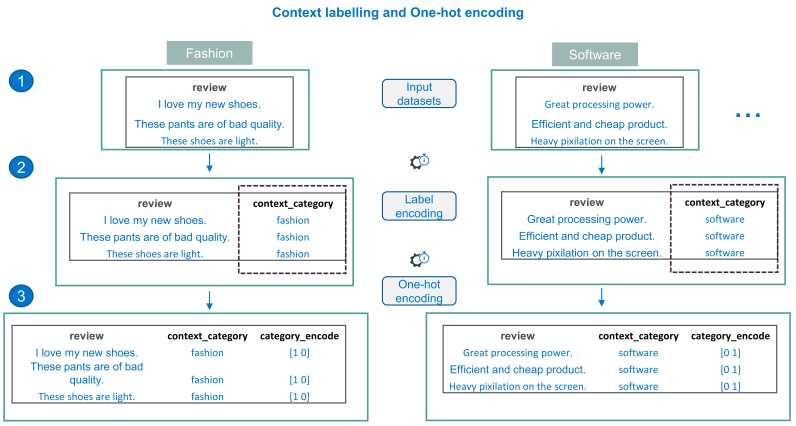

Figure 4.4: Context label categorization and one hot encoding process

The one-hot encoded feature will serve as input into the context-aware deep learning models covered in Section 4.3, which will provide a basis for allowing the demonstration of the effectiveness of the model in disambiguating words to achieve the current sentiment.

4.1.2.3 Sentiment Category Classification and Target Variable Creation

This step is centred on the creation of a sentiment category feature which allows for the distinction between positive and negative reviews to be made. This feature is based solely on the feature known as *overall* from the Amazon reviews datasets. This feature effectively provides a customer rating of an experience with a particular Amazon product. The rating ranges from 1-5 with a rating of 1 being the most negative experience and a rating of 5 representative of the most positive experience. This is commonly known as the Likert scale (Likert, 1932), and is widely used by businesses when measuring the customer satisfaction.

A Likert scale assumes that an attitudes' strength is ordinal. For example, there is a numerical value associated with each of the five responses, that is used to measure the desired attitude. As part of this research rating values encompassed within the *overall* feature were grouped to create the following sentiment categories shown in Table 4.2. Ratings 1,2 and 3 are regarded as negative, whilst ratings 4 and 5 were grouped as positive.

Table 4.2: Selected Amazon product categories & number of reviews

Rating	**Sentiment**	**Sentiment Indicator**
1,2,3	Negative	0
4,5	Positive	1

One could argue that the rating score three should be in a category of its own, however for the purpose of this research ratings were grouped into two separate categories. This will ensure ease and effectiveness of implementation of the deep learning algorithms in the domain of sentiment analysis. The *Sentiment* column serves as the categorical feature and the *Sentiment Indicator* refers to the binary feature which will be representative of our target class variable.

The target variable in a dataset refers to the attribute about which you want deeper insight into. This is essentially the variable that the predictive model aims to predict. For the purpose of this research, the aim is to classify Amazon product reviews as either a positive experience (1) or negative experience (0) given the product context category. Figure 4.5 provides a diagrammatic illustration of the above-mentioned enrichment to the Amazon reviews datasets.

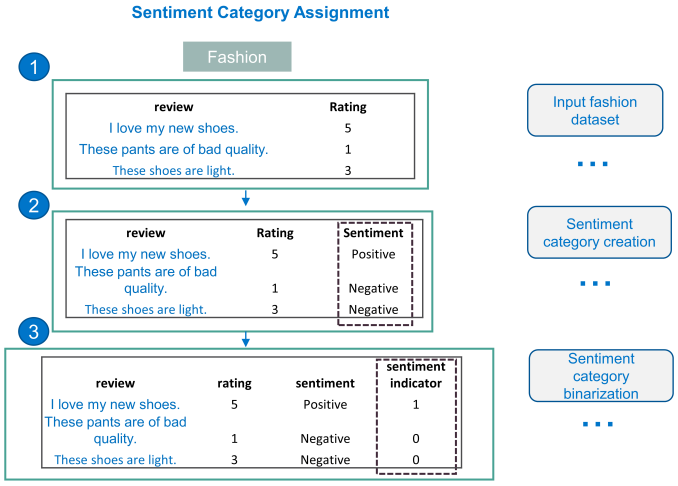

Figure 4.5: Sentiment category classification process

4.1.2.4 Text Pre-processing

Text pre-processing is the process of cleaning and preparing the input text for NLP tasks, in the case of this research sentiment classification given a particular context would be the end goal. Many words encompassed within review texts do not have an impact on the general orientation of the review itself. As such, the removal of the noise and uninformative parts of the text allows for a cleaner input and a more effective implementation of classification models in the domain of sentiment analysis.

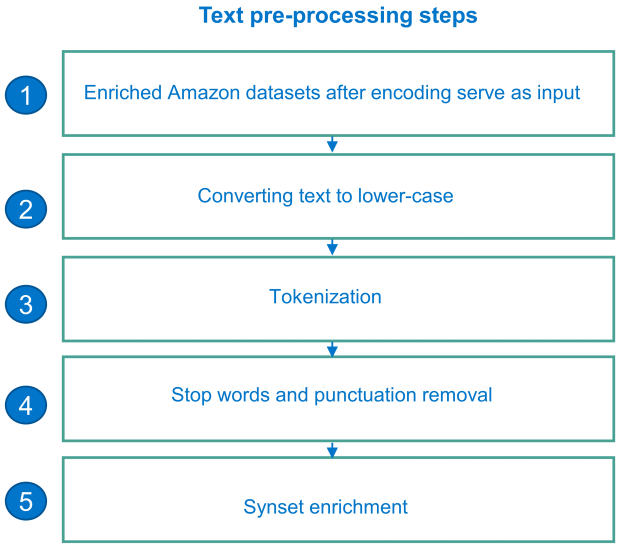

Figure 4.6: Text pre-processing components

There are four key components of text pre-processing that will be described in detail; lowercasing of text, tokenization, stop word removal and punctuation removal. Figure 4.6 diagrammatically illustrates the text pre-processing steps conducted as part of this research. This section will focus on the description and implementation of the text pre-processing components shown in Figure 4.6

Lowercasing of text

The first step of the text pre-processing utilized within this research is the lowercasing of input review text data. Lowercasing of text allows the input reviews to be in a consistent form throughout the NLP process. This will aid in the prevention of the deep learning models from treating a word which is at the beginning of a sentence with a capital letter different from the same word which appears in the latter part of the sentence, but without any capital letter.

Tokenization

Tokenization is a common task when dealing with textual data. Tokenization is a process that splits a phrase, paragraph, individual sentence or an entire document into smaller units. These smaller units are referred to as tokens and can be in the form of words, numbers and punctuation marks. Word boundaries enable these smaller units to be created. These refer to the end point of a word and the beginning of the proceeding word.

Stop word removal

The next component of the text pre-processing process relates to stop word removal. Stop words are the most common words in any language and usually repeated features which emerge in every text document. These words do not add much information to the text and as such would add little or no value to the sentiment classification process. Examples of a few stop words in the English language are “the”, “a”, “an”, “so” and “what”.

Removing stop words allows for the removal of low-level information from the review text in order to give more focus to the important information. This implies that the removal of such words does not showcase any negative consequences on the deep learning models that would be trained as part of this research. Removal of stop words reduces the input dataset size and thus will have a time reduction impact upon the training of the deep learning models. This is due to a fewer number of tokens involved in the training process.

Punctuation removal

The removal of punctuation within text is another noise removal technique as it involves removing unhelpful parts of the data. Drawing from the knowledge from word tokenization, the actual punctuation will be considered as a token. Given that this is not essential to the investigation and analysis, punctuations were removed.

It must also be noted that when working with NLP, word embeddings (discussed in Section 3.3.1.1) such as GloVe (Pennington et al., 2014), Word2Vec (Mikolov et al., 2013b) are used. As such, many word embedding matrices handle punctuation differently. Many word embedding matrices support punctuations and special symbols, in that case one could retain punctuation. The removal of noise from the textual data results in the most informative and meaningful features. Figure 4.7 demonstrates the text pre-processing steps discussed above.

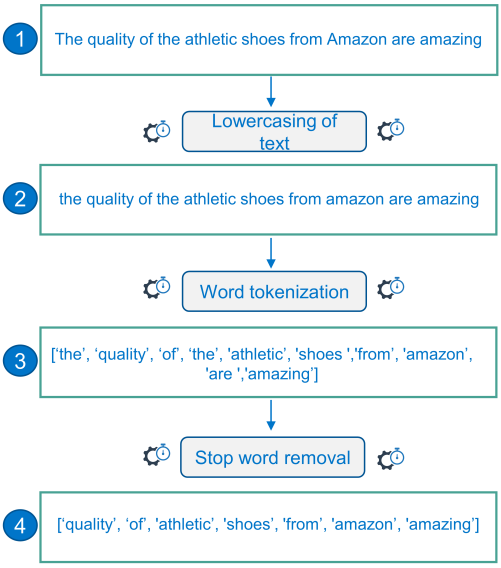

Figure 4.7: Text pre-processing components

For the datasets used within this research, the text pre-processing steps illustrated in Figure 4.6 will be applied to the *review* column. This is effectively the opinion expressed by the Amazon customer based on a product purchased within a specific product category. After the noise has been removed from the textual data, the most informative and meaningful features remain.

Synset enrichment

Given that the aim of this research is assessing whether deep learning can have a positive impact on WSD within the domain of sentiment analysis, words or tokens that are going to add some level of ambiguity to a review or a given text need to be considered. The process of ascertaining words that allow for ambiguity will be conducted after the tokenization is complete. For this research it was considered to look at reviews that contain words which are linked to multiple synonymous words.

In order to determine the synonymous words, synsets were used from the WordNet database. WordNet is a lexical database which has been tailored for NLP applications. In essence, WordNet resembles a thesaurus, in that words are grouped together based on their meanings. WordNet is not a traditional dictionary. WordNet supersedes an ordinary dictionary/thesaurus as it provides more powerful capabilities (Miller, 1995). WordNet focuses on the relationship between words along with their definitions, this makes WordNet a network instead of a list.

In the WordNet network, the words are connected by linguistic relations. These linguistic relations (e.g., hypernym, hyponym, meronym, holonym), are an important component of WordNet. One of the key components that WordNet offers are synsets. A synset can be defined as a set of words that are interchangeable in a certain context. For example, the set {house, home, building} form a synset since the words can be used interchangeably referring to the concept of real estate. The synsets are in essence a grouping of various parts of speech linked to a word. Table 4.3 provides a mapping description of the parts of speech categorizations encompassed within WordNet. The tag refers to the abbreviated form of the parts of speech.

Table 4.3: Wordnet synset parts of speech with the associated tag abbreviation

Parts of speech	**Tag**
Noun	n
Verb	v
Adjective	a
Adjective Satellite	s
Adverb	r

Considering the aim of this research, which is effectively assessing whether deep learning can have a positive impact on WSD within the domain of sentiment analysis, synsets can aid in understanding levels synonymous words which can be viewed as a proxy for ambiguity. Hence, allowing for the application of WSD to be explored within this research.

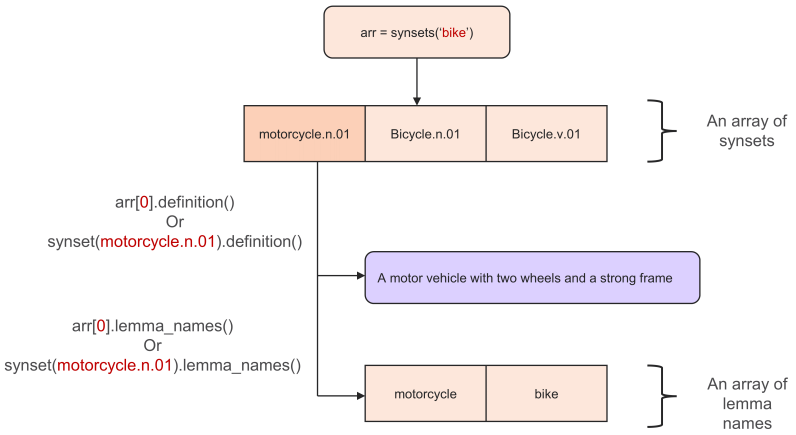

Figure 4.8: Synsets of the word bike
(adapted from Edpresso (2021))

Figure 4.8 showcases examples of a synset linked to the word "bike". It is evident that there are multiple meanings associated with the word bike. As such, it could be represented as a motorcycle (noun), a bicycle (noun) or bicycle (verb). These are represented as three synsets in the WordNet database with the following unique names: motorcycle.n.01, bicycle.n.01 and bicycle.v.01 (Edpresso, 2021).

An array of lemma names that share the same concept are associated with each synset. Thus, motorcycle.n.01 has the words 'motorcycle' and 'bike'. It is evident that 'bike' must be present in all three synsets, with each synset representing a different meaning.

