Transfer Learning of Sentiment Analysis Between Highly Dissimilar Domains

By

Guy Coop

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Department of Electronic, Electrical and Systems Engineering
School of Engineering
College of Engineering and Physical Sciences
University of Birmingham
May 2023

ABSTRACT

The aim of this thesis is to improve upon the current state of transfer learning technology in text based problems, with a specific focus on sentiment analysis. This improvement is in the form of increased distance between training and target domains. It proposes a method for defining inter-domain distance. Then examines the shortcomings of classical transfer learning methods, and proposes two novel approaches. The first 'TransferGAN' extends on the use of Adversarial Learning Techniques for transfer learning, and the second demonstrates how Grammatical Evolution can be used to optimize existing sentiment analysis techniques to better suit transfer learning between dissimilar domains. Both of these methods demonstrate improvement over the comparison systems for transfer learning sentiment analysis with a high inter-domain distance. Finally these systems are demonstrated to be effective against the Motivational Interview dataset (A dataset of clinical conversations between diabetes patients and clinicians), which was the primary motivator for this work. This work could be further extended by field-testing of the systems against real world usage of the Motivational Interviews, and by working with clinicians to refine the system for automatic assessment of the Interviews.

ACKNOWLEDGMENTS

I would first like to thank my incredible supervisors: Dr Peter Jančovič, and Professor Martin Russell for their incredible support and guidance, not just in my PhD, but a significant portion of my academic Career. I would definitely not be where I am today without them.

I would also like to thank my brilliant wife Emily for her encouragement to complete this thesis, even when I didn't want it, which was the times that I needed it the most. My son Theo for his enthusiastic support, even if he hasn't yet learned the words to give it. My family for their belief in me, the kind that only a family can have because it often defies reason. And my dog Lila for the companionship she provided on long days of juggling work and thesis writing.

Contents

List of Figures

List of Tables

Acronyms

Δ (in the context of MFCC Vectors) The change in MFCC between timesteps

$\Delta\Delta$ (in the context of MFCC Vectors) The change in MFCC-Δ features between timesteps

ASBA Aspect-Based Sentiment Analysis

BERT Bidirectional Encoder Representations from Transformers

BNF Backus-Naur Form

CFG Context-free Grammar

CNN Convolutional Neural Network

D2V Doc-To-Vec (Document Vectorisation System)

DCGAN Deep Convolutional Generative Adversarial Network

DNN Deep Neural Network

EA Evolutionary Algorithm

FCNN Fully Connected Neural Network

FFT Fast Fourier Transform

GAN Generative Adversarial Network

GE Grammatical Evolution

GP Genetic Programming

GPT Generative Pre-trained Transformer

GRU Gated Recurrent Unit (an RNN style unit type)

HAGAN Heirarchical Attention Generative Adversarial Network

HAN Heirarchical Attentive Network

HMM Hidden Markov Model

JS Jensen-Shannon (Divergence)

KL Kullback-Leibler (Divergence)

LLM Large Language Model

LSTM Long Short-Term Memory (an RNN style unit type)

MDSD2 Multi Domain Sentiment Dataset 2

MEMM Maximum Entropy Markov Models

MFCC Mel Frequency Cepstral Coefficient

MI Motivational Interview

ML Machine Learning

NHS National Health Service

NLP Natural Language Processing

NLTK Natural Language Toolkit

NN Neural Network

POS Part-of-speech

RAVDESS Ryerson Audio-visual Database of Emotional Speech and Song

RNN Recurrent Neural Network

SA Simulated Annealing

SE-2017-T4 SemEval-2017 Task 4 (A dataset from the workshop on Semantic Evaluation focused on twitter extracts)

SemEval An alternate shorthand for SE-2017-T4

SMOTE Synthetic Minority Over-sampling Technique

TF TensorFlow

TF-IDF Term Frequency - Inverse Document Frequency

TL Transfer Learning

W2V Word-To-Vec (Word Vectorisation System)

WGAN Wasserstein Generative Adversarial Network

WV Word Vector

Chapter One

Introduction

The NHS uses "Motivational Interviews" (MIs) as a core way of guiding patients suffering from a wide range of conditions to better manage those conditions, and to help triage patients that may be in need of greater intervention (Lundahl et al., 2013) (Soderlund, 2018). This work will focus only on MIs of Diabetes patients, which alone, through medication and complications of the disease cost the NHS an estimated £9bn per year. The primary goal of this thesis is to work toward the automatic evaluation of these interviews in order to improve patient outcome while helping to reduce the human resources required for evaluation.

These interviews present a number of challenges for automatic assessment: They are noisy and conversational in tone, they are 2 party conversations where the "assessment" of the interview is a moderately subjective act, and the data available for this project lacks sufficient labelling to train any form of standard classification system. This Thesis will primarily address the latter of these concerns by developing and evaluating a number of novel transfer-learning techniques, with the aim of using datasets from a variety of other sources to produce meaningful results on the MI dataset. And advance the field of transfer learning, with a focus on the analysis of sentiment and emotion.

The Thesis contains a review of the essential literature surrounding this project, as well as brief explanations of some core concepts. There is then a review of the work and experiments undertaken, and Finally an in depth discussion of the implications of that work, as well as a discussion of the possible avenues for future extensions on this work to help further the goal described above.

1.1 Research Questions

This thesis attempts to address the following research questions:

1. *"To what extent can we classify emotion in conversational speech? And can this be used as an analogy for the classification of sentiment and conversational engagement?"*
2. *"Can we define a measure similarity between different domains of text based data"*
3. *"In the scenario of having little or no labelled data for a sentiment analysis problem, what performance can we achieve using current transfer learning methodologies on similar data?"*
4. *"Can the performance of the above scenario be improved using novel transfer-learning methodologies?"*
5. *"To what extent does the performance degrade as we decrease similarity between the source and target domain datasets? and can this degradation be mitigated?"*
6. *"Can these novel methodologies be applied to the Motivational Interviews in order to gain new understanding of the data and apply meaningful labels?"*

1.2 Chapter Outline

Chapter 2 Chapter 2 is a literature review that aims to introduce many of the technologies and methods built-upon by this thesis, highlights other research that is working toward solving the problems discussed in this thesis, and outlines any other important literature relevant to the wider field.

Chapter 3 Chapter 3 introduces the datasets that will be discussed and utilised for experimentation within this thesis. It includes summary statistics, preprocessing considerations, and any other information relevant to fully understand the nature of the dataset and the reason for it's inclusion in this thesis.

Chapter 4 Chapter 4 introduces the parameter-optimisation techniques that will be used on systems throughout this thesis. Additionally it demonstrates the effectiveness of emotion classification human speech in order to allow recommendation of further research into multi-modal approaches to sentiment classification of recordings and transcriptions.

Chapter 5 Chapter 5 demonstrates an LSTM based sentiment classifier that will be used as the baseline result for a number of experiments in later chapters. Additionally it compares Jensen-Shannon Divergence with a novel method for measuring domain similarity proposed by this thesis. This Domain similarity method is then utilised in later chapters to quantify "similar", "dissimilar", and "highly dissimilar" domains.

Chapter 6 Chapter 6 thoroughly investigates the HAGAN system, proposes and evaluates a number of suggested improvements to the system, demonstrates it's effectiveness in a number of scenarios. And investigates and explains some of the potential short-comings of the system.

Chapter 7 Chapter 7 proposes an alternate novel method for transfer learning. Utilising Genetic Programming to learn optimised processes for the goal of transfer learning. This meta-optimisation is shown to be effective using a simple demonstration grammar. This chapter does not present a complete analysis of this style of system, but proposes a method that could be explored as an avenue for further research.

Chapter 8 Chapter 8 Completes the thesis by demonstrating the systems developed in Chapter 5, 6, and 7 against the motivational interview data that is the primary motivator for this thesis. These systems are demonstrated to be at least partially effective for the goal of transfer learning onto a previously unseen dataset.

Chapter 9 Chapter 9 is the conclusion and discussion, in which the findings of the thesis are discussed and related back to the research questions in order to demonstrate the contribution of the work.

1.3 Contributions of the Thesis

This Thesis provides the following contributions:

- Exploration of the possibility and effectiveness of Sentiment Analysis techniques for Motivational Interview data.
- Evidence for recommendation for multi-modal Sentiment Analysis of Motivation Interviews.
- A significant analysis of the HAGAN system, multiple suggestions for improvements to the methodology, and demonstration and analysis of some of the potential failure modes of the system.
- Proposal of a novel method for meta-optimisation using Genetic Programming to discover rule-sets to segment data into groups for separate classification.
- Demonstration of the effectiveness of transfer learning between highly dissimilar domains, showing how highly-available product review data can be used to learn complex sentiment features of conversational transcriptions.