Based on this, all reviews in the input datasets must contain at least one word with a synset greater than one and the desired word must occur in all review categories. This simply means if a word like "small" which has multiple synsets appears within the software and fashion Amazon review datasets, this word will be flagged as a favourable word or token and thus the corresponding review will be considered as an input into the deep learning models. Furthermore, this ensures that some level of loose ambiguity through the multiple meanings of words are included.

Conversely, if there are reviews which do not contain overlapping words that have a synset greater than one, then these reviews will not form part of the final input dataset. The rationale behind this is to ensure that the implementation of the deep learning models are consistent throughout various product categories and that the disambiguation of words can be effectively explored. Overlapping words can have different meanings in different product categories or contexts.

As such, it would make for a more effective analysis to consider only those loosely ambiguous words that feature in reviews for all product context categories used. This further creates a level of assurance that sentences with loosely ambiguous words are considered to demonstrate the concept of WSD and ultimately test the effect of the deep learning models on WSD. Whereas, in the case of including reviews or sentences that do not contain any evidence of ambiguity the model might render the concept of investigating WSD null and void.

Figure 4.9 diagrammatically demonstrates the process of the overlapping word selection. In this example the words 'light' and 'heavy' occur in all three contexts and as such reviews that contain these words, will be selected as part of the input data.

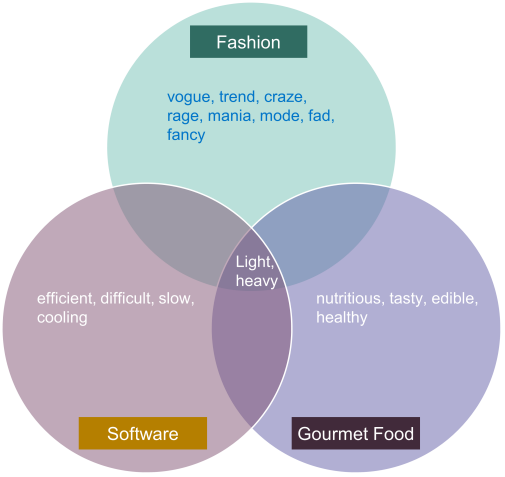

Figure 4.9: Overlapping word selection

4.1.3 Data Exploration

This section focuses on a basic exploratory view of the normalized data after the text pre-processing has been conducted. The data exploration allows for a deeper insight into the data before the predictive modelling stage of this research.

4.1.3.1 Target Class Balance

For classification problems, understanding the distribution of the target classes is integral. This distribution provides a guiding factor on whether an intervention needs to occur in order to adjust for the class imbalance if one does exist. An imbalance occurs when there is a difference in the proportions of the various classes within the training dataset (Kuhn et al., 2013).

The majority class refers to a class/classes with ample datapoints. In contrast to this, the minority class refers to a class with few datapoints (Fernández et al., 2018). Sampling techniques are commonly used in order to bridge the gap between the balance of the target classes (Colton and Hofmann, 2019). These include oversampling, under sampling, Synthetic Minority Over-sampling (Chawla et al., 2002) are but to name a few.

In addition to this, understanding the spread of the target classes further allows for an appropriate choice of statistical evaluation metrics for the predictive model (detailed in Section 4.4). Table 4.4 and Table 4.5 provide a frequency view of the target class proportional split between the negative and positive class once all the necessary text pre-processing has been conducted (detailed in Section 4.1.2.4) for two and three context category applications. It must be noted that these frequency tables focus on the full dataset prior to the split of training, validation and test.

Table 4.4: Target class frequency table - 2 categories

Target class	**Total count**	**Proportion (%)**
Positive	11378	73
Negative	4294	27
Total	**15672**	

Table 4.5: Target class frequency table - 3 categories

Target class	**Total count**	**Proportion (%)**
Positive	57355	82
Negative	12400	18
Total	**69755**	

It is evident from Table 4.4 and Table 4.5 that there is an imbalance between the two sentiment categories, with the positive sentiment class being representative of the majority class whilst the negative sentiment class represent the minority class of the data. Although an imbalance exists between the two classes, no sampling methods were utilized in order to balance the majority and minority classes. This is due to the following reasons:

- Oversampling of the minority class would effectively result in creating duplicates of negative reviews which makes overfitting likely (Weiss et al., 2007).
- The disadvantage with under sampling of the majority class is that the positive reviews would be largely discarded and in turn resulting in the loss of potentially useful data.

As such the initial class imbalance showcased in Table 4.4 and Table 4.5 will be used, however the appropriate performance metrics will be used upon the evaluation of the deep learning methods. This will be described in detail in Section 4.4.

4.1.3.2 Dataset Partitioning

This section describes the overall input dataset in terms of the predictive modelling partitions. The reason for the emphasis on understanding the split proportions is due to the fact that the training dataset proportion size affects the learning ability of the machine learning algorithm. If a small proportion of data is allocated towards the training set, then the machine learning algorithm could have insufficient datapoints to learn effectively. Furthermore, if there is insufficient data allocated toward the validation dataset, then the accuracy, recall, precision and F1 score could yield a large variance.

On a general scale, input data is split into 3 main components: training, validation and test (James et al., 2013). Figure 4.10 provides a diagrammatic visualization of a split on an input dataset.

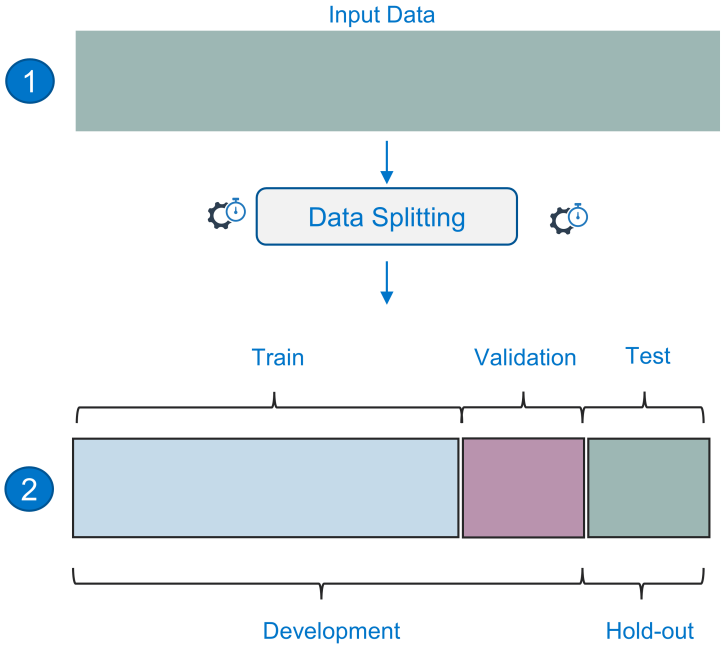

Figure 4.10: Predictive modelling data partitioning process

Training Dataset

This refers to the actual dataset in which the predictive model is trained or fit on i.e., the data in which the model conducts the learning process and optimizes the model parameters. The model performance would not be evaluated based on the training dataset as this would result in a biased measure of evaluation. Therefore, the predictive model is evaluated on the hold-out sample in order to ensure that an unbiased estimate of the model's performance is generated.

Validation Dataset

The validation set is used to fine-tune the predictive model hyperparameters in order to provide an optimal model selection. This dataset is occasionally "seen" by the model, but never does the model actually "learn" from this dataset. The validation set indirectly affects a model, as this dataset is only used within the development stage of the model.

Test Dataset

The test dataset, also referred to as the "hold-out" dataset is used to provide an unbiased evaluation of the final model fit on the training dataset. The evaluation is considered unbiased as the data is unseen to the model and it is generally well curated. The utilization of this set of data is only applicable once the model has been fully trained.

For this research a 60:20:20 split was used for the training, validation and test data respectively using a simple random sampling approach (Reitermanova, 2010). This can be viewed as 80% of the input data allocated for development and 20% of the input data allocated to the hold-out. This stems from the fact that Raghavan (2006) demonstrated that the 80:20 data splitting ratio proved to yield superior outcomes in the machine learning context as opposed to other data splitting ratios. Table 4.6 summarizes the data splitting ratios used for the predictive modelling data set-up for this research. The above-mentioned set the basis for the model experimental set-up discussed in Section 4.3.

Table 4.6: Predictive modelling dataset split partitioning

Dataset Partition	**Split Proportion (%)**
Training	60
Validation	20
Test	20

4.1.3.3 Intra-word Similarity

Based on the assumption that was made within this research that reviews only consisting of ambiguous words (words having more than one sense) were used as part of the predictive model build process. Words deemed as ambiguous (as detailed in Section 4.1.2.4) were only selected if they formed part of each of the context datasets included. As such this, methodological approach resulted in a total of *993* and *973* words for the two category context and three category context applications respectively.

Table 4.7 provides a subset view of the overlapping research defined ambiguous words encompassed within the reviews across the fashion, software and gourmet food datasets.

Table 4.7: Overlapping words and their synset relation

Overlapping Words	Noun	Verb	Adjective	Adjective Satellite	Adverb	Number of Senses
small	2	0	1	9	1	13
dark	5	0	2	9	0	16
edge	6	4	0	0	0	10
bright	6	0	1	3	1	11
ripped	0	4	0	0	0	4

4.2 GloVe Embedding Application

As detailed in Section 3.3.1.1 GloVe pre-trained word embeddings were utilized as part of this research in order to provide a vectorized representation of words that models the actual and semantic meaning of the words within the input dataset. This is encompassed in both the embedding matrices and embedding layers of the desired deep learning algorithms applied within this research.

Whilst Section 3.3.1.1 provides an example of the workings of GloVe on a general scale, to better position this in relation to this research consider the following example review relating to fashion: "Great fashion range, I love shoes and I love fashion".

To demonstrate this practically, consider the following sentence in a corpus: "Great fashion range, I love shoes and I love fashion". The co-occurrence matrix relating to this sentence is depicted in Table 4.8.

Table 4.8: Word co-occurrence matrix example

	Great	**fashion**	**range**	**I**	**love**	**shoes**	**and**
Great	0	1	0	0	0	0	0
fashion	1	0	1	0	1	0	0
range	0	1	0	0	0	0	0
I	0	0	0	0	2	0	1
love	0	1	0	2	0	1	0
shoes	0	0	0	0	1	0	1
and	0	0	0	1	0	1	0

As highlighted in Section 3.3.1.1 establishing a metric that measures semantic similarity between words from the co-occurrence matrix is key. For illustration purposes, consider the word "fashion". The conditional probabilities of the word "fashion" with respect to its paired words in the given sentence is shown below:

- $P(great|fashion) = 1$
- $P(fashion|love) = 0.5$

Drawing from the knowledge presented in Section 3.3.1.1, these standalone conditional probabilities will not be sufficient to provide inference on the relevance of the words. As such the ratio of the probabilities are computed as follows :

- $P(great|fashion)/P(fashion|love) = 1 \ / \ 0.5 = 2$

Given that the output of the ratio of conditional probabilities is greater than 1, it can be inferred that there is a greater level of relevance between the word "fashion" and "great" than to the word "love". Similarly, if the probability ratio was closer to 1, then it could be inferred that both the words "love" and "great" have a greater relevance to the word "fashion". These statistics allow for relationships between words to be built, which provides a practical illustration of the concept of GloVe embedding.

4.3 Model Experimental Set-up

This section provides detailed insight into the model experimental set-up which is focused on the varying architectures of the deep learning models in order to make effective comparisons between models and ultimately understand the impact of context within the sentiment analysis domain.

Given that the focus of this research is to understand whether deep learning provides the capability of disambiguating words given a particular context, it is vital that a variety of deep learning parameters are varied in order to obtain the optimal solutions based on the out of sample statistical performance.

It must be noted that given the limited research available on the exact applications conducted within this research there were no defined performance benchmarks that could be utilized in order to make a standardized comparison on the effectiveness of the models calibrated. As such, a novel methodology is proposed to cater for this and will be unpacked as this section unfolds.

The implementation of the deep learning experiments was conducted within the Jupyter notebook Python 3.7 environment. The experimental set up was conducted using a system with the following specifications: ASUS GeForce GTX 1070 ROG STRIX with 8 GB onboard memory on an AMD Ryzen 7 CPU with 32 GB RAM.

The training model configurations for each deep learning implementation is summarized in Table 4.9.

Table 4.9: Experimental training parameters

Parameters	**Parameter Metric**
Number of epochs	20
Batch size	32
Learning rate	0.0001
Loss function	Binary crossentropy
Optimizer	Adam

A batch size of 32 was utilized, this refers to the number of samples of data within the training dataset to work through before the update of internal model parameters. A batch size of 32 has shown to be widely used in the application of sentiment analysis for CNNs. CNNs using a batch size of 32 produced a higher accuracy relative to varying batch sizes tested (Altun and Abdulnabi, 2016).

This further feeds into the next hyperparameter which is an epoch. An epoch indicates the number of passes of the entire training dataset the deep learning algorithm has completed. The selection of 20 was decided upon after a validation analysis was conducted to assess the validation loss at varying epochs.

Binary crossentropy was used as a loss function. This is due to the fact that the problem being modelled is a binary classification problem. As such, binary crossentropy is widely used for binary classification problems (Ruby and Yendapalli, 2020). From an optimization perspective, the Adam optimizer was utilized as it is widely used for NLP sentiment related applications (Ali et al., 2019) and is superior to that of stochastic gradient descent, in terms of the overall training time of algorithms (Kingma and Ba, 2014).

In addition to this a form of regularization called early stopping has been applied to all the deep learning models implemented as part of this research. This stems from the fact that neural networks are prone to overfitting (Prechelt, 1998). Overfitting occurs when the machine learning model is able to perform well on the training data, however unable to extend this performance to the test data.

The idea behind early stopping is that the predictive model tries to chase the loss function. Given that data is reserved as part of the validation set, as the model continues to be trained, the loss function value on the validation dataset is kept on record, and when no improvement in performance exists the training is stopped at that desired epoch, hence not going through all epochs.

Many studies have used early stopping to alleviate the problem of overfitting within neural network applications (Kim, 2014; Zhang et al., 2021). As part of the preamble of this research it was also shown that early stopping reduced the overfitting phenomenon in 1D CNNs. As such, early stopping was used within the training process of the deep learning algorithms. For the implementation of early stopping validation loss is the performance measure that was monitored in order to end training.

Furthermore, a learning rate is established in order to qualify improvement. This provides a minimum value of the absolute change from epoch to epoch in the tracked metric. The learning rate which governs the rate at which an algorithm updates the parameter estimates is set to 0.0001. The reason for this was due to the fact that lower learning rates have shown to scale well for large amounts of data and produce a higher maximum generalization accuracy (Wilson and Martinez, 2001).

This Section is further divided into 2 sub-sections as follows:

- **Baseline Model Comparison** - This sub-section looks at a base level comparison between the deep learning models selected in order to provide a preambular understanding of the effect of deep learning on WSD in predicting the correct sentiment category.
- **Model Parameter Tuning** - This sub-section takes into consideration the hyper parameter tuning of the context-aware superior model after the baseline comparison has been performed. This would allow further assessment if improvements can be made to the chosen deep learning model for WSD in the domain of sentiment category prediction.

4.3.1 Baseline Model Comparison

The baseline model comparison refers to a base-level comparison between the 1D CNN and LSTM models in terms of the performance metrics detailed in Section 4.4. As part of the baseline comparison three deep learning models were calibrated; two corresponding to the CNN's and one model relating to LSTMs. These models were run over a 10 run-sample in order to have a clearer understanding of the statistical performance of these models as opposed to limiting the analysis to a single which could cause distortion and biased results.

Table 4.10 provides a high-level description of the baseline for each deep learning category. Each model is tagged with a unique model ID for ease of reference and articulation of results. In addition to this, Figure 4.11 and 4.12 provides an architectural view of the 1D baseline CNN models whilst Figure 4.13 provides an architectural view of the baseline LSTM model.

Table 4.10: Baseline deep learning model descriptions

Model ID	Model Description
cnn_no_context_1d_seq_0	This refers to the baseline sequential 1D CNN, without context as an input.
cnn_context_1d_2b_0	This refers to the baseline 2-branch 1D CNN context-aware model.
lstm_context_2b_0	This refers to the baseline 2-branch LSTM context-aware model.

1D CNN Baseline architectures

cnn_no_context_1d_seq_0

The proposed 1D CNN baseline model architecture without context as an input is diagrammatically illustrated in Figure 4.11.

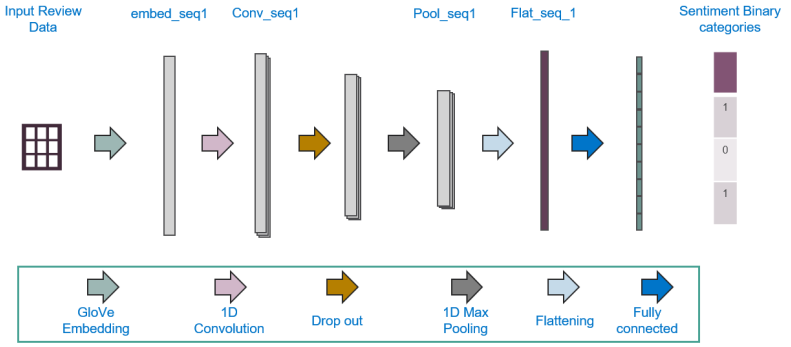

Figure 4.11: `cnn_no_context_1d_seq_0` network architecture

The description of the architecture is articulated by the following points:

- This neural network is considered to be a 1 branch neural network model, in that it only has one branch of input which is the Amazon review and the output value is the sentiment result either 1 (positive) or 0 (negative).
- The Amazon review input is followed by a GloVe pre-trained embedding layer (*embed_seq1*). This allows for words to be represented by dense vectors. A vector is a representation of a word in a continuous vector space. The words that surround a particular word allows for the position of a desired word with in a vector space to be learned.
- The embedding layer feeds into a single 1D convolution layer (*Conv_seq1*) with a filter size of 128, a kernel size of 5 and an ReLU activation function. The choice of the 128 filter size stems from Van Han et al. (2020) in which promising results were attained using this filter size. In addition to this, the choice for ReLU as opposed to sigmoid is due to the fact that ReLU has a faster convergence rate than sigmoid activation functions (Nair and Hinton, 2010).

- A dropout layer (*drop_seq_cnn1*) having a threshold of 0.5 was applied after the 1D convolution (Labach et al., 2019). As discussed previously, dropouts aid in preventing overfitting. The dropout rate determines whether a neuron drops the output or not in a specific layer.
- After dropout has been applied, max pooling was initiated with a pool size of two. This allows for the reduction in the dimensionality of a given mapping while highlighting the prominent features. The maximum pooling layer further aims in reducing overfitting (Srivastava et al., 2014).
- The output of the convolution layers may have a depth greater than one. The Flattening layer (*flat_seq_cnn1*) concatenates the output from the convolution layers to form a flat structure which can then be fed as input to fully connected network.
- The flattening layer feeds into a dense fully connected layer with a ReLU activation function.
- Lastly, the fully connected layer feeds into the output layer in which the outcome is a binary class i.e., 1 denoting a positive sentiment and 0 a negative sentiment. The sigmoid activation function is used due to the binary nature of the output. Furthermore, a probability threshold of 0.5 was applied, which implies that any observation with a probabilistic outcome greater than 0.5 will be classified as a positive class.

In order to make an evaluation of the effect of deep learning on WSD the model `cnn_context_1d_2b_0` is calibrated. This is the baseline 1D CNN model in which context is incorporated as a second input into the network architecture.

cnn_context_1d_2b_0

The baseline 1D CNN context-aware model shown in Figure 4.12 is very similar to the baseline CNN without context (`cnn_no_context_1d_seq_0`), with the only difference being 1 additional branch of input being the context (*input_2_cnn*). Furthermore, the context input layer is concatenated with the output from the flattening layer (*flat1*).

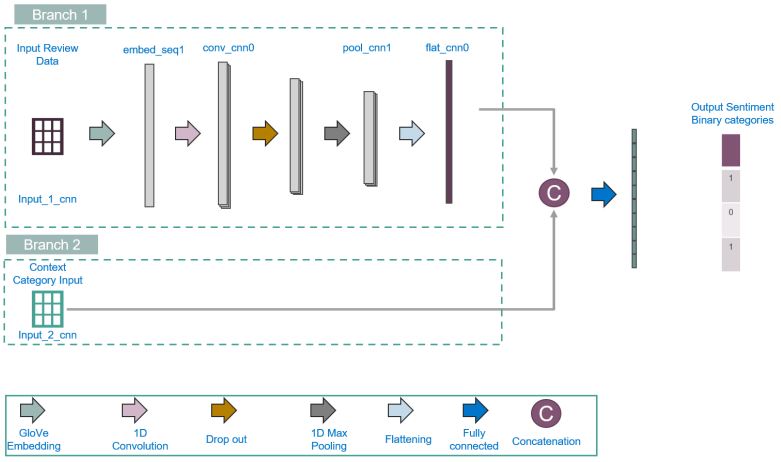

Figure 4.12: `cnn_context_1d_2b_0` network architecture

The inclusion of context as an input allows for the following:

- To assess whether incorporating context into the overall model's effectiveness from a statistical performance perspective. In addition to this it allows for assessment on whether the deep learning model is able to disambiguate words given a particular context category.
- To have a baseline benchmark model to compare against upon refinement or hyper tuning of the model.

In order to add flexibility in terms of the model evaluation, an LSTM context-aware model `lstm_context_2b_0` was calibrated to assess the effectiveness of the algorithm within WSD and make a further comparison with the 1D CNN context-aware model (`cnn_context_1d_2b_0`).

LSTM Baseline architectures

The proposed LSTM baseline context-aware model architecture is diagrammatically illustrated in Figure 4.13.

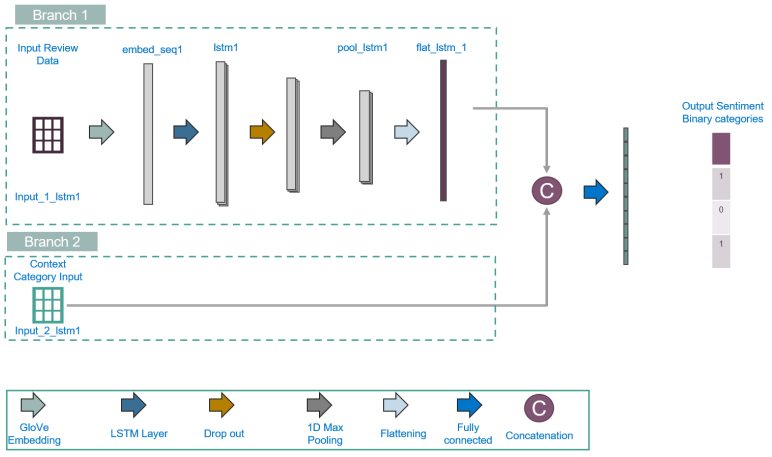

Figure 4.13: `lstm_context_2b_0` network architecture

This model is very much similar to the baseline 1D CNN context-aware model (`cnn_context_1d_2b_0`), with the only difference being the replacement of the 1D convolutional layer with that of the LSTM layer (*lstm_1*). This LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. These memory blocks house one or more recurrently connected memory cells which allow for the write, read and reset operations on the input text data (Graves and Schmidhuber, 2005).

This LSTM layer has a filter size of 128 and a recurrent dropout of 0.5 is applied. As demonstrated earlier dropout removes some elements of one layer of input at random. A common and important tool in RNNs is a recurrent dropout. This refers to the removal of inputs between time steps as opposed to inputs between layers as seen in traditional dropout methods. Just as with regular dropout, recurrent dropout has a regularizing effect and can prevent overfitting hence the rationale for including this within the LSTM layer.

From a comparative perspective the following steps were followed:

- A comparison between the baseline 1D CNN models was conducted across all performance metrics detailed in Section 4.4 after a 10 run-sample. This refers to comparing model id's `cnn_context_1d_2b_0` with `cnn_no_context_1d_seq_0`. This will provide an effective assessment of performance between a context-aware model and a model without context. This comparison will aid in under-

standing whether having the context present as an input does in fact improve the deep learning process and ultimately whether the deep learning 1D CNN model is able to disambiguate words within a review given the context.

- Given that the previous step focuses on understanding the performance of the deep learning algorithms within the same method i.e., between baseline 1D CNN models, a transitioning was made to compare the 1D CNN baseline context-aware model's performance with that of the LSTM's model's performance in order to further assess the performance from a statistical and efficiency perspective across different deep learning methods.

4.3.2 Parameter Model Tuning

After a detailed baseline performance review was conducted between the CNN and LSTM deep learning models, hyper parameter tuning was performed on the baseline context-aware model. Hyperparameters are of great importance as they have direct control on the behaviour of the training algorithm and contribute significantly to the performance of the model that is being trained. The choice of appropriate hyperparameters are crucial in determining the success of the neural network architecture. Without the effect of hyperparameter tuning, sub-optimal results will be produced (Zheng et al., 2019). In this section the following model hyper parameters are explored:

4.3.2.1 Dense Layer Addition – cnn_context_1d_2b_1

The `cnn_context_1d_2b_1` model incorporates the enrichment of the baseline `cnn_context_1d_2b_0` model in that a dense layer with 10 filters and a ReLU activation function is added at the outset of the second branch of 2-branch 1D CNN. The rationale behind this is essentially that incorporating an additional dense layer could potentially learn features from all the combinations of the features of the previous layer, thus offering a richer network.