1.4 Details of Experimental Setup

Unless explicitly stated, all experiments in this Thesis were performed using the following hardware / software versions

Hardware It can be noted by the hardware list, that an additional non-functional goal of this Thesis was to develop techniques that can be applied using readily-available consumer hardware.

- Processor: i7 6700k
- Graphics Card: Nvidia GTX 1070
- RAM: 24GB DDR4 3000MHz

Core Development Environment

- Windows 10
- Anaconda developer environment
- Python 3.9
- cudatoolkit 11.2.2
- CUDNN 8.1.0
- tensorflow 2.10.1
- keras 2.10.0

Where other packages or developer tools are significant to experimental setup, those details will be listed separately.

Chapter Two

Literature Review

The following topics represent the core elements required to complete the overall goal of this project: To perform automatic sentiment analysis on interviews that are very sparsely labelled for the features of interest (sentiment, emotion, and engagement)

2.1 Word and Document Representation Methods

When performing both para-linguistic (analysis of the non-language component of human speech), and linguistic (analysis of the language component, usually from a transcription or speech recognition system) analysis; it is necessary to convert the raw waveform or raw text information into a vector format so that it can be better manipulated by a machine learning algorithm or other computer processing system. Some methods of performing this transformation (vectorisation) are detailed below.

There are a number of methods of representing audio human speech in vector form, while these are not critical to the content of this report (as it is primarily textual), they are still worth noting as they are relevant to prior work, and they will likely regain relevance in future work as it tends back towards audio focused and hybrid speech-and-transcript systems. Mel Frequency Cepstral Coefficient (MFCC) (Muda, Begam, and Elamvazuthi, 2010) vectors are a common method of representing speech signals as windowed vectors, however while this method is an excellent baseline system to compare against, they will not be discussed in detail as they are considered to be surpassed by modern systems that exploit more abundant computing power. I-vectors (Dehak et al., 2010) are a method of mapping a distribution from a high dimensional vector space (such as MFCCs or other frame vectors) into a compressed lower dimensional space that is referred to as the "total variability space": containing both the speaker variability information and other channel variability information. This method was originally demonstrated to be effective in speaker verification problems, however it has been demonstrated as a powerful addition to other systems to improve speaker independence and robustness to other variability. While vectorisation methods like those described above have been the defacto standard in speech and speaker classification tasks for many years, recent advances in computing power have made it possible to manipulate and perform classification on the raw waveform directly (Trigeorgis et al., 2016). This method seems to be approaching the new standard as computing power and machine learning techniques continue to improve, and it can be clearly understood that any vector representation will always be omitting some parts of the raw signal, and so if it is possible to manipulate the signal directly, this is likely to be the better performing method.

The "Word2Vec" system (Mikolov et al., 2013) was an extremely important innovation in the field of natural language processing. It is a word embedding system that is considered to perform a good approximation of the semantic meaning of the words during the embedding process. The core of the system is a neural network that is trained to predict a distribution of the words that are likely to be in the immediate context of the "target" word. The input and output of the system are both a wide vector that is a one-hot system for the entire dictionary in the training set. The "word vector" is then extracted from a bottleneck layer in the system after it has been trained on a very large corpus of text (hundreds of millions of words in document form). Because the system is tasked with predicting the words likely to surround the "target" word. It can be thought to be estimating the meaning of the word. This can be verified by performing simple vector maths in the embedding space and observing the nearest neighbor to the output. For instance $V(\text{king}) - V(\text{man}) + V(\text{woman}) \approx V(\text{queen})$. This and many other examples verify that the system appears to be generating an embedding space that is arranged in a semantically sensible way. One minor disadvantage of this system however is that it is not able to easily differentiate between words of different meaning with the same spelling such as "rose" (a flower) and "rose" (past tense of "rise") and instead blends the meaning together and considers them as one word.

The "Doc2Vec" system (Le and Mikolov, 2014) is an extension to the "Word2Vec" system described above. The Word2Vec system functions by estimating the semantic meaning of each word by estimating the distribution of it's surrounding words. The Doc2Vec system includes an additional input unit that contains the paragraph (or document) ID, which is then passed forward to its own hidden layer that will contain the document vector. When generating a document vector, the word-vector system is trained on the document in question, now the hidden layer where the word-vector is extracted from is now an approximation of the context words surrounding the target-word, with a bias towards the specific context found in the target-document, Finally the document vector can be extracted from the corresponding hidden layer by inputting the appropriate document ID. Similar to the Word2Vec system above, this document vector can be considered to be a semantically valid representation of the document as it contains all the necessary information to predict a given word **specifically localised to the target-document** when provided the words immediately surrounding it.

2.1.1 Part-of-Speech (POS) tagging

Part-of-speech (POS) tagging is a task in natural language processing that involves the assignment of a grammatical category, such as noun, verb, etc. to words in a provided document. POS tagging has a number of useful applications in areas such as classification, and machine translation.

The earliest approaches to POS tagging was to use rule-based systems (Brill, 1992) to define the category of a word. However these systems were very limited in terms of accuracy, and it required extensive manual effort to create the rules, and therefore limited the scalability. This was improved upon by the application of statistical models to this problem.

The most popular approach was to use Hidden Markov Models (HMMs) (Kupiec, 1992) which form a probability distribution of likely POS tags based on the tag of the previous word. It is also common to use Maximum Entropy Markov Models (MEMM) for this problem (Ratnaparkhi, 1996). Both of these models have been show to achieve highly accurate POS tags in real-world applications.

The implementation of these models is outside of the scope of this thesis. When POS tags are used, they are built modeled using NLTK (Natural Language Toolkit) (Bird, Klein, and Loper, 2009) which is a Python library that contains many useful tools and models for interacting with Natural Language data; including implementations of POS tagging algorithms, such as a HMM and the Brill Tagger (which is a rule-based approach that uses transformational rules to improve tags). NLTK provides pre-trained models for POS tagging in multiple languages, which greatly simplifies utilisation in further research. (Straka and Straková, 2017)

2.1.2 Domain Similarity Measurement in Documents

In 5.3 we present a novel method to infer a measure of similarity between large domains of documents. It is necessary to provide a classical alternative implementation to compare against in order to demonstrate the additional benefits of the proposed solution. One such methodology uses a distribution of Term Frequency Inverse Document Frequency (TF-IDF) (Aizawa, 2003) vectors for a domain and compares them to another such vector using Jensen-Shannon (JS) Divergence (Menéndez et al., 1997). The calculation for the TF-IDF of a single document is given below:

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$\mathrm{IDF}(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

$$\text{TF-IDF}(t,d,D) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t,D)$$

$$\mathbf{v}_d = [\text{TF-IDF}(t_1,d,D), \text{TF-IDF}(t_2,d,D), \ldots, \text{TF-IDF}(t_n,d,D)]$$

These vectors are then summed and normalised across all the documents in a domain to produce domain vectors P and Q that are passed into the Jensen-Shannon Divergence calculation given by:

$$\mathrm{JS}(P \parallel Q) = \frac{1}{2}\left(D_{\mathrm{KL}}(P \parallel M) + D_{\mathrm{KL}}(Q \parallel M)\right)$$

$$\text{where } M = \frac{1}{2}(P + Q)$$

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

2.2 Sentiment Analysis and Domain Transfer Learning

Transfer learning (TL) in the context of machine learning problems is simply the process by which one can take knowledge gained in one field or domain and apply it to another. For instance a system trained to recognise cars may be tweaked using minimal additional data to recognise vans or other similar vehicles instead. This is useful in machine learning problems, as it allows us to gain knowledge from an area that is rich in available training data, and transfer it to an area with minimal training data. The following is a sample of literature demonstrating a number for different methods for sentiment analysis and cross-domain sentiment transfer learning.

"A survey of sentiment analysis based on transfer learning" (Liu et al., 2019) serves as an excellent secondary resource for understanding some of the basic types of transfer learning that can be applied to this problem, as well as giving insight into some modern techniques used in standard sentiment analysis, such as a CNN architecture for NLP problems, which will be described further in section 6.7. This thesis gives an overview of the following transfer learning methods:

- **Parameter-Transfer Methods** which leverage parameter sharing model between the source and target domains

- **Instance-Transfer Methods** which exploit similarities between elements of the Source Dataset and elements of the Target dataset to augment the Target training set with new items.

- **Feature-Representation-Transfer methods** which utilise a shared feature space between the source and target data in order to learn shared features between the two domains. This is the method that will be the primary focus of this thesis.

Yuan et al. present a strong example of this kind of Feature-Representation-Transfer in their paper "Domain attention model for multi-domain sentiment classification" (Yuan et al., 2018). Their method utilises shared features between multiple domains to learn a joint representation that can then be fine tuned by an attention layer in order to select specific domain relevant features. This method proves to be very effective at creating a robust a sentiment analysis model that is capable of training on very low-resource datasets, provided there are additional training domains available to support the model.

A contrasting method is presented in "An evolutionary strategy for concept based multi-domain sentiment analysis" (Dragoni, 2019) This method aims to utilize the language models of SenticNet and Wordnet in order to build nodes of "semantic concepts". Evolutionary algorithms are then employed to estimate the domain-specific polarity of these specific concepts to ensure that the solution space is well explored. At test time a document is parsed into a sequence of concept nodes, which are then checked to determine their domain specific polarity, these polarities are then combined to form the final classification for the document. While not strictly relevant to the content of this thesis. This method demonstrates a highly novel approach to domain-transfer sentiment analysis that seems likely to be equally functional on both long and short-form datasets (i.e documents vs tweets)

2.3 The BERT model

The BERT (Bidirectional Encoder Representation from Transformers) System (Devlin et al., 2019) is a more recent innovation in word embedding technology, It aims to alleviate some of the shortcomings of the "Word2Vec" system, as well as providing even more robust and well trained word embeddings. The BERT system, unlike "Word2Vec" is a dynamic system based on neural networks, meaning that after training time, the network must still be evaluated each time a new embedding needs to be generated. This makes it slower and much less lightweight that the Word2Vec system which can be converted into a dictionary format after training and then words can be looked up instantaneously. However the BERT system performs context-sensitive embeddings (hence the need for network evaluation at run-time). This means that at embedding time, and entire sentence or document should be passed through the system at once, and each word embedding takes into account the actual real-world context that is surrounding it (rather than the general-case context as in W2V). This method creates extremely powerful word embeddings that have been proven to have improved performance over W2V in a variety of applications. Additionally, because the BERT system requires the document to be passed concurrently. It has an additional output called the `CLF` ("classification") output, which can be thought of as a document vector for the input document. This `CLF` output is designed so that in real world uses, a fully-connected classification layer can be added to the end of the BERT system, and then the system can be partially retrained on relevant classifiable data and function as a fully fledged classification system. This system alone has been shown to out-perform many state of the art classification systems for a wide range of problems including sentiment analysis (Munikar, Shakya, and Shrestha, 2019). This is partially due to the quality of the pre-training data and the ability of the system to represent the semantic content of a sentence or document.

2.3.1 BERT for sentiment classification in out-of-domain scenarios.

Traditionally, sentiment analysis focuses on determining the overall sentiment of a text—whether it is positive, negative, or neutral. However, this approach fails to capture more nuanced information, such as identifying which specific entities or topics within the text the sentiment is directed toward. This limitation gave rise to aspect-based sentiment analysis (ABSA), a more complex task that aims to identify both the sentiment and the aspects (topics or entities) within a text.

"Aspect-based sentiment analysis using BERT" (Hoang, Bihorac, and Rouces, 2019) proposes leveraging pre-trained language models, specifically BERT (Bidirectional Encoder Representations from Transformers), to enhance the performance of ABSA, particularly in out-of-domain scenarios where the model encounters data that differs from its training set. By fine-tuning BERT with additional generated text, the authors demonstrate a significant improvement in ABSA performance, surpassing previous state-of-the-art results on benchmark datasets like SemEval-2015 and SemEval-2016. The proposed model uses BERT's sentence pair classification technique to capture semantic similarities between text and aspects, allowing for more accurate sentiment and aspect classification. This approach marks an important advancement in applying transfer learning for complex tasks like ABSA, particularly in domains where labeled data is scarce.

2.4 State of the Art in Sentiment Transfer Learning

"State of the Art: A Review of Sentiment Analysis Based on Sequential Transfer Learning" (Chan et al., 2023) introduces the current state-of-the-art approaches to sentiment analysis. This paper expands on a three-phase approach and elaborates on the history of this methodology and how it has improved over time. The first phase focuses on the development of pre-trained word embeddings, which marked a significant shift in how words are represented in computational models. Early methods like one-hot encoding were insufficient for capturing the semantic relationships between words, but the introduction of distributed word representations, such as word2vec and GloVe, enabled models to encode nuanced meaning and contextual information by relying on large-scale corpora. These pre-trained embeddings are essential because they allow models to generalize across a wide range of tasks by leveraging vast amounts of unlabelled data.

The second phase emphasizes the emergence of contextual pre-trained encoders, such as BERT, which further enhanced the ability to capture the dynamic nature of language. Unlike static word embeddings, contextual models consider the meaning of words in relation to their surrounding context, allowing them to handle polysemy and other complexities of natural language. Models like BERT are pre-trained on enormous datasets, often comprising billions of words, and are able to learn deep contextual information through bidirectional language modeling. This reliance on large-scale pre-training has become a cornerstone of modern natural language processing (NLP) tasks, as it allows models to encode a richer understanding of language that can be fine-tuned for more specific downstream tasks, such as sentiment analysis, with comparatively smaller datasets.

The third phase, knowledge adaptation through fine-tuning, involves adapting these pre-trained models to specific tasks by transferring the learned representations. Fine-tuning techniques allow models to adjust their learned parameters to the specific characteristics of a target task, improving performance while requiring far less data than training from scratch. The reliance on large, pre-trained models becomes evident in this phase, as their success in fine-tuning depends on the quality and scale of the pre-training phase. Techniques such as discriminative fine-tuning and layer-wise adaptation ensure that these pre-trained models can be effectively tailored to handle tasks like sentiment classification, even in specialized domains with limited labeled data.

Overall, the sequential transfer learning approach described in the paper highlights the critical role of large-scale pre-trained models in modern NLP. These models enable efficient learning and adaptation across a wide variety of tasks by leveraging the vast amounts of data used in pre-training, which forms the foundation for more sophisticated fine-tuning approaches in specific applications.

2.5 Adversarial Learning

Generally speaking, Adversarial Learning is a class of Machine Learning problem in which the system comprises 2 or more agents working independently toward their own goal.

For the purpose of this thesis we will be focusing on a specific subclass of Adversarial Learning: Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). And it is important to fully understand their typical functionality in order to understand the main work presented in this thesis. A typical use-case of a GAN would be to generate artificial but realistic images ("fake images"), meaning that given a training set containing a large number of images, the GAN would be able to generate new images that are thematically similar to the images provided, and often appear to be real photographs (as opposed to being computer generated), but are entirely new entities.

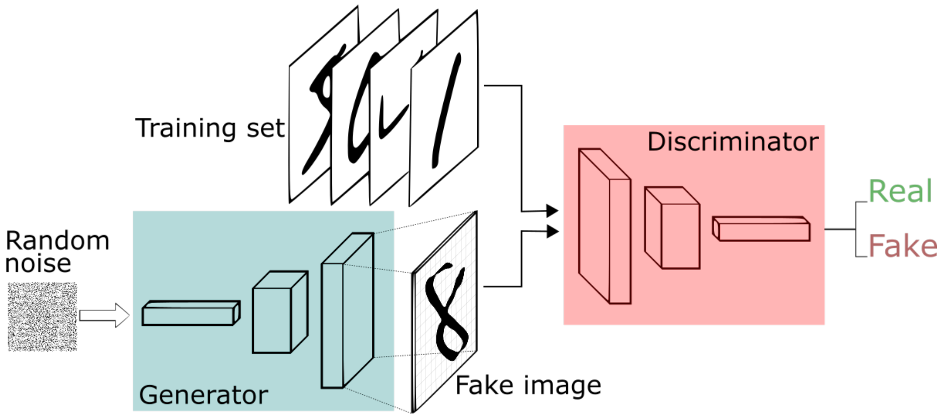


Figure 2.1: An Example GAN Operation (on an image processing task)

Formally speaking, a GAN is made up of two networks: The Generator, and the Discriminator. The Generator is a network responsible for converting some arbitrary lower dimensional space (called the "latent space") into an embedding of some specified dimension (i.e. 256x256x3 for a small image). During training the Discriminator network is then asked to compare images from the training set with images that have been produced by the generator, in order to determine which is "real" (meaning which image is from the training set and which has been generated). By this method, the Discriminator can be thought of as learning to approximate a complex loss function that describes "what it means to be an image that is part of the training set". The Generator is able to learn how to map its random noise input to a point (i.e. an image) that is within the subspace represented by the training set.