4.3.2.2 Dropout Percentage Adjustment – cnn_context_1d_2b_2

As discussed earlier, dropout aids in reducing the statistical noise in the data upon training the deep learning model and in turn aids in reducing overfitting. The dropout percentage specifies the probability at which outputs of the layer are dropped out and within the `cnn_context_1d_2b_2` model the dropout percentage is increased from 0.5 to 0.8. The rationale for this is that a threshold of 0.5 is commonly used for retaining the output of each node in a hidden layer. Whereas a value close to 1, such as 0.8, is used for the retaining of inputs within the visible layer (Srivastava et al., 2014).

4.3.2.3 Global Max Pooling – cnn_context_1d_2b_3

After the convolutional layer produces each feature map vector, a one-dimensional global max pooling layer is applied, generating the most important features. This is achieved through the process of taking the maximum value from each feature map vector. Primarily, it is used in this research to reduce the dimensionality of the feature maps which is applied to the last feature map of the feature extractor and furthermore to replace the flattening layer (Christlein et al., 2019). As part of the parameter tuning for this research, the flattening layer is replaced with the global max pooling layer giving rise to a unique model `cnn_context_1d_2b_3`.

4.3.2.4 1D CNN layer Addition – cnn_context_1d_2b_4 and cnn_context_1d_2b_5

For text classification CNN takes advantage of the convolutional layers/filters that allow for the automatic learning of features suitable for the given application. For sentiment classification the convolutional filters within the 1D CNN model may capture semantic features of sentimental expressions (Rios and Kavuluru, 2015).

It has been shown that comparable performance can be achieved through just having a single convolutional layer or a combination of convolutional filters without the need for intricate hyperparameter adjustment (Kim, 2014). As such for this research, the addition of three 1D convolution layers to the context-aware 1D CNN architecture is explored and identified with the model id of `cnn_context_1d_2b_5` as shown in Table 4.11. Table 4.11 provides a consolidated view of the various models that will be implemented as part of the methodology.

Table 4.11: Parameter tuned experimental model descriptions

Model ID	**Method Description**
cnn_context_1d_2b_1	Refers to the enrichment of the baseline cnn_context_1d_2b_0 model with the addition of 1 dense layer to the second branch of the model.
cnn_context_1d_2b_2	Refers to the enrichment of the baseline cnn_context_1d_2b_0 model in terms of increasing the drop out percentage from 0.5 to 0.8.
cnn_context_1d_2b_3	Refers to the enrichment of the cnn_context_1d_2b_2 model by having the flattening layer changed to Global max pooling.
cnn_context_1d_2b_4	Refers to the enrichment of the cnn_context_1d_2b_3 model by incorporating 3 1D CNN layers whilst removing max pooling.
cnn_context_1d_2b_5	Refers to the enrichment of the cnn_context_1d_2b_3 model through the addition of max pooling back into the model whilst having 3 1D CNN layers.

4.4 Performance Measure Analysis Set-up

Performance Metrics and Prediction Threshold

As discussed in Section 3.3.1.5 the prediction threshold governs the way in which the predicted probabilities are converted into predicted classes. For the implementation within this research, a threshold of 0.5 was utilized. This implies that every predicted probability greater than 0.5 will be classified as a positive sentiment and every predicted probability less than 0.5 will be classified as negative. This prediction threshold will form the basis for the classification of predicted classes for the inputs into the performance metrics used to evaluate the implemented deep learning algorithms.

The following performance measures (detailed in Section 3.1.6) will be used to effectively critique and compare the various deep learning techniques; Accuracy, Recall, Precision and F1 score. Even though all of these measures will be reported on, a greater emphasis will be placed on the F1 score performance metric. Since the F1 score is an average of Precision and Recall, it means that the F1 score gives equal weight to Precision and Recall which is beneficial to this research implementation due to the imbalanced nature of the target classes within the input datasets.

Boxplot & Confidence Interval Analysis

Boxplots will be utilized to better position the results attained from performance metrics from a descriptive perspective. This will provide information on the distribution of each of the performance metrics across each of the implemented deep learning algorithms. The aim of the boxplot is to provide a view of how tightly the data is grouped and the level of skewness of the data.

In addition to this, 95% confidence intervals were created for each of the model performance measures; accuracy, precision, recall, F1 score. Given that it is computationally expensive to run the deep learning algorithms over a large number of iterations, a t-statistic confidence is used as opposed to a z-score statistic due to the iteration sample being 10. The confidence interval formula is given by equation 4.1.

$$95\%\ Confidence\ Interval = \bar{x} \pm\ t\frac{s}{\sqrt{n}} \qquad df = n - 1 \tag{4.1}$$

where:

$\bar{x}$ = sample mean
s = sample standard deviation
t = t-statistic confidence level value
n = sample size

Wall Time Efficiency

Efficiency of machine learning algorithms is of great importance when conducting the up-scaling of deep learning implementations. Understanding the time taken to train deep learning algorithms in relation to the out of sample performance allows for the researcher to understand the cost to benefit ratio between implementation training time and statistical performance of the model. This simply means that one can make an informed decision whether the cost (in terms of allocating heavy technical resources) is worth initiating for either a large or small improvement in out of sample performance.

As a by-product of this research the training efficiency of the algorithms listed and described in Section 4.3.2 will be monitored over 10 iterations per model. The metric utilized for this monitoring is the wall time. Wall time refers to the actual time taken for a program to execute its assigned tasks.

Chapter 5

Results and Discussion

This chapter focuses on the presentation and discussion of the results of this work with reference to the aim of this research, which was to solve the problem of WSD within the domain of sentiment analysis using deep learning. The two sub-aims are as follows:

- To assess the effect of including context as an input into the deep learning algorithms on predicting the correct sentiment and ultimately understanding whether the deep learning model is able to disambiguate words given a particular context category.
- Assess the impact of hyper-parameter tuning on the best performing context-aware model, to further understand whether improvements in the out of sample performance can be made to a baseline context-aware model.

5.1 The Effect of Context as an Input in WSD - 2 Categories 1D CNN

The section focuses on discussing the results attained from the comparison between the baseline 1D CNN model without context `cnn_no_context_1d_seq_0` and the baseline context-aware model `cnn_context_1d_2b_0` with two context categories as input (fashion and software). This section is further broken down into two subsections in which Section 5.1.1 focuses on the quantitative evaluation of the analysis whilst Section 5.1.2 presents the practical view to better understand the effect of context on reviews explicitly.

5.1.1 Evaluation

Given that the data utilized for the implementation of the deep learning models is imbalanced in terms of the target classes, the F1 score based on the validation dataset will be used in order to assess the effectiveness of the 1D CNN in WSD given a particular context. The F1 score box-plot presented in Figure 5.1 provides a distributional view of this measure across both baseline models.

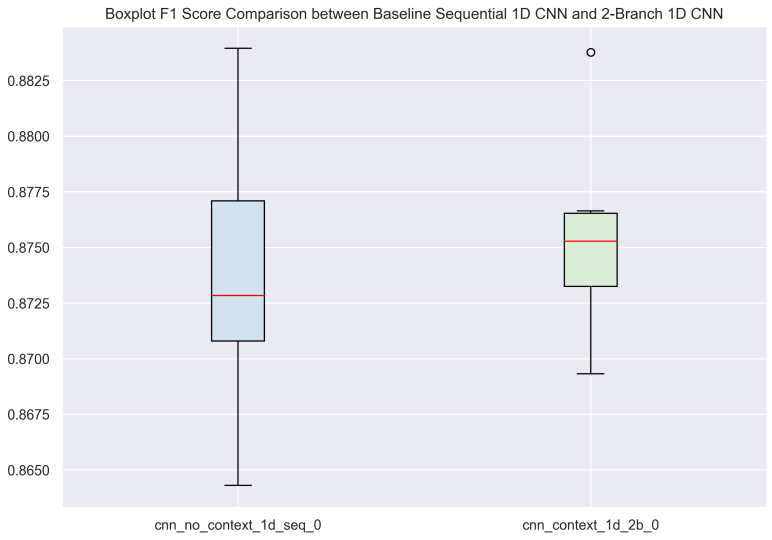

Figure 5.1: F1 score comparison between `cnn_no_context_1d_seq_0` and `cnn_context_1d_2b_0` - 2 categories

As depicted in Figure 5.1, it is evident that on average the context-aware model does perform slightly better than the `cnn_no_context_1d_seq_0` model. It is observed that a slightly higher on average F1 score is exhibited by the `cnn_context_1d_2b_0` model (0.875) than that of the `cnn_no_context_1d_seq_0` (0.874).

The spread of the box-plot for the `cnn_context_1d_2b_0` model is narrow and as such can be seen as more favourable, as this indicates a lesser level of variability within the validation predictions and thus indicative of a more stable and consistent prediction range. The `cnn_context_1d_2b_0` does showcase a higher minimum of the validation F1 score (0.869) than the `cnn_no_context_1d_seq_0` model. This result indicates the context-aware model will always produce a higher minimum F1 score within a 10 iteration range and provides a potential motivation for scaling this to a higher number of iterations.

Even though the `cnn_no_context_1d_seq_0` does exhibit a higher maximum F1 score than the `cnn_context_1d_2b_0`, the box-plot range is quite spread out indicating higher levels of variability. This indicates the stability of the context-aware model and motivational grounds that having context as input allows for the ability of the 1D CNN model to be able to extract the most important and prominent features of the input text in relation to the context category. This ultimately adds to the ability to disambiguate words and predict the correct sentiment.

Given that the F1 score allows a model to be evaluated by taking both the precision and recall into account, it would still be beneficial in analysing other statistical performance measures. Although Accuracy should not be viewed in great detail given the class imbalance of the data, Figure 5.2 provides a box-plot of the validation accuracy between the two baseline CNN models.

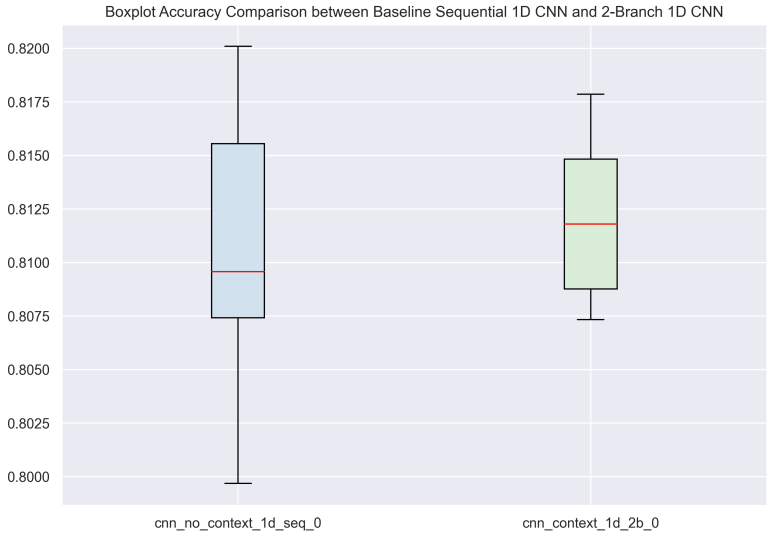

Figure 5.2: Accuracy comparison between `cnn_no_context_1d_seq_0` and `cnn_context_1d_2b_0` - 2 categories

It is evident that the trend shown in the F1 score analysis extends to the accuracy metric, in that the `cnn_context_1d_2b_0` does produce a slightly higher accuracy on average whilst also maintaining a lower minimum F1 score. In addition to this, the box-plot of the context-aware model is much narrow, indicating less room for variability of predictions within a 10 iteration window, thereby further confirming a more stable prediction window.

In order to widen the analysis across additional performance metrics, Table 5.1 provides a summarized view of the t-statistic confidence intervals (described in Section 4.4) over a 10 run iteration for the various statistical performance measures, whilst Table 5.2 provides the wall time exhibited after a 10 run execution.

Table 5.1: Performance evaluation confidence intervals - CNN baseline comparison

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
cnn_context_1d_2b_0	[0.809, 0.815]	[0.834, 0.849]	[0.899, 0.927]	[0.873, 0.878]
cnn_no_context_1d_seq_0	[0.806, 0.814]	[0.835, 0.847]	[0.895, 0.925]	[0.870, 0.877]

Based on the performance evaluation metrics shown in Table 5.1, apart from the F1 score, the context-aware model does provide better performance on both the accuracy and precision metrics as we observe higher lower and upper bounds for these confidence intervals as opposed to the `cnn_no_context_1d_seq_0` model. Although accuracy should not be looked at as an in-depth indicator of performance, it is refreshing to see that the `cnn_context_1d_2b_0` does perform better than the `cnn_no_context_1d_seq_0` model across majority of the performance evaluation metrics.