The common analogy used to understand the training process of a GAN is to imagine a currency *Forger* (playing the role of the Generator) and a *Detective* (playing the role of the Discriminator). At the start of training, neither the Forger nor the Detective have ever seen a dollar bill. Each iteration of their "game", the Forger will create a new forged bill to the best of his ability, then the detective will be presented with the forgery and a real bill and asked to try and determine which is which. The Detective is then told which is the real bill, and he learns something about how to spot a real bill from a forgery; at the same time, the Forger is told how difficult it was for the Detective to tell the difference between the bills and he gains some information about what he might do better next time.

This "game" then continues for many iterations, with the Forger constantly improving the quality of the fake bills, and the Detective getting better at separating real and forged bills. After this training has been running for some time, to an outside observer the quality of the forged bills has now improved to be almost indistinguishable from the real thing. In this way, a GAN can be used to generate artificial examples of any media, that are representative of the training set that it is provided with.

"Progressive Growing of GANs for Improved Quality, Stability. and Variation" (Karras et al., 2018) demonstrates, very effectively, the capability of GAN based systems. As the title suggests, this system generates realistic images of human faces based on the `CelebA` dataset by "progressively growing" the GAN system. Beginning with images of size `4x4x3`, training the GAN on these for a sufficient amount of time, before adding the next layer of `8x8x3`. The process is continued up to a size of `1024x1024x3`. Growing the GAN in this was allows it to learn critical structure of the target images in its lower dimensional space before it is scaled up. If a system was used that started at a target size of `1024x1024x3` the images would likely become very unstable and not optimize toward believable images, no matter how long the system was trained for.

It is also possible to use GANs or Adversarial Learning techniques in order to generate any other form of media or data, provided it can be vectorised in some way. Additionally it must be possible to structure the system in such a way as to be able to adequately generate media of that format. For instance, using the WaveGAN (Donahue, McAuley, and Puckette, 2018), which is a modified version of the "Deep Convolutional Generative Adversarial Networks" (DCGAN) model (Radford, Metz, and Chintala, 2015) it has been demonstrated that it is possible to generate convincing samples of raw waveform audio in a number of domains, including music, and bird vocalisations.

Arjovsky et al. present an improved training methods for GANs which they refer to as Wasserstein GANS (WGANS) (Arjovsky, Chintala, and Bottou, 2017). This method suggests that using a modified version of the Earth-Mover Distance is theoretically and experimentally more effective than the classic GAN algorithm (which utilises Jensen-Shannon divergence). A full discussion of the theoretical proof of this is beyond the scope of this report to discuss, however the practical relevance to this project is an alternative method of constructing the GAN loss function that may lead to improved performance Gulrajani et al., 2017.

2.6 Generative Adversarial Networks (GANs) for Transfer Learning

The Hierarchical Attention Generative Adversarial Network (HAGAN) (Zhang, Miao, and Jiaqi Wang, 2019) system is one of the primary sources for this work. In this paper the authors demonstrate a fairly novel use of a GAN system to perform transfer learning between domains of the MDSD2 dataset (section: 3.4). This system demonstrates use of a GAN as a **transformation** network from one meaningful vector to another, in contrast to the traditional GAN methodologies discussed above, where a random input is given and converted into a meaningful vector (in order to build a mapping from the latent space to the target space). The goal of this system is to perform transfer learning from a *Source* sentiment-labelled dataset onto a *Target* sentiment-unlabelled dataset. This is achieved by utilising a GAN that performs simultaneous optimization of two problems:

- Map input text into a document-embedding space that is optimal for classifying the sentiment of the input
- Map input text into a document-embedding space that is **inseparable with respect to the domain** (meaning that data from the *Source* domain cannot be separated from data from the *Target* domain.)

In a typical GAN architecture, the system would not experience any pressure to make sensible mappings from it's input layer to it's embedding layer, provided that the discriminator was unable to separate domain A from domain B. However the addition of the sentiment-classification task on the same network, forces a collaborative task between the **Generator** and the **Discriminator**. This collaboration is used to anchor the Generator into performing intelligent mappings into the embedding space. The hypothesis of this system is that by mapping the *Target* (out-of-domain) data to be inseparable from the *Source* (in-domain) data, the system will also learn sentiment information about the un-labeled *Target* data by exploiting the overlap between the two domains.

This HAGAN system makes use of a Hierarchical Attentive Network (HAN) (Yang et al., 2016) for the generator network. When applied to text these networks typically contain two layers: A first layer to map from individual words to sentence-representations; followed by a second layer that maps from these sentence-representations to a whole document-representation. Both of these layers also feature an self-attention mechanism so the system can apply greater importance to particularly relevant words or sentences. The implementation details of a HAN are discussed in further detail in section 6.7. The discriminator layer of the HAGAN network is then implemented using a simple 2 layer fully connected network, in order to map from the generated document embeddings to the output probabilities for both the **sentiment-classification** decision, and the **domain-classification** decision.

GANs have also been evaluated on the similar problem of low-resource (low availability of labelled data) sentiment analysis (Gupta, 2019). This author converts textual documents into document-vectors using the Doc2Vec method mentioned above. Then utilises a GAN to perform data augmentation to generate additional training examples in order to help refine a standard sentiment analysis system. This paper demonstrates some of the problems of this method and notes that the data generated by the GAN does not appear to completely fill the distribution of data represented in the training set. Using this method, the author notes minimal improvement in the training quality when using the augmented dataset, this could be as a result of the difficulty training the GAN in this situation, and it is possible that making a more sophisticated GAN system that generated raw text directly, would make the system more inspectable and therefore more tunable to generate high-quality augmented data would show an improvement in training quality of sentiment analysis tasks. While this may seem like a fallacy as it implies the ability to generate new data from a training set with no additional input; it would be possible to employ language models and large amounts of unlabeled data in order to implicitly learn sentimental language without the need for labelled data.

2.7 Genetic Programming and Grammatical Evolution

Genetic Programming (GP) (Koza et al., 1994; Espejo, Ventura, and Herrera, 2009) is a field of evolutionary computation focused on the generation of executable computer programs that are optimized using evolutionary algorithms.

Grammatical Evolution (GE) (Ryan, Collins, and O'Neill, 1998) is a powerful form of GP that uses a context-free grammar to represent the possible solution-space. This imposes an important set of constraints on the system that enables it to develop potential solutions to an objective while always producing syntactically and semantically valid solutions.

While Grammatical Evolution is typically optimized through the use of Evolutionary Algorithms, it fundamentally involves the definition of a search space using a grammar, and then an exploration through the search space to find a close to global optimum result for the objective. It is possible to use non-EA algorithms for searching the solution space. An alternative algorithm: Simulated Annealing, is discussed later in this section.

2.7.1 Context Free Grammars

Context-free grammars (CFGs) (Seki et al., 1991) are formal languages used not just in GE applications, but in a range of problems in computer-science, as well as linguistics and natural language processing. CFGs rely on essentially building a large graph of non-terminal (meaning they can be further expanded) and terminal symbols. There is then a start symbol, and a set of production rules that define the relationships between symbols. CFGs have some limitations: They cannot handle some aspects of natural language syntax, such as cross-serial dependencies and long-distance dependencies. And more sophisticated systems have been developed for use in Natural Language Applications. However CFGs are ideal for the definition of computational languages and are still commonly used for this purpose.

For this thesis we will be focusing primarily on Backus-Naur Form CFGs (JW, 1959) which is the most common CFG used for defining computer programming languages.

See the following example of a very simple grammar used to produce mathematical expressions:

```
# NON TERMINALS
<start> ::= <expression>
<expression> ::= <expression> <symbol> <expression> | <number>
<number> ::= <digit> | <number><digit>

# TERMINALS
<symbol> ::= + | - | * | /
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Using just these symbols and production rules. We can produce any simple mathematical expression that uses the symbols in the 'symbol' list. For example:

```
<start>
<expression>
<expression> <symbol> <expression>
<expression> <symbol> <expression> <symbol> <expression>
<number> <symbol> <number> <symbol> <number>
<digit> <symbol> <number><digit> <symbol> <number><number><digit>
6 + <digit>5 - <digit><digit>9
6 + 25 - 129

# we now have a mathematical "program" that was guaranteed to be executable
# we can evaluate this "program"
= -98
```

From this example we can see the strengths of using a BNF grammar to define the possible rules of a language. And it is clear how programs are guaranteed to be syntactically valid. However a grammar needs to be very robustly defined in order to guarantee the validity of the programs. In the example above, while it is a given that a program will not return nonsense. It is still possible to produce expressions that are mathematically impossible to resolve, such as 0/0 (or any division by 0).

2.7.2 Simulated Annealing

Simulated Annealing is a powerful optimization technique that is applied to a wide variety of problems in various fields. It is a stochastic optimization method inspired by the physical process of annealing, in which a material is heated to a high temperature and then slowly cooled over time, reducing its energy and improving the stability of its internal structure. Due to it's stochastic nature and the ability to define arbitrary cost functions, it is a powerful technique for solving complex optimization problems where traditional methods are not as effective.

One of the key advantages of simulated annealing is its ability to handle non-convex and multi-modal objective functions. Traditional optimization methods, such as gradient descent, may get stuck in local optima and fail to find the global optimum. Simulated annealing can explore different regions of the solution space and converge to a near-optimal solution, even in the presence of multiple local optima.

2.7.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of search and optimisation techniques inspired by the idea of natural selection. EAs are used to find a close to global optimum of a given objective function starting with an initial population of candidates spread throughout the solution space, and then repeated iterations of selection, reproduction, and mutation.

For the purpose of this thesis, we will focus on the difference between EAs and Simulated Annealing, (described above): EAs use a population based approach that allows them to initially cover a significant portion of the solution space, and therefore have higher likelihood of finding a global optimum. Each generation of the algorithm requires the evaluation of an entire population of (often several hundred) individuals, which can make it a more computationally expensive process that simpler hill-climbing algorithms, and hybrid approaches like SA. Additionally, EAs use a crossover "reproduction" based method to generate new candidate solutions, whereas SA uses mutation of the current candidate.

2.7.4 Transfer Learning In Genetic Programming

In my literature review, I examined existing research on the use of transfer learning within genetic programming (Muñoz, Trujillo, and Silva, 2020)(Dinh, Chu, and Nguyen, 2015). These papers provide valuable insights into how transfer learning can be applied to enhance genetic programming models. However, the focus of my thesis is distinct in that it investigates how genetic programming can be utilized as a tool for identifying and optimizing transfer learning methodologies for other systems, rather than the direct application of transfer learning to genetic programming itself. While the reviewed papers contribute to the broader understanding of transfer learning within the field, they are not directly relevant to the research conducted within this thesis. Upon extensive review of the literature, it appears that this specific approach has not been previously explored, which underscores the novelty of this concept, and strengthens the case for further exploration of the idea.

2.8 Large Language Models

Since 2022, Large Language Models such as the GPT3 system (Brown et al., 2020; Floridi and Chiriatti, 2020) Have entered the mainstream and revolutionised public understanding of what is possible with Natural Language Processing. While they are not strictly related to the work of this thesis (where much of the work presented in this thesis was conducted prior to GPT models becoming so relevant in the sphere of NLP, both academic and practical). It is important to acknowledge this fundamental shift in the landscape and perform an examination of LLMs and their applicability to Sentiment Analysis and Transfer Learning.

2.8.1 LLMs for Sentiment Analysis