Table 5.2: Average wall time - CNN baseline comparison

Model	**Wall Time**
cnn_context_1d_2b_0	20min 14s
cnn_no_context_1d_seq_0	19min 56s

In terms of run-time efficiency, it is evident from Table 5.2 that the wall times for the baseline 1D CNN models are fairly similar. The wall time in conjunction with the statistical performance confidence intervals can be viewed as a cost to benefit indicator. In the case of the context-aware model, a higher F1 score & Accuracy is attained on average whilst exhibiting a wall time very close to that of the `cnn_no_context_1d_seq_0` model. Although, these results demonstrate the performance capabilities on two categories, there is motivational grounds to utilize the context-aware model with the addition of more data and more context categories.

5.1.2 Practical Illustration

The analysis presented in Section 5.1.1 provided a quantitative evaluation of the effect of context as an input on WSD, however having a practical illustration of the actual predictions in relation to the input text is integral for a deeper dive into the effectiveness of the deep learning algorithms on WSD. This section will demonstrate the practical evaluation by considering reviews deemed to be ambiguous from the out of sample dataset. In addition to this the predicted probability will be used as a guiding metric to assess the influence of context as an input on WSD. The predicted probability refers to the likelihood of a review being positive. A probability threshold of 0.5 was used to indicate whether the predicted class was positive or negative.

Review 1: "I really thought this was one of those programs I could get a lot of use out of. However, I have used this program so very little that I cannot give a meaningful review to it. From what I've seen of it and had I really got into using it, I can say that its a top shelf program."

Review 1 is rated as a positive review within the Amazon software dataset, whilst being positive it does offer a level of ambiguity as a whole rather than having a word specifically driving the ambiguity. Table 5.3 showcases the associated predicted outcomes for both the context-aware model and the `cnn_no_context_1d_seq_0` model.

Table 5.3: Review 1 prediction comparison

Model	Predicted Probability	Predicted Class
cnn_context_1d_2b_0	0.664	Positive
cnn_no_context_1d_seq_0	0.432	Negative

It is evident that the `cnn_context_1d_2b_0` has a better capability in predicting the overall sentiment, as the model predicts the correct positive sentiment whilst the `cnn_no_context_1d_seq_0` model exhibited a converse prediction. This is refreshing to see as the sentence does provide some level of ambiguity holistically. Furthermore, in terms of the predicted probability the relative strength of prediction is 23% higher for the context-aware model, implying that the prediction was not a minute level over the 50% prediction threshold, but rather a relatively confident prediction. This provides some level of practical confirmation that the inclusion of context does provide value add to the model in terms of being able to disambiguate a sentence given a particular context category.

Review 2: "Have used Microsoft office for many years. This version has enhanced previous versions. Still trying to get used to the "ribbon"."

Table 5.4: Review 2 prediction comparison

Model	Predicted Probability	Predicted Class
cnn_context_1d_2b_0	0.932	Positive
cnn_no_context_1d_seq_0	0.356	Negative

Review 2 has been indicated as a positive review from the Amazon software dataset, however while that may be true in totality, the second part of the third sentence of the review does contribute to some level of ambiguity. In a similar trend to review 1, `cnn_context_1d_2b_0` has a better capability in predicting the correct sentiment as a higher predicted probability is attained, which demonstrates a higher level of confidence in the prediction by the `cnn_context_1d_2b_0` than the `cnn_no_context_1d_seq_0` model. This provides some level of comfort in knowing that the context-aware model is able to associate the words of the above mentioned reviews in the software context category, and potentially extract the most meaningful features in relation to this context.

Review 3 provides an example of sentence ambiguity and a case in which the actual sentiment was negative however both models predicted the sentiment as positive.

Review 3: "I was thrilled to see Amazon present these, as I had wanted the rosetta stone version for years. since I never used rosetta stone, I can't compare, but I will say this was fun, but only for a few weeks. after the newness wears off, it didn't seem much different than the memorization needed by other types of language programs. still, a very inexpensive way to try a language out."

Table 5.5: Review 3 prediction comparison

Model	Predicted Probability	Predicted Class
cnn_context_1d_2b_0	0.743	Positive
cnn_no_context_1d_seq_0	0.864	Positive

Based on the results it is evident, that having context included in the model does have merit in some cases, however there are scenarios in which the context-aware model's prediction does struggle in predicting the correct sentiment. It is important to note that whilst both models predicted the incorrect sentiment, the `cnn_context_1d_2b_0` model has a 12% lower predicted probability than the `cnn_no_context_1d_seq_0`, showcasing to some degree that the context-aware model picks up less of a positive prediction than the `cnn_no_context_1d_seq_0 model`.

Based on the baseline model comparison for two context categories, the context-aware model `cnn_context_1d_2b_0` did showcase relative performance strength when compared to the 1D CNN model without context `cnn_no_context_1d_seq_0`, both from a quantitative as well as practical perspective. This is potentially due to the fact that adding a context layer provides the 1D CNN with the ability to learn the associations of words of an Amazon review related to a particular context and furthermore enabling the network to extract the most important and prominent features in relation to a given context. Although one of the strong advantages of CNN models is in the independence of requiring prior knowledge and human effort in feature design, having multiple features as inputs in CNN text classification models have shown to yield good quality predictions in a reasonable amount of time (Parcheta et al., 2019).

5.2 Context-aware Model Comparison - 1D CNN vs LSTM 2 Categories

Given that the context-aware 1D CNN model has shown promising results when compared to the `cnn_no_context_1d_seq_0` and that both LSTM and CNN models have shown to be widely used methods within WSD sentiment classification applications. Therefore, it would be essential in understanding the comparative performance of these models from a statistical and a computational efficiency perspective.

Additionally, given that limited research and analysis exists for the WSD in the domain of sentiment analysis, this is a further motivational factor for the context-aware model comparison between the `cnn_context_1d_2b_0` and the `lstm_context_2b_0`. Table 5.6 provides a summary of the average of statistics over 10 iterations, whilst Table 5.7 provides an overview of the wall time efficiency.

Table 5.6: Average performance evaluation metrics - 1D CNN vs LSTM

Model	**Average Accuracy**	**Average Precision**	**Average Recall**	**Average F1 score**
cnn_context_1d_2b_0	0.812	0.841	0.913	0.875
lstm_context_2b_0	0.738	0.738	0.999	0.849

It is evident that the 1D CNN context-aware model performs better across the board of the various performance evaluation metrics. The `cnn_context_1d_2b_0` model showcases a 2.6% higher F1 score than the `lstm_context_2b_0` model. Although accuracy should not be viewed as a true measure of performance given the imbalanced nature of the dataset, the `cnn_context_1d_2b_0` does prove to have a

significantly higher average accuracy (0.812) than that of the `lstm_context_2b_0` model (0.738). The 8% difference in performance could potentially be due to the fact that the current implementation within this research focused on the classification of long text, which appeals to the nature of CNNs. This is due to the fact that CNNs can remember much longer sequences as opposed to the LSTMs, this has been shown in text classification by Zhang et al. (2015) and Bai et al. (2018).

Table 5.7: Average wall time - 1D CNN vs LSTM

Model	**Wall Time**
cnn_context_1d_2b_0	20min 14s
lstm_context_2b_0	5h 15min 37s

In terms of execution efficiency, it is evident from Table 5.7 that the `cnn_context_1d_2b_0` model provides a more efficient run-time over a 10 run iteration than that of the `lstm_context_2b_0`. The wall time exhibited by the `cnn_context_1d_2b_0` model is roughly 15 times faster than the `lstm_context_2b_0` model. From a cost to benefit stand-point, the implementation of the 1D CNN context-aware model would seem more favourable than the LSTM context-aware model, as a better out of sample statistical performance is attained within a reasonable execution time-frame.

This could potentially be driven by the fact that CNNs are faster by design, since the computations in CNNs can happen in parallel, while RNNs need to be processed sequentially, since the subsequent steps depend on previous ones. Singh et al. (2017) confirm that the training of CNN networks are much faster than LSTM-based networks. It was as a result of the training time efficiency of CNNs that many fields of research utilize some components of CNNs in their RNN and LSTM applications to create hybrid approaches in order to speed up the overall training process (Bradbury et al., 2016).

Based on the above analysis it was demonstrated that on a baseline comparison using two context categories, the 1D CNN context-aware model has out-performed the LSTM context-aware model in terms of a statistical evaluation as well as in terms of the efficiency of the execution. As such, this provides reasonable motivation and insight in understanding whether the 1D CNN baseline model can be improved upon through hyper-parameter tuning.

5.3 Context-aware Model Comparison - 1D CNN Hyper Parameter Tuning

This section will focus on the validation analysis of various hyper-parameter tuned models as described in Section 4.3.2, in order to assess the effectiveness of certain hyper parameters on the validation performance of the 1D CNN context-aware deep learning algorithm. It must be noted that the baseline `cnn_context_1d_2b_0` model that leveraged off two context categories, was used as the reference point as each suit of hyper-parameters were implemented.

Figure 5.3 provides a box plot of the out of sample F1 score for the various hyper-parameter tuned 1D CNN models. In addition to this, Table 5.8 provides a t-statistic confidence interval spread across the various out of sample statistical evaluation metrics to provide a spread of performance of the models over a 10 run iteration on the validation dataset.

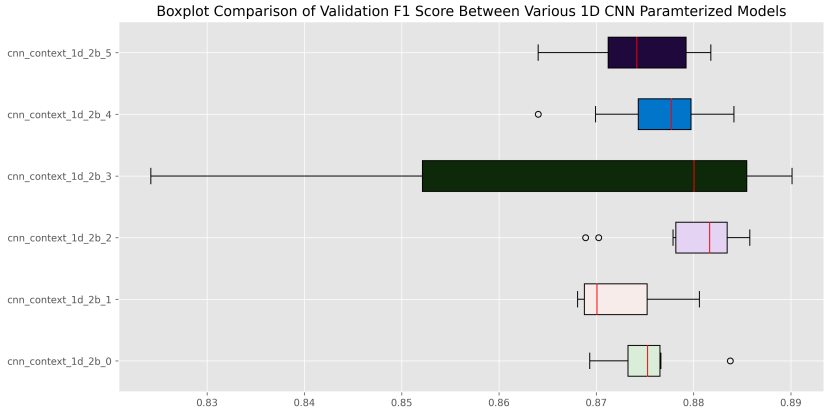

Figure 5.3: F1 score comparison between 1D CNN hyper-parameter tuned models

Based on both Figure 5.3 and Table 5.8 it is evident that the inclusion of an additional dense layer given by model `cnn_context_1d_2b_1` does not really add value in aiding the 1D CNN to better classify sentiments of reviews as a lower on average validation F1 score is observed. Furthermore, the range of the confidence interval is much wider allowing for attaining much lower lower-bound than the baseline context-aware model.

Conversely, model `cnn_context_1d_2b_2` does provide a slightly better performance on average as a 0.006 increase is observed in the upper bound of the F1 score confidence interval. Although increasing the level drop-out does potentially add to

Table 5.8: Performance evaluation confidence intervals - hyper parameter tuned 1D CNN models

Model	Accuracy	Precision	Recall	F1 Score
cnn_context_1d_2b_0	[0.809, 0.815]	[0.834, 0.849]	[0.899, 0.927]	[0.873, 0.878]
cnn_context_1d_2b_1	[0.807, 0.812]	[0.838, 0.856]	[0.883, 0.917]	[0.869, 0.876]
cnn_context_1d_2b_2	[0.814, 0.822]	[0.836, 0.851]	[0.903, 0.936]	[0.875, 0.884]
cnn_context_1d_2b_3	[0.792, 0.827]	[0.846, 0.892]	[0.816, 0.928]	[0.849, 0.886]
cnn_context_1d_2b_4	[0.811, 0.818]	[0.836, 0.857]	[0.890, 0.929]	[0.872, 0.880]
cnn_context_1d_2b_5	[0.812, 0.818]	[0.844, 0.870]	[0.872, 0.916]	[0.871, 0.878]

a better level of regularization to the model, however one needs to be careful on the selection of the dropout rate as removing too much of information and features could potentially harm the 1D CNN from understanding the word to context relations.

It is important to note that by changing the flattening layer to a global max pooling given by model `cnn_context_1d_2b_3`, does yield a higher maximum validation F1 score on average as an increase in the upper bound of the F1 score from 0.878 to 0.884 is observed when comparing the baseline 1D CNN `cnn_context_1d_2b_0` to `cnn_context_1d_2b_0`. This is potentially due to the fact that global max pooling has the advantage of having no parameter to optimize thus avoiding overfitting.

In addition to this, global max pooling represents the whole text as a single vector and further aids in the extraction of the most prominent features after the main features are extracted by the convolutional layers. Although this model does attain a higher upper bound validation F1 score, the range is very wide and as such may not seem to be that favourable, as we would want to minimize the spread in the F1 score and contain it to a smaller window of values. Although this result may seem promising from the maximum F1 score observed. The `cnn_context_1d_2b_3` is plagued with the effect of its validation F1 score having the lowest lower bound F1 score value of 0.849 from the t-statistic confidence interval.