Large language models (LLMs), such as GPT and BERT leverage vast amounts of data to predict and generate text. These models are pre-trained on extensive corpora, learning to capture the intricacies of language, context, and meaning. While LLMs are highly effective for general language tasks, they are not specifically designed or optimized for targeted tasks like sentiment analysis. Sentiment analysis requires identifying the emotional tone or opinion expressed in text, a task that benefits from focused classification. LLMs, however, are built to generate text based on probabilities rather than making categorical decisions. As a result, they often require fine-tuning or additional layers of training to adapt to tasks like sentiment analysis, where precision and interpretability around sentiment categories are crucial.

In "Sentiment analysis through llm negotiations" (Sun et al., 2023), Sun et al. propose a novel framework for leveraging LLMs in a cooperative manner to perform sentiment analysis tasks. This is a demonstration of how pre-trained LLMs can be leveraged trained on peripheral tasks such as learning to provide answers and reasoning, and then transitioned into classification tasks such as sentiment.

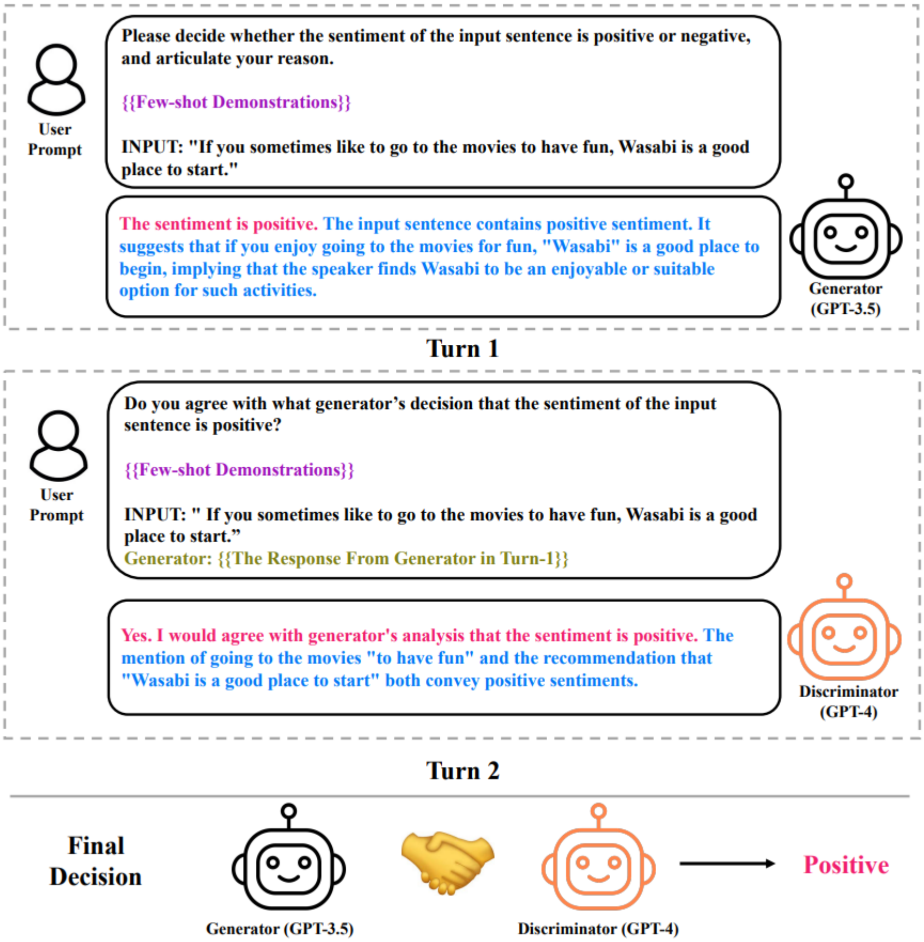


Figure 2.2: LLM Negotiations

This technique was demonstrated to be effective on a wide range of sentiment analysis benchmarks. The main drawback of this method is its computational inefficiency, as it relies on two extremely resource intensive LLM models in order to perform sentiment analysis tasks that could be accomplished by a much more compact system. While not explicitly demonstrated in this paper, it is possible that due to the massive data-volume pre-training of LLM systems: this system would be extremely robust to out-of-domain classification task such as on the MI data discussed in this thesis.

Chapter Three

Datasets and Corpora

This chapter introduces the datasets used throughout the thesis, providing an overview of their key characteristics. The datasets will be summarized, highlighting their structure and content, followed by a detailed explanation of class distributions and other relevant statistical measures. Additionally, the chapter will discuss the pre-processing steps and considerations that were applied to the databases. Finally, a justification will be presented for each dataset, explaining its relevance and why it was included in the research.

3.1 Ryerson Audio-visual Database of Emotional Speech and Song

3.1.1 Introduction to the data

This database (Livingstone SR, 2018) contains 24 (12 male, 12 female) professional actors vocalising two lexically matched statements in a neutral North-American accent. It contains both spoken and sung versions of the sentences in eight distinct emotional classes: Neutral, Calm, Happy, Sad, Angry, Fearful, Disgusted, Surprised. The dataset contains video as an additional modality that will not be used within this thesis.

This dataset is excellent for investigations into the audio-signal effects of emotional speech, i.e. how the voice changes during different extremes of emotion. But as the sentences are identical, it cannot be used for multi-modal investigation into sentimental and emotional human speech. Additionally as the dataset is acted it may not be a true representation of the voice qualities indicative of the listed emotions.

3.1.2 Relevant Statistics

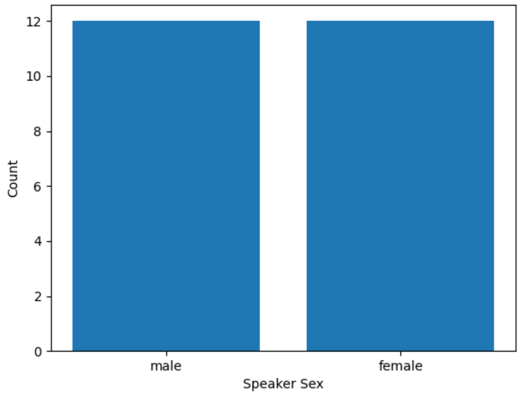


Figure 3.1: The Distribution of Speaker Sex in the RAVDESS Dataset

As mentioned in the summary of the dataset, there are 24 speakers: 12 Male and 12 Female, giving a perfectly balanced distribution of speaker sex.

There are a total of 8 acted emotions present in the dataset, they each feature "low intensity" and "high intensity" examples. Except for the neutral emotion which does not have examples of "high intensity" as this is not a possible combination.

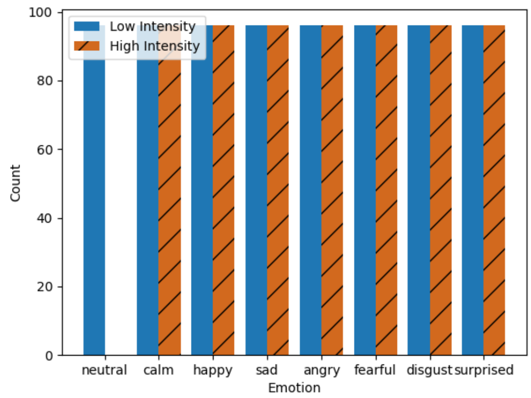


Figure 3.2: The Distribution of Emotion in the RAVDESS dataset

3.1.3 Preprocessing Considerations

As will be discussed in more detail in Chapter 4, the primary tool to preprocess these audio waveforms was to convert them into Mel-frequency-cepstral-coefficient (MFCC) Muda, Begam, and Elamvazuthi, 2010 vectors. The only other preprocessing consideration was to trim the initial sections of the provided audio file which have an amplitude of 0, as this prevents the proper calculation of a Fourier transform that underpins the MFCC process.

3.1.4 Justification for Inclusion

This dataset is used in the early work of this thesis in order to demonstrate classification of emotion on recordings of spoken language. As will be seen in Chapter 4, this suggests promising avenues for further research involving multi-modal classification of emotion (classification of both the speech and transcriptions of the speech).

3.2 SemEval-2017 Task 4: Sentiment Analysis in Twitter

3.2.1 Introduction to the data