Lastly, having three 1D convolutional layers have shown to not have much of an impact on the 1D CNN model, given by model's `cnn_context_1d_2b_4` and `cnn_context_1d_2b_5` as the result of the F1 score on average seems to be fairly similar to the baseline model `cnn_context_1d_2b_0`. On a positive note, these two models do achieve a higher upper bound validation F1 score than the baseline model. For the application of the 1D CNN algorithm on two context category inputs it seems as if 1 convolutional layer suffices and is optimal.

This is in contrast to the general view that the deeper the network the better the learning process for the CNN (Le et al., 2018b). Given that this analysis was only performed using two context categories it might be beneficial to consider scaling the analysis further for 1D CNNs with multiple convolutional layers with the inclusion of more data and more context categories. This could potentially allow for the 1D CNN to be able to extract more features and learn the relation of these features given a particular context category.

5.4 Context-aware Model Comparison - 3 Categories

In order to assess the effectiveness of the context-aware model further, it is hypothesized that the inclusion of another context category could potentially contribute positively to the out of sample performance and improve the deep learning model's ability to disambiguate words and predict the correct sentiment. The third context-category that was included as part of the data input phase is the Grocery and Gourmet Food category as indicated in Table 4.5. It must be noted that by including the additional context category, the amount of input data has increased overall for this implementation and analysis.

The three context categories assessment was applied to both the `cnn_context_1d_2b_0` and `lstm_context_2b_0` models. Table 5.9 provides the evaluation of the `cnn_context_1d_2b_0` model across the out of sample average F1 score and Accuracy.

Table 5.9: Accuracy and F1 score comparison for varying number of context categories - CNN

Average F1 score	**Average Accuracy**	**Total Categories**
0.875	0.812	2
0.915	0.857	3

It is evident from Table 5.9 that the inclusion of a third context category has improved the performance of the `cnn_context_1d_2b_0` model. It is observed to have exhibited a 4% increase on average F1 score and a 4.5% increase in average Accuracy.

Table 5.10 provides the t-statistic confidence intervals for all the performance evaluation metrics for both two context categories and three context categories, so as to gain an indication of the spread of the statistical performance over a 10 run iteration.

Table 5.10: Performance measure confidence intervals across category sizes - 1D CNN

Accuracy	Precision	Recall	F1 Score	Total Categories
[0.806, 0.815]	[0.835, 0.847]	[0.895, 0.925]	[0.870, 0.878]	2
[0.854, 0.860]	[0.888, 0.898]	[0.928, 0.949]	[0.913, 0.918]	3

It is important to note that with the increase in the number of categories from two to three, the performance of the 1D CNN context-aware model `cnn_context_1d_2b_0`, improves across all performance evaluation metrics. This is evident by the higher lower bound values as well as the higher upper bound values of the t-statistic confidence intervals.

Additionally, it must be noted that with the increase in the number of categories the confidence interval spread narrowed. This is beneficial as it provides some level of comfort to know that the out of sample values will not deviate too far from the mean which allows for more stable out of sample results to be reported on for further investigations.

In order to better demonstrate the impact of context categories on the overall predictive power of the deep learning model, the isolated out of sample performance metrics will need to be calculated for each of the context categories and compared to the cumulative benchmark out of sample performance which encompasses all three context categories. Table 5.11 showcases the isolated average F1 score for each context category in relation to the benchmark. The benchmark represents the performance of the 1D CNN in totality.

Table 5.11: Average F1 score comparison for each context category in relation to the three category benchmark - CNN

Average F1 score	Category
0.915	Benchmark
0.814	Amazon Fashion
0.87	Grocery and Gourmet Food
0.716	Software

Based on Table 5.11 it is evident that context categories holistically add a positive level of performance to the deep learning algorithm as the benchmark F1 score supersedes the F1 scores of all other categories in isolation. This provides a more conclusive view that having an additional context category could potentially allow for the 1D CNN to learn the relations of words in a given context setting, allowing

for the correct sentiment to be output whilst disambiguating the correct word sense. It must be further noted that although an additional context category was included, this was also accompanied by the increase in the amount of data as indicated in Table 4.5.

This is evident in the isolated performance of the Grocery and Gourmet Food in which it showcases the highest isolated F1 score performance of 87% in relation to the Fashion and Software context categories. Grocery and Gourmet Food context category contributes the largest in terms of the data volume. As such, this provides somewhat of an indication, that having a larger amount of data in a given context could potentially alleviate the problem of WSD. This could potentially be due to the fact that there are sufficient amount of features for a given context that enables the network to better associate these features within a given context and produce a correct prediction. While not directly part of this analysis, a similar trend is exhibited for the 1D CNN model without context, shown in Table B.1.

Although having shown that the results for the LSTM context-aware model were not as competitive as the 1D CNN across two context categories, it has been shown in research that LSTM models do scale with more data and more inputs applied to the architecture (Boulmaiz et al., 2020). As such the performance of the `lstm_context_2b_0` model was evaluated for three context categories.

The evaluation of the `lstm_context_2b_0` model follows a similar trend to that of the evaluation of the `cnn_context_1d_2b_0`. Table 5.12 showcases the average out of sample F1 score and Accuracy of the `lstm_context_2b_0` when applied to both two and three category context inputs.

Table 5.12: Accuracy and F1 score comparison for varying number of context categories - LSTM

Average F1 score	**Average Accuracy**	**Total Categories**
0.849	0.738	2
0.903	0.825	3

Whilst the performance of the 1D CNN baseline is still better on average across both the two and three context category input LSTM baseline model, it must be noted that the `lstm_context_2b_0` has shown a significant improvement in both out of sample F1 score and Accuracy. There exists an 8.7% increase in F1 score and a 5.4% increase in Accuracy as the context categories are increased from two to three. This does provide motivational grounds that the LSTM model does scale well in terms of out of sample performance when given additional context for disambiguation.

Lastly, understanding the cost to benefit ratio between efficiency and out of sample performance is key especially when scaling on larger amounts of data and with the addition of more context categories. Table 5.13 showcases the wall time for each of the baseline models at both two and three category implementations after 10 run iterations.

Table 5.13: Wall time across category sizes - 1D CNN and LSTM

Model	**2 Categories**	**3 Categories**
cnn_context_1d_2b_0	20min 14s	48min 5s
lstm_branch_1	5h 15min 37s	10h 54min 25s

It is evident from Table 5.13 that the `cnn_context_1d_2b_0` provides a better cost to benefit ratio as not only does it provide a better out of sample performance than the LSTM as more context categories are added but also yields better efficiency as the wall time is significantly less across both context category numbers than the `lstm_context_2b_0` model. This further provides an indication of the scaling efficiency that the 1D CNN has over the LSTM. This is potentially due to the nature of the algorithm and the architectural efficiency of the algorithm to allow for data to be processed in a parallel as opposed to a sequential form.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The main aim of this research was to investigate the effect of deep learning algorithms on WSD within the domain of sentiment analysis. This gave rise to the objectives (as outlined in Section 1.2). As such, this section will focus on the overall conclusions attained from this research in relation to these key objectives and to the overarching aim.

The effect of context as an input to deep learning models in WSD

Through the experimental results it was shown that the inclusion of context as an input does slightly improve the 1D CNNs ability to better predict the sentiment class when compared to the 1D CNN without context through the observations given by the F1 score confidence interval in that the context-aware 1D CNN would always produce a higher on average F1 score (as demonstrated in Table 5.1).

The assessment of hyper-parameter tuning on the overall model performance

Through the exploration of hyper-parameter tuning of the 1D CNN two category context-aware model, it was shown that the usage of global max pooling allowed for the baseline 1D CNN model to reach a higher F1 score on average. Conversely, the addition of more 1D convolutional layers did not seem to significantly improve the baseline context-aware model's results.

The assessment of model performance between 1D CNN and LSTM context-aware models

It was demonstrated that the 1D CNN context-aware model outperformed the LSTM context-aware model across all statistical performance evaluation metrics. From an efficiency perspective, the 1D CNN algorithms have shown to be significantly faster in terms of wall time than the LSTM context-aware model across both two and three context category applications. This further demonstrates the fact that the 1D CNN context-aware model has a better cost to benefit ratio by being able to balance a good out of sample performance with efficiency.

The effect of the addition of a context category on the overall performance of the model

It was demonstrated that by increasing the amount of context categories from two to three the out of sample F1 score and Accuracy improved on average for both the 1D CNN and LSTM baseline context-aware models (as shown in Table 5.9, 5.10 and 5.11). This further showcased the potential of both algorithms to scale with more data and more context categories. In addition to this, given that sufficiently more data was added when including the third category it does make a compelling case that with the inclusion of more data the network could potentially be able to associate more prominent features with a given context and as such could potentially alleviate the problem of WSD.

Whilst there exists many studies in NLP which focus on WSD in the domain of text classification, there is however a weak link between WSD and sentiment analysis. As such, the findings of this research will aid in providing a base-level of knowledge on future potential exploration and applications for solving WSD within the sentiment analysis domain.

6.2 Recommendations

Recommendations for future work are presented below from ideas and unanswered questions emerging from the findings of this study.

- **Neutral sentiment category:** This research focused on sentiment prediction for a two class problem i.e., positive or negative. While this does create a simplified solution, it does lack in the sense of understanding whether a review is neither positive or negative which is depicted by the neutral category and is representative of a real world scenario. This does however add an additional level of complexity to the problem application, but can potentially create a widened view to test the full strength of the neural network in disambiguating

words given a particular context.

- **Run iteration extension:** Due to the limited computational power and the time taken for the implementation of the neural networks within this research, only 10 iterations were conducted for the evaluation of the out of sample results. As such, extending the number of iterations could potentially yield a better and more consistent spread of performance measures for a more effective evaluation of the deep learning algorithms.
- **Multiple context category inclusion:** As part of this research the maximum amount of categories that were included as part of the input into the deep learning context-aware models was three. Given that it was shown that the inclusion of more context categories does potentially impact positively on the model's out of sample performance, it might be of interest to include more context categories to assess the capability of the deep learning algorithms in more depth.
- **Data volume increase:** Deep learning models have shown to require large amounts of data for better estimations. Based on this, it could be of value to increase the amount of data per context category into the 1D CNN model so as to better extract meaningful features in relation to the wide spread of data and context categories present. With a large amount of data WSD can potentially be ignored if the deep learning model is able to make good enough predictions with the meaningful features extracted from the input data.
- **Word embeddings:** As part of this research, GloVe embeddings (Pennington et al., 2014) was utilized. This approach is a pre-trained approach that has been widely adopted in the NLP research domain. It might prove to be beneficial to create an embedding layer from scratch in order to train the embeddings with respect to the problem at hand. This offers a more specific focus on the desired NLP problem as opposed to having a pre-trained word embedding layer which offers a more generic implementation. In addition to this, a comparative analysis between pre-trained word embeddings and an embedding layer from scratch would be key in understanding the relative impact and potential benefits that one method has over the other by examining the out of sample performance.

- **Feature context association:** Understanding model performance is key from a statistical standpoint, however supplementing this statistical performance with a more descriptive lens can allow for a more in depth analysis of the deep learning algorithm. One way of performing this descriptive analysis is to understand which meaningful features have been selected by the deep learning algorithm in relation to a specific context. This paints a rich picture of the results from a descriptive perspective, as it provides a more practical illustration of the learning and feature extraction process of the deep learning algorithm.

Whilst WSD has been explored extensively in a variety of NLP domains, there exists limited research within the domain of sentiment analysis. As such, the outcomes of this research has demonstrated a good base-level of knowledge of the impact of deep learning in the area of WSD within the sentiment analysis domain. Furthermore, having the findings of this research enriched with the elements from the listed recommendations could potentially expand upon and provide widened contributions to a number of industries considering research of WSD within sentiment analysis on text data.

Appendices

Appendix A

Amazon Product Reviews Metadata

Table A.1: Amazon Product Reviews Metadata

Attribute	**Description**
reviewerID	Unique ID of the reviewer.
asin	Unique ID of the product.
reviewerName	Name of the reviewer.
vote	Helpful votes of the review.
style	A dictionary of the product metadata, e.g., "Format" is "Hardcover".
reviewText	Text of the review
overall	The overall Rating of the product. This rating ranges between 1-5, 1 implying the worst experience and 5 the best experience
summary	A summary of the review.
unixReviewTime	Time of the review (unix time).
reviewTime	Time of the review (raw).
image	Images that users post after they have received the product.

Appendix B

Context Category F1 score Performance Relative Comparison - cnn_no_context_1d_seq_0

Table B.1: Average F1 score comparison for each context category in relation to the three category benchmark - `cnn_no_context_1d_seq_0`

Average F1 score	**Category**
0.912	Benchmark
0.828	Amazon Fashion
0.861	Grocery and Gourmet Food
0.715	Software

Bibliography

Abid, F., Alam, M., Yasir, M. and Li, C. (2019), 'Sentiment analysis through recurrent variants latterly on convolutional neural network of twitter', *Future Generation Computer Systems* **95**, pp. 292–308.

Aggarwal, C. C. et al. (2018), 'Neural networks and deep learning: A textbook', *Springer* **10**.

Aghdam, H. and Heravi, E. (2017), *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*.

Agirre, E. and Edmonds, P. (2007), *Word Sense Disambiguation: Algorithms and Applications*, Vol. 33, Springer Science & Business Media.

Agirre, E. and Rigau, G. (1995), 'A proposal for word sense disambiguation using conceptual distance'.

Ain, Q. T., Ali, M., Riaz, A., Noureen, A., Kamran, M., Hayat, B. and Rehman, A. (2017), 'Sentiment analysis using deep learning techniques: a review', *Int J Adv Comput Sci Appl* **8**(6), p. 424.

Akkaya, C., Wiebe, J. and Mihalcea, R. (2009), Subjectivity word sense disambiguation, *in* 'Proceedings of the 2009 conference on empirical methods in natural language processing', pp. 190–199.

Albayaty, B. F. Z. and Joshi, S. (2014), Empirical implementation decision tree classifier to wsd problem, *in* 'International Conference on Emerging Trends Science and Cutting Edge Technology (ICETSCET), YMCA', Vol. 28.

Ali, M., Sarowar, M., Rahman, M. L., Chaki, J., Dey, N. and Tavares, J. (2019), 'Adam deep learning with som for human sentiment classification', *International Journal of Ambient Computing and Intelligence* **10**, 92–116.

Altun, O. and Abdulnabi, N. (2016), 'Batch size for training convolutional neural networks for sentence classiiication'.

Arel, I., Rose, D. C. and Karnowski, T. P. (2010), 'Deep machine learning-a new frontier in artificial intelligence research [research frontier]', *IEEE computational intelligence magazine* **5**(4), pp. 13–18.

Artstein, R. and Poesio, M. (2008), 'Inter-coder agreement for computational linguistics', *Computational Linguistics* **34**(4), pp. 555–596.

Baccianella, S., Esuli, A. and Sebastiani, F. (2010), Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining., *in* 'Lrec', Vol. 10, pp. 2200–2204.

Bai, S., Kolter, J. Z. and Koltun, V. (2018), 'An empirical evaluation of generic convolutional and recurrent networks for sequence modeling'.

Bengio, Y., Courville, A. and Vincent, P. (2013), 'Representation learning: A review and new perspectives', *IEEE transactions on pattern analysis and machine intelligence* **35**(8), pp. 1798–1828.

Bhavitha, B., Rodrigues, A. P. and Chiplunkar, N. N. (2017), Comparative study of machine learning techniques in sentimental analysis, *in* '2017 International conference on inventive communication and computational technologies (ICICCT)', IEEE, pp. 216–221.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg.

Bloehdorn, S. and Hotho, A. (2004), Text classification by boosting weak learners based on terms and concepts, *in* 'Fourth IEEE International Conference on Data Mining (ICDM'04)', IEEE, pp. 331–334.

Boulmaiz, T., Guermoui, M. and Boutaghane, H. (2020), 'Impact of training data size on the lstm performances for rainfall–runoff modeling', *Modeling Earth Systems and Environment* **6**, pp. 2153–2164.

Bradbury, J., Merity, S., Xiong, C. and Socher, R. (2016), 'Quasi-recurrent neural networks'.

Branco, P., Torgo, L. and Ribeiro, R. (2015), 'A survey of predictive modelling under imbalanced distributions'.

Brownlee, J. (2020), 'How to calculate precision, recall, and f-measure for imbalanced classification', `https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/`. (Accessed: 2021-06-30).

Carpuat, M. and Wu, D. (2005), Word sense disambiguation vs. statistical machine translation, *in* 'Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)', pp. 387–394.

Chan, Y. S., Ng, H. T. and Chiang, D. (2007), Word sense disambiguation improves statistical machine translation, *in* 'Proceedings of the 45th annual meeting of the association of computational linguistics', pp. 33–40.

Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002), 'Smote: synthetic minority over-sampling technique', *Journal of artificial intelligence research* **16**, pp. 321–357.

Chen, P., Ding, W., Bowes, C. and Brown, D. (2009), A fully unsupervised word sense disambiguation method using dependency knowledge, *in* 'Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics', pp. 28–36.

Chen, X., Liu, Z. and Sun, M. (2014), A unified model for word sense representation and disambiguation, *in* 'Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)', pp. 1025–1035.

Christlein, V., Spranger, L., Seuret, M., Nicolaou, A., Král, P. and Maier, A. (2019), Deep generalized max pooling, *in* '2019 International Conference on Document Analysis and Recognition (ICDAR)', IEEE, pp. 1090–1096.

Clough, P. and Stevenson, M. (2004), Cross-language information retrieval using eurowordnet and word sense disambiguation, *in* 'European Conference on Information Retrieval', Springer, pp. 327–337.

Collobert, R. (2011), Deep learning for efficient discriminative parsing, *in* 'Proceedings of the fourteenth international conference on artificial intelligence and statistics', JMLR Workshop and Conference Proceedings, pp. 224–232.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2011), 'Natural language processing (almost) from scratch', *Journal of machine learning research* **12**, pp. 2493–2537.

Colton, D. and Hofmann, M. (2019), 'Sampling techniques to overcome class imbalance in a cyberbullying context', *Journal of Computer-Assisted Linguistic Research* **3**(3), pp. 21–40.

Cumming, G. and Calin-Jageman, R. (2016), *Introduction to the new statistics: Estimation, open science, and beyond*, Routledge.

Da Silva, N. F., Hruschka, E. R. and Hruschka Jr, E. R. (2014), 'Tweet sentiment analysis with classifier ensembles', *Decision Support Systems* **66**, pp. 170–179.

Deng, L. and Li, X. (2013), 'Machine learning paradigms for speech recognition: An overview', *IEEE Transactions on Audio, Speech, and Language Processing* **21**(5), pp. 1060–1089.

Dessì, D., Recupero, D. R. and Sack, H. (2021), 'An assessment of deep learning models and word embeddings for toxicity detection within online textual comments', *Electronics* **10**(7), p. 779.

Dixit, V., Dutta, K. and Singh, P. (2015), 'Word sense disambiguation and its approaches', *CPUH-Research Journal* **1**(2), pp. 54–58.

Edpresso (2021), 'How to use wordnet in python', `https://www.educative.io/edpresso/how-to-use-wordnet-in-python`. (Accessed: 2021-06-30).

Elkahky, A. M., Song, Y. and He, X. (2015), A multi-view deep learning approach for cross domain user modeling in recommendation systems, *in* 'Proceedings of the 24th international conference on world wide web', pp. 278–288.

Esuli, A. and Sebastiani, F. (2006), Sentiwordnet: A publicly available lexical resource for opinion mining.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B. and Herrera, F. (2018), 'Learning from imbalanced data sets', **10**.

Festag, S. and Spreckelsen, C. (2017), Word sense disambiguation of medical terms via recurrent convolutional neural networks, *in* 'Health Informatics Meets eHealth', IOS Press, pp. 8–15.

Firmino Alves, A. L., Baptista, C. D. S., Firmino, A. A., Oliveira, M. G. d. and 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.

Fulmari, A. and Chandak, M. B. (2013), 'A survey on supervised learning for word sense disambiguation', *Int. J. Adv. Res. Comput. Commun. Eng* **2**(12), pp. 2278–1021.

Gomes, L. (2014), 'Machine-learning maestro michael jordan on the delusions of big data and other huge engineering efforts', *IEEE spectrum* **20**.

Goodfellow, I. J., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press, Cambridge, MA, USA.

Graves, A. and Schmidhuber, J. (2005), 'Framewise phoneme classification with bidirectional lstm networks', *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* **4**, pp. 2047–2052.

Gupta, S., Namavari, A. and Smith, T. O. (2016), 'Word sense disambiguation using skip-gram and lstm models'.

Haykin, S. (2010), *Neural networks and learning machines, 3/E*, Pearson Education India.

He, R. and McAuley, J. (2016), Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, *in* 'proceedings of the 25th international conference on world wide web', pp. 507–517.

Hinton, G., Srivastava, N. and Swersky, K. (2012), 'Neural networks for machine learning lecture 6a overview of mini-batch gradient descent', *Cited on* **14**(8), 2.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**, pp. 1735–80.

Hornik, K. (1991), 'Approximation capabilities of multilayer feedforward networks', *Neural networks* **4**(2), pp. 251–257.

Huq, M. R., Ali, A. and Rahman, A. (2017), 'Sentiment analysis on twitter data using knn and svm', *International Journal of Advanced Computer Science and Applications* **8**(6), pp. 19–25.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Jeong, B., Yoon, J. and Lee, J.-M. (2019), 'Social media mining for product planning: A product opportunity mining approach based on topic modeling and sentiment analysis', *International Journal of Information Management* **48**, pp. 280–290.

Joopudi, V., Dandala, B. and Devarakonda, M. (2018), 'A convolutional route to abbreviation disambiguation in clinical text', *Journal of biomedical informatics* **86**, pp. 71–78.

Jose, R. and Chooralil, V. S. (2015), Prediction of election result by enhanced sentiment analysis on twitter data using word sense disambiguation, *in* '2015 International Conference on Control Communication & Computing India (ICCC)', IEEE, pp. 638–641.

Kai, R. and Wen, W. S. (2016), 'Applying convolutional neural network model and auto - expanded corpus to biomedical abbreviation disambiguation', *Journal of Engineering Science and Technology Review* **9**, pp. 178–184.

Kang, E. (2017), 'Long short-term memory (lstm): Concept', `https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359`. (Accessed: 2021-05-30).

Kansara, D. and Sawant, V. (2020), Comparison of traditional machine learning and deep learning approaches for sentiment analysis, *in* 'Advanced Computing Technologies and Applications', Springer, pp. 365–377.

Kaur, J. (2014), 'Word sense disambiguation (wsd)', *International Journal For Technological Research In Engineering* **1**(5), pp. 2347–4718.

Ketkar, N. (2017), Stochastic gradient descent, *in* 'Deep learning with Python', Springer, pp. 113–132.

Khan, M. F., Khan, A. and Khan, K. (2013), 'Efficient word sense disambiguation technique for sentence level sentiment classification of online reviews.', *Science International* **25**(4).

Kim, Y. (2014), 'Convolutional neural networks for sentence classification'.

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization'.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**, pp. 1097–1105.

Kuhn, M., Johnson, K. et al. (2013), *Applied predictive modeling*, Vol. 26, Springer.

Kuo, P.-H. and Huang, C.-J. (2018), 'A green energy application in energy management systems by an artificial intelligence-based solar radiation forecasting model', *Energies* **11**, 819.

Labach, A., Salehinejad, H. and Valaee, S. (2019), 'Survey of dropout methods for deep neural networks'.

Le, H. T., Cerisara, C. and Denis, A. (2018b), Do convolutional networks need to be deep for text classification?, *in* 'Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence'.

Le, M., Postma, M., Urbani, J. and Vossen, P. (2018a), A deep dive into word sense disambiguation with lstm, *in* 'Proceedings of the 27th international conference on computational linguistics', pp. 354–365.

LeCun, Y., Bengio, Y. and Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), pp. 436–444.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), pp. 2278–2324.

Lee, Y. K. and Ng, H. T. (2002), An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation, *in* 'Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)', pp. 41–48.

Lesk, M. (1986), Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone, *in* 'Proceedings of the 5th annual international conference on Systems documentation', pp. 24–26.

Likert, R. (1932), 'A technique for the measurement of attitudes.', *Archives of psychology* .

Liu, B. (2012), 'Sentiment analysis and opinion mining', *Synthesis lectures on human language technologies* **5**(1), pp. 1–167.

Madhavan, S. (2017), 'Deep learning architectures', `https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/`. (Accessed: 2021-05-30).

Malik, V. and Kumar, A. (2018), 'Sentiment analysis of twitter data using naive bayes algorithm', *International Journal on Recent and Innovation Trends in Computing and Communication* **6**(4), pp. 120–125.

Mäntylä, M. V., Graziotin, D. and Kuutila, M. (2018), 'The evolution of sentiment analysis—a review of research topics, venues, and top cited papers', *Computer Science Review* **27**, pp. 16–32.

Mathworks (2021), 'What is deep learning? 3 things you need to know', `https://www.mathworks.com/discovery/deep-learning.html`. (Accessed: 2021-05-30).

Mehra, N., Khandelwal, S. and Patel, P. (2002), 'Sentiment identification using maximum entropy analysis of movie reviews'.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013b), 'Efficient estimation of word representations in vector space'.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J. (2013a), 'Distributed representations of words and phrases and their compositionality'.