This dataset (Rosenthal, Farra, and Nakov, 2017) [1] is part of a published competition by the Workshop on Semantic Evaluation. For the purpose of this report it will be referred to as SE-2017-T4. The dataset features a large collection of tweets ($n \approx 48,000$) on a variety of topics, that have each been hand labelled for sentiment, either positive, negative, or neutral.

3.2.2 Relevant Statistics

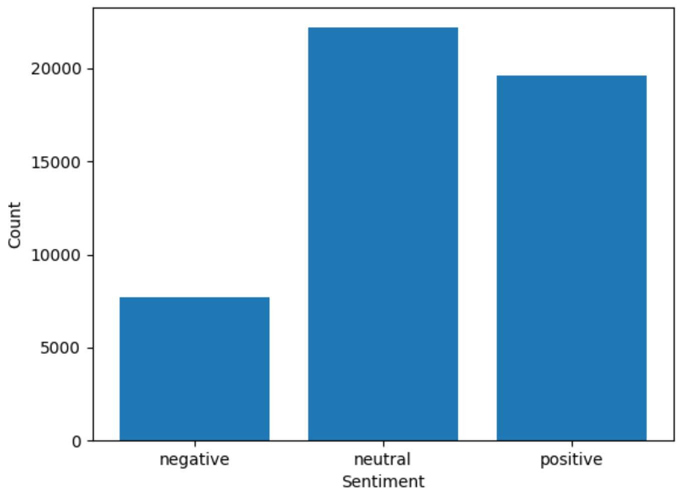


Figure 3.3: The Distribution of Sentiment in the SemEval dataset

Star rating: The distribution of sentiment is close to balanced for positive and neutral. but there are approximately 1/3 as many negative items as there are neutral items. So this dataset is considered to be biased away from the negative sentiment class.

[1] available at `https://alt.qcri.org/semeval2017/task4/index.php?id=data-and-tools`

Word count: There are a significant number of tweets that are precisely 25 words long, this is the peak of the distribution. Additionally the majority of tweets (95.1%) are 33 words or less. In Experiments in Chapter 5 the precise word count used for truncation and padding is selected as part of the hyper-parameter tuning, rather than being selected just off these statistics.

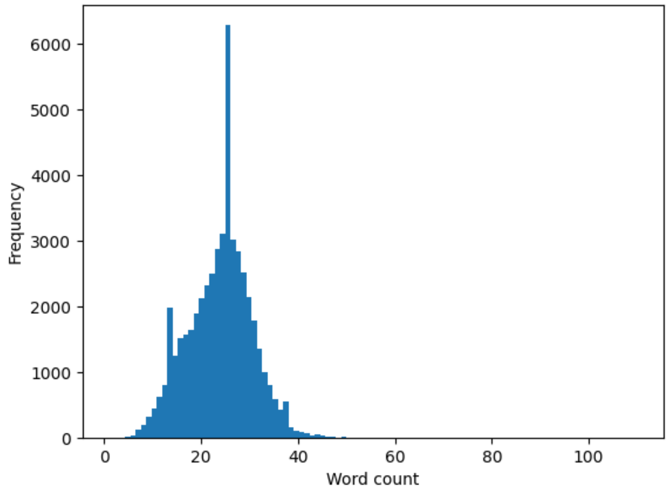


Figure 3.4: Histogram of the word count of tweets in SemEval

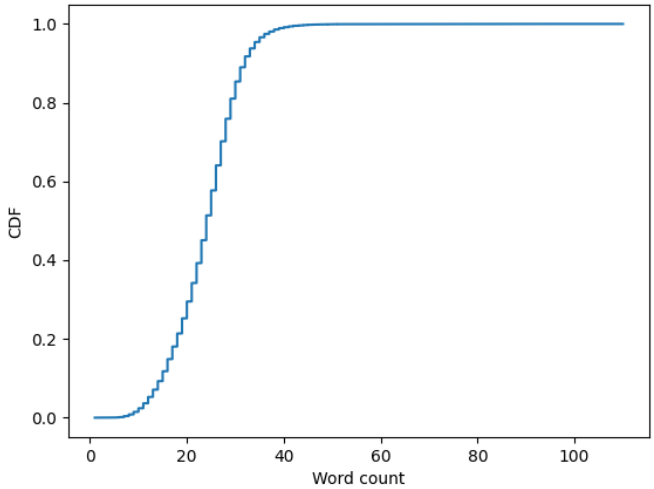


Figure 3.5: Cumulative frequency curve of the word count of tweets in SemEval

3.2.3 Preprocessing Considerations

The preprocessing methodology for this dataset is based on the preprocessing methodology defined for the pre-trained twitter methodology used in the glove models (Pennington, Socher, and Manning, 2014), the original ruby implementation of this pre-processing can be found at [2] but this was re-implemented for this thesis using python, with the system and software versions detailed in 1.4

- The text has any alternate versions of standard characters substituted for consistency
- Any URLs are replaced with a `<URL>` token
- Any twitter usernames i.e. `"@someName"` are replaced with a `<USER>` tag
- Any twitter hashtags i.e. `"#SomeHashTag"` are replaced with a token and the hashtag is split using the casing i.e. `"<HASHTAG> some hash tag"`
- Any numbers are replaced with a `<NUMBER>` token
- Repeated punctuation i.e. `"!!!"` is replaced with a repeat token: `"! <REPEAT>"`
- elongated standard words such as `"soooo"` are replaced with an elongated token: `"so <ELONGATED>"`
- The text is then split into unique tokens using a tokeniser (Bird, Klein, and Loper, 2009)
- The tokens are then vectorised using the pretrained word2vec model (Mikolov et al., 2013)(Pennington, Socher, and Manning, 2014) in either 25, 50, 100, or 200 dimensional variants depending on the experimental parameters.

[2] https://nlp.stanford.edu/projects/glove/preprocess-twitter.rb

3.2.4 Justification for Inclusion

This dataset was included for two reasons: Primarily it is a sentiment labelled dataset that has been well studied and can be used to robustly demonstrate sentiment analysis on casual conversational text. Additionally when combined with the Amazon Review datasets described in 3.3, 3.4, SemEval can be considered as a "highly dissimilar domain" for transfer learning with respect to the amazon reviews.

3.3 Amazon Review Data (2018)

3.3.1 Introduction to the data

The Amazon Review Dataset (Ni, Li, and McAuley, 2019) [3] Is the primary dataset used in a large portion of this research. It features an extremely large number of amazon reviews (233.1 million) across a wide variety of categories. That include star rating labels that are considered to be a soft representation of sentiment for the purpose of this research.

In this thesis the sub-dataset of "5-core" (Where each reviewer has made at least 5 reviews, and each product has received at least 5 reviews) is used. This dataset contains a total of 75.26 Million items. For the majority of experiments, unless specified otherwise, a further randomly sampled subset of a maximum of 64,000 items per each of the 29 categories was used. These precise counts will be given below in 3.3.2

Additionally, for some experiments in this thesis, it was was useful to have balanced datasets. As such for the following domains: *Automotive*; *ToolsHomeImprovement*; *Electronics*; and

[3]available at `https://nijianmo.github.io/amazon/index.html`

Books; a dataset of 90,000 items per domain was created. This dataset has 15,000 1-star, 15,000 2-star, 30,000 3-star, 15,000 4-star, 15,000 5-star items (in order to create 30,000 each of [negative, neutral, positive] reviews).

3.3.2 Relevant Statistics

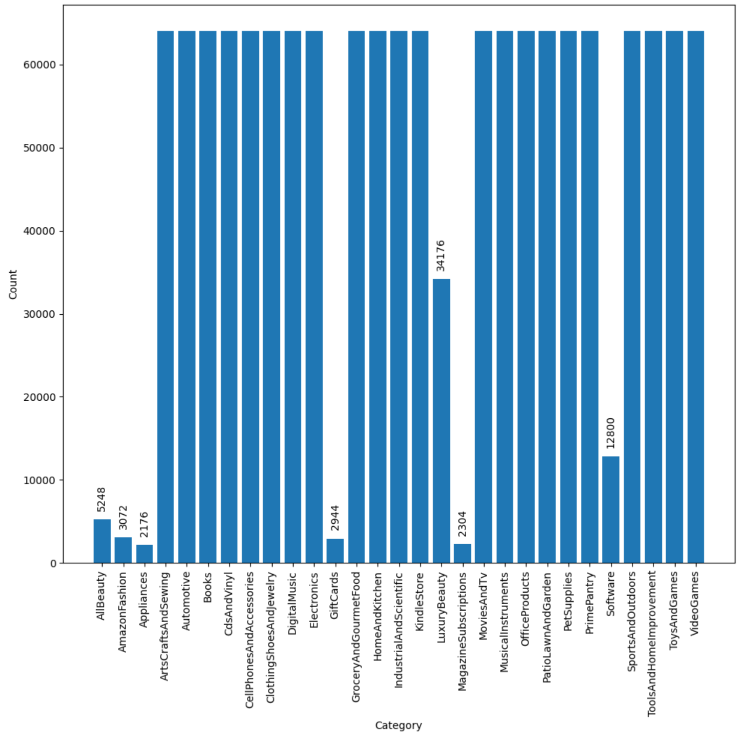


Figure 3.6: The item counts per category in the Amazon Review Dataset

Items per category: As mentioned in the dataset summary, to reduce training times and to provide a more balanced dataset for categories with less than 64,000 items. A maximum of 64,000 items were used per category unless explicitly stated otherwise.

Star rating: The star rating distribution in the set is heavily biased toward higher star ratings, it is interesting to note that 2 has the lowest count of ratings, being marginally lower than the number of 1 star ratings in the dataset. For some experiments in this thesis a different subset of data was used with a precisely balanced count of star ratings in order to mitigate the data imbalance. This will be clearly stated for the experiments where it is used.