Miller, G. A. (1995), 'Wordnet: a lexical database for english', *Communications of the ACM* **38**(11), pp. 39–41.

Miller, T., Biemann, C., Zesch, T. and Gurevych, I. (2012), Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation, *in* 'Proceedings of COLING 2012', pp. 1781–1796.

Munot, N. and Govilkar, S. (2015), 'Conceptual framework for abstractive text summarization', *International Journal on Natural Language Computing* **4**, 39–50.

Nair, V. and Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Icml'.

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A. and Wilson, T. (2013), Task 2: Sentiment analysis in twitter, *in* 'Proceedings of the 7th International Workshop on Semantic Evaluation, Atlanta, Georgia'.

Nassirtoussi, A., Aghabozorgi, S., Wah, T. and Ngo, D. (2014), 'Text mining for market prediction: A systematic review', *Expert Systems with Applications* **41**, pp. 7653–7670.

Navigli, R. (2009), 'Word sense disambiguation: A survey', *ACM Comput. Surv.* **41**.

Nithyanandan, S. and Raseek, C. (2019), Deep learning models for word sense disambiguation: A comparative study, *in* 'Proceedings of the International Conference on Systems, Energy & Environment (ICSEE), Kerala, India'.

Pal, A. and Saha, D. (2015), 'Word sense disambiguation: A survey', *International Journal of Control Theory and Computer Modeling* **5**(3), pp. 1–16.

Pamungkas, E. W. and Putri, D. G. P. (2017), Word sense disambiguation for lexicon-based sentiment analysis, *in* 'Proceedings of the 9th International Conference on Machine Learning and Computing', pp. 442–446.

Pang, B., Lee, L. and Vaithyanathan, S. (2002), 'Thumbs up? sentiment classification using machine learning techniques'.

Parcheta, Z., Sanchis-Trilles, G., Casacuberta, F. and Redahl, R. (2019), Multi-input cnn for text classification in commercial scenarios, *in* 'International Work-Conference on Artificial Neural Networks', Springer, pp. 596–608.

Paredes-Valverde, M. A., Colomo-Palacios, R., Salas-Zárate, M. d. P. and Valencia-García, R. (2017), 'Sentiment analysis in spanish for improvement of products and services: A deep learning approach', *Scientific Programming* **2017**.

Pennington, J., Socher, R. and Manning, C. D. (2014), Glove: Global vectors for word representation, *in* 'Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)', pp. 1532–1543.

Pesaranghader, A., Matwin, S., Sokolova, M. and Pesaranghader, A. (2019), 'deepbiowsd: effective deep neural word sense disambiguation of biomedical text data', *Journal of the American Medical Informatics Association* **26**(5), pp. 438–446.

Pesaranghader, A., Pesaranghader, A., Matwin, S. and Sokolova, M. (2018), 'One single deep bidirectional lstm network for word sense disambiguation of text data'.

Pham, D.-H. and Le, A.-C. (2018), 'Learning multiple layers of knowledge representation for aspect based sentiment analysis', *Data & Knowledge Engineering* **114**, pp. 26–39.

Pinto, D., McCallum, A., Wei, X. and Croft, W. B. (2003), Table extraction using conditional random fields, *in* 'Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval', pp. 235–242.

Polyak, B. T. (1964), 'Some methods of speeding up the convergence of iteration methods', *Ussr computational mathematics and mathematical physics* **4**(5), pp. 1–17.

Prabha, M. I. and Srikanth, G. U. (2019), Survey of sentiment analysis using deep learning techniques, pp. 1–9.

Prechelt, L. (1998), Early stopping-but when?, *in* 'Neural Networks: Tricks of the trade', Springer, pp. 55–69.

Preethi, G., Krishna, P. V., Obaidat, M. S., Saritha, V. and Yenduri, S. (2017), Application of deep learning to sentiment analysis for recommender system on cloud, *in* '2017 International conference on computer, information and telecommunication systems (CITS)', IEEE, pp. 93–97.

Qian, J., Niu, Z. and Shi, C. (2018), Sentiment analysis model on weather related tweets with deep neural network, *in* 'Proceedings of the 2018 10th international conference on machine learning and computing', pp. 31–35.

Raghavan, R. (2006), Study of the relationship of training set size to error rate in yet another decision tree And random forest algorithms, PhD thesis, Texas Tech University.

Ramadhani, A. M. and Goo, H. S. (2017), Twitter sentiment analysis using deep learning methods, *in* '2017 7th International annual engineering seminar (InAES)', IEEE, pp. 1–4.

Raunak, V., Gupta, V. and Metze, F. (2019), Effective dimensionality reduction for word embeddings, *in* 'Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)', pp. 235–243.

Reitermanova, Z. (2010), 'Data splitting', **10**, pp. 31–36.

Rentoumi, V., Giannakopoulos, G., Karkaletsis, V. and Vouros, G. A. (2009), Sentiment analysis of figurative language using a word sense disambiguation approach, *in* 'Proceedings of the International Conference RANLP-2009', pp. 370–375.

Rios, A. and Kavuluru, R. (2015), Convolutional neural networks for biomedical text classification: application in indexing biomedical articles, *in* 'Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics', pp. 258–267.

Rong, X. (2014), 'word2vec parameter learning explained'.

Roshanfekr, B., Khadivi, S. and Rahmati, M. (2017), Sentiment analysis using deep learning on persian texts, *in* '2017 Iranian Conference on Electrical Engineering (ICEE)', IEEE, pp. 1503–1508.

Ruby, U. and Yendapalli, V. (2020), 'Binary cross entropy with deep learning technique for image classification', *International Journal of Advanced Trends in Computer Science and Engineering* **9**(10).

Sak, H., Senior, A., Rao, K. and Beaufays, F. (2015), 'Fast and accurate recurrent neural network acoustic models for speech recognition'.

Salas-Zárate, M. d. P., Medina-Moreira, J., Lagos-Ortiz, K., Luna-Aveiga, H., Rodriguez-Garcia, M. A. and Valencia-Garcia, R. (2017), 'Sentiment analysis on tweets about diabetes: an aspect-level approach', *Computational and mathematical methods in medicine* **2017**.

Sanderson, M. (1994), Word sense disambiguation and information retrieval, *in* 'SIGIR'94', Springer, pp. 142–151.

Scherer, D., Müller, A. and Behnke, S. (2010), Evaluation of pooling operations in convolutional architectures for object recognition, *in* 'International conference on artificial neural networks', Springer, pp. 92–101.

Schmidhuber, J. (2015), 'Deep learning in neural networks: An overview', *Neural networks* **61**, pp. 85–117.

Seifollahi, S. and Shajari, M. (2019), 'Word sense disambiguation application in sentiment analysis of news headlines: an applied approach to forex market prediction', *Journal of Intelligent Information Systems* **52**(1), pp. 57–83.

Sharef, N. M., Zin, H. M. and Nadali, S. (2016), 'Overview and future opportunities of sentiment analysis approaches for big data.', *Journal of Computer Science* **12**(3), 153–168.

Shen, D., Wu, G. and Suk, H.-I. (2017), 'Deep learning in medical image analysis', *Annual review of biomedical engineering* **19**, 221–248.

Singh, D., Merdivan, E., Hanke, S., Kropf, J., Geist, M. and Holzinger, A. (2017), Convolutional and recurrent neural networks for activity recognition in smart environment, *in* 'Towards integrative machine learning and knowledge extraction', Springer, pp. 194–205.

Singhal, P. and Bhattacharyya, P. (2016), 'Sentiment analysis and deep learning: a survey', *Center for Indian Language Technology, Indian Institute of Technology, Bombay* .

Sinha, R. and Mihalcea, R. (2007), Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity, *in* 'International conference on semantic computing (ICSC 2007)', IEEE, pp. 363–369.

Sohangir, S., Wang, D., Pomeranets, A. and Khoshgoftaar, T. M. (2018), 'Big data: Deep learning for financial sentiment analysis', *Journal of Big Data* **5**(1), pp. 1–25.

Song, H. A. and Lee, S.-Y. (2013), Hierarchical representation using nmf, *in* 'International conference on neural information processing', Springer, pp. 466–473.

Soni, S. and Sharaff, A. (2015), Sentiment analysis of customer reviews based on hidden markov model, *in* 'Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)', pp. 1–5.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), pp. 1929–1958.

Stokoe, C., Oakes, M. P. and Tait, J. (2003), Word sense disambiguation in information retrieval revisited, *in* 'Proceedings of the 26th annual international ACM

SIGIR conference on Research and development in informaion retrieval', pp. 159–166.

Sumanth, C. and Inkpen, D. (2015), How much does word sense disambiguation help in sentiment analysis of micropost data?, *in* 'Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis', pp. 115–121.

Suppala, K. and Rao, N. (2019), 'Sentiment analysis using naïve bayes classifier', *Int. J. Innov. Technol. Explor. Eng* **8**(8), pp. 264–269.

Tang, D., Qin, B. and Liu, T. (2015), 'Deep learning for sentiment analysis: successful approaches and future challenges', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **5**(6), pp. 292–303.

Ting, S., Ip, W., Tsang, A. H. et al. (2011), 'Is naive bayes a good classifier for document classification', *International Journal of Software Engineering and Its Applications* **5**(3), pp. 37–46.

Toha, S. F. and Tokhi, M. O. (2008), Mlp and elman recurrent neural network modelling for the trms, *in* '2008 7th IEEE international conference on cybernetic intelligent systems', IEEE, pp. 1–6.

Triola, M. F. (2010), 'Bayes' theorem'.

Turian, J., Bergstra, J. and Bengio, Y. (2009), Quadratic features and deep architectures for chunking, *in* 'Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers', pp. 245–248.

Van Den Oord, A., Dieleman, S. and Schrauwen, B. (2013), 'Deep content-based music recommendation', **26**.

Van Han, N., Ulfarsson, M. and Sveinsson, J. (2020), 'Sure based convolutional neural networks for hyperspectral image denoising'.

Vateekul, P. and Koomsubha, T. (2016), A study of sentiment analysis using deep learning techniques on thai twitter data, *in* '2016 13th International joint conference on computer science and software engineering (JCSSE)', IEEE, pp. 1–6.

Vossen, P., Rigau, G., Alegria, I., Agirre, E., Farwell, D., Fuentes, M. et al. (2006), Meaningful results for information retrieval in the meaning project, *in* 'Proc. of the 3rd Global Wordnet Conference', pp. 22–26.

Weiss, G. M., McCarthy, K. and Zabar, B. (2007), 'Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?', *Dmin* **7**(35-41), p. 24.

Wiebe, J. (1990), Identifying subjective characters in narrative, *in* 'COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics'.

Wilson, D. and Martinez, T. (2001), 'The need for small learning rates on large problems', **1**, pp. 115 – 119.

Wilson, T., Wiebe, J. and Hoffmann, P. (2005), Recognizing contextual polarity in phrase-level sentiment analysis, *in* 'Proceedings of human language technology conference and conference on empirical methods in natural language processing', pp. 347–354.

Wiriyathammabhum, P., Kijsirikul, B., Takamura, H. and Okumura, M. (2012), 'Applying deep belief networks to word sense disambiguation'.

Xiong, M., Chen, J., Wang, Z., Liang, C., Zheng, Q., Han, Z. and Sun, K. (2015), Deep feature representation via multiple stack auto-encoders, *in* 'Pacific Rim Conference on Multimedia', Springer, pp. 275–284.

Yarowsky, D. (1995), Unsupervised word sense disambiguation rivaling supervised methods, *in* '33rd annual meeting of the association for computational linguistics', pp. 189–196.

Young, T., Hazarika, D., Poria, S. and Cambria, E. (2018), 'Recent trends in deep learning based natural language processing', *IEEE Computational intelligence magazine* **13**(3), pp. 55–75.

Yuan, D., Richardson, J., Doherty, R., Evans, C. and Altendorf, E. (2016), 'Semi-supervised word sense disambiguation with neural models'.

Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O. (2021), 'Understanding deep learning (still) requires rethinking generalization', *Communications of the ACM* **64**(3), pp. 107–115.

Zhang, X., Zhao, J. and LeCun, Y. (2015), 'Character-level convolutional networks for text classification', *Advances in neural information processing systems* **28**, pp. 649–657.

Zhang, X. and Zheng, X. (2016), Comparison of text sentiment analysis based on machine learning, *in* '2016 15th international symposium on parallel and distributed computing (ISPDC)', IEEE, pp. 230–233.

Zheng, A., Shelby, N. and Volckhausen, E. (2019), 'Evaluating machine learning models', *Machine Learning in the AWS Cloud* .

Zhou, X. and Han, H. (2005), Survey of word sense disambiguation approaches., *in* 'FLAIRS conference', pp. 307–313.

Zvonarev, A. and Bilyi, A. (2019), A comparison of machine learning methods of sentiment analysis based on russian language twitter data, *in* 'Proceedings of the 11th Majorov International Conference on Software Engineering and Computer Systems (MICSECS)', pp. 1–7.