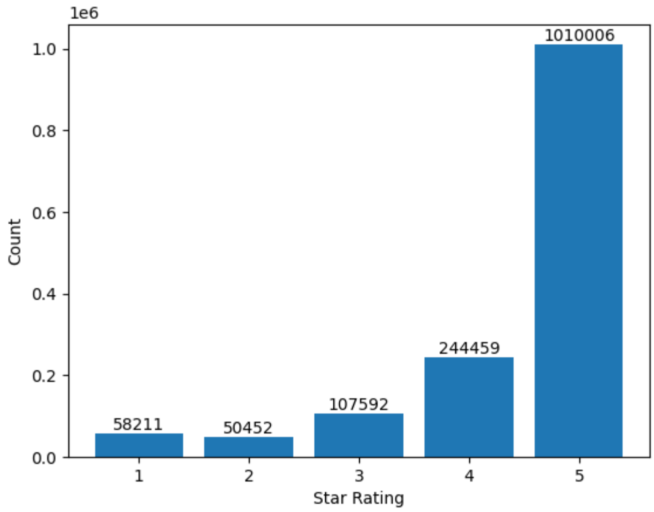


Figure 3.7: The Distribution of Star Ratings in the Amazon Review Dataset

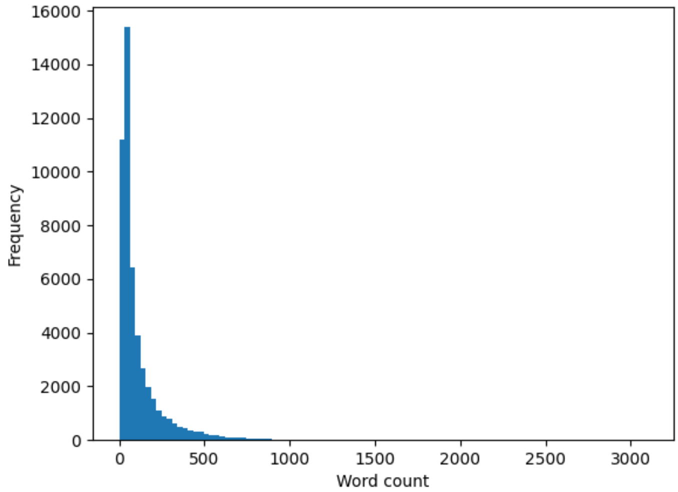


Figure 3.8: Histogram of the word count of amazon reviews

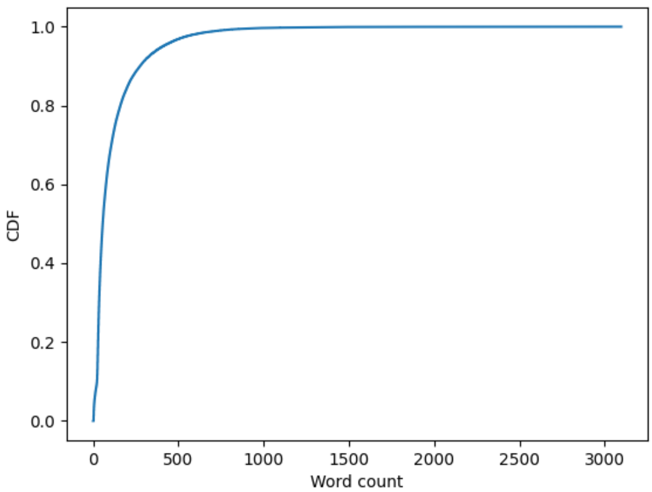


Figure 3.9: Cumulative frequency curve of the word count of amazon reviews

Word count: As can be seen from the figures, some of the amazon review data items were up to 3200 words in length. However the majority (over 95.2%) of the dataset is 400 words or less. Therefore for all experiments using this dataset within this thesis, the reviews were truncated or padded to a length of 400 words after pre-processing.

3.3.3 Preprocessing Considerations

The pre-processing methodology for this dataset is a simplified version of that used for the SEMEVAL dataset(3.2) based on the methodology used when training the Glove model (Pennington, Socher, and Manning, 2014):

- The text has any alternate versions of standard characters substituted for consistency
- Any URLs are replaced with a `<URL>` token
- Any numbers are replaced with a `<NUMBER>` token
- Repeated punctuation i.e. `"!!!"` is replaced with a repeat token: `"! <REPEAT>"`
- elongated standard words such as `"soooo"` are replaced with an elongated token: `"so <ELONGATED>"`
- The text is then split into unique tokens using a tokeniser (Bird, Klein, and Loper, 2009)
- The tokens are then vectorised using the pretrained word2vec model (Mikolov et al., 2013)(Pennington, Socher, and Manning, 2014)

Using this preprocessing methodology, and the same pretrained word2vec model, ensures consistency with the SemEval dataset which allows for reliable experimentation across datasets.

3.3.4 Justification for Inclusion

This dataset provides a large number of different product domains, which are labelled for sentiment with the use of star-ratings. This is one of the primary datasets used within the thesis to demonstrate techniques for transfer learning sentiment between domains. It also provides a high-volume "source" domain of data that can be combined with the SemEval dataset described in (3.2) to demonstrate transfer learning between dissimilar domains (where different categories of products are considered to be different but similar domains).

3.4 Multi Domain Sentiment Dataset 2.0

3.4.1 Introduction to the data

The Multi Domain Sentiment Dataset 2 (Blitzer, Dredze, and Pereira, 2007 [4] (MDSD2) Features a smaller collection of amazon reviews for a variety of products. The reviews have a star rating from 1 to 5 stars, and we also consider this to be a self assessment of the sentiment of the review. For the sake of this work, we will be considering 1 and 2 star reviews to be "negative", while 4 and 5 star reviews will be "positive". This dataset does not include 3 star reviews as it is designed for positive/negative classification tasks.

[4] available at `https://www.cs.jhu.edu/ mdredze/datasets/sentiment/`

3.4.2 Relevant Statistics

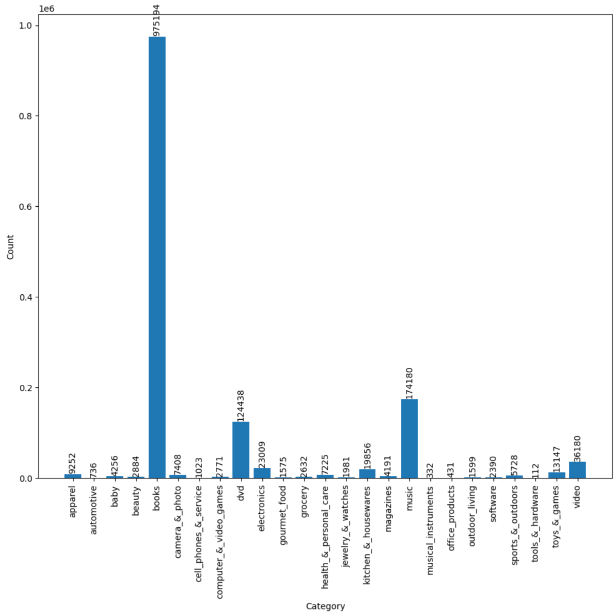


Figure 3.10: The original review volumes by category for MDSD2

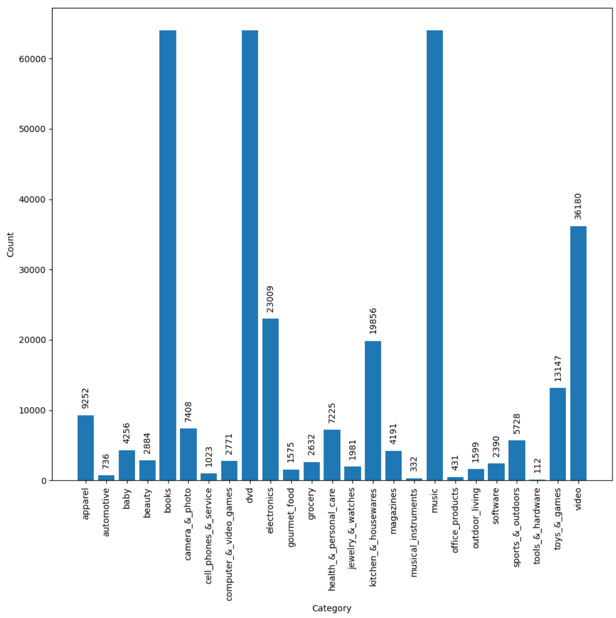


Figure 3.11: The truncated review volumes in MDSD2 used in this thesis

Items per category: These figures show the number of reviews for available for each category, both in the original raw dataset, and a version used in this thesis where each category is truncated to a maximum of 64,000 reviews. It is notable that the available review volume for books is far higher than for any other product. This is due to the age of the dataset, where at the time Amazon was primarily a book seller with some other products available.

Star rating: This figure shows the star rating distribution in the truncated version of the dataset discussed above. The distribution of 1, 2, 4, and 5 star reviews is comparable to the dataset discussed in 3.3, except for the notable lack of 3 star reviews. This dataset is produced explicitly for the task of positive/negative sentiment classification, and the 3 star reviews have been omitted as they are considered "neutral" for sentiment.

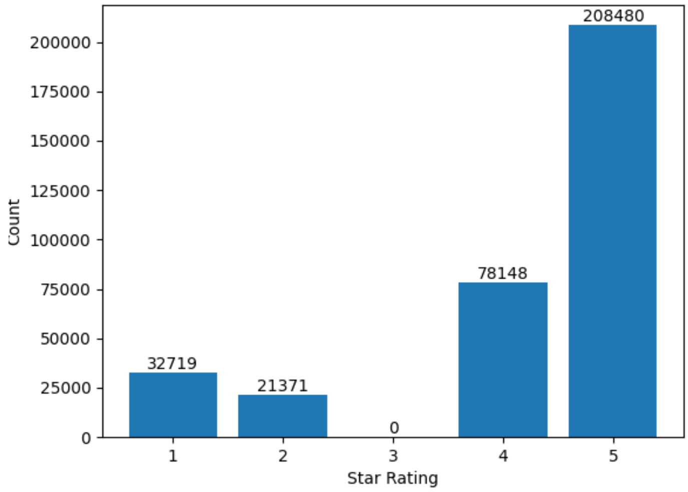


Figure 3.12: The star rating distribution of the MDSD2 dataset

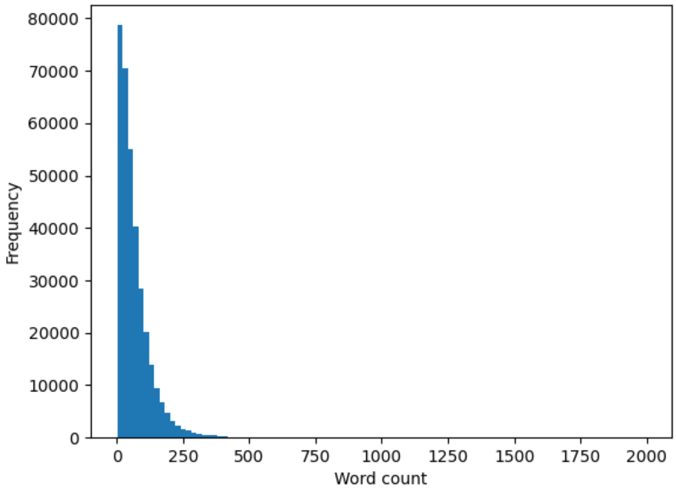


Figure 3.13: Histogram of the word count of reviews in the MDSD2 dataset

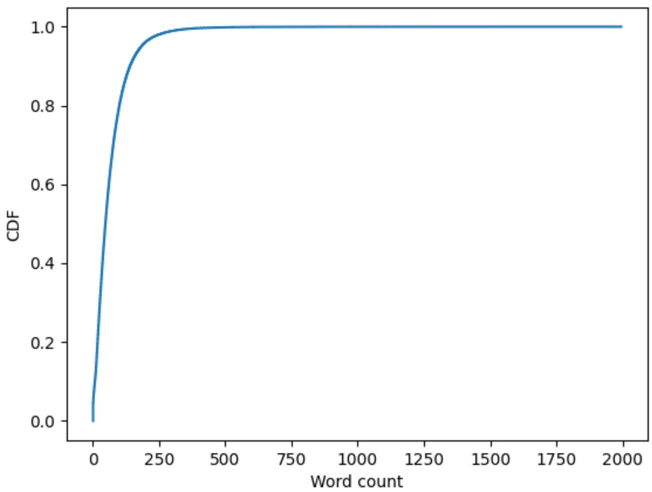


Figure 3.14: Cumulative frequency curve of the word count of reviews in the MDSD2 dataset

Word count: This distribution is similar to the distribution seen in figures 3.8, 3.9. the MDSD2 dataset features a slightly shorter average review length, and a shorter maximum review length. But for the purpose of this thesis, to allow for consistent cross dataset experimentation. This dataset will also be truncated/padded to 400 words.

3.4.3 Preprocessing Considerations

For consistency, an identical preprocessing methodology to that described in 3.3.3 was used.

3.4.4 Justification for Inclusion

This dataset is similar to the Amazon Review dataset described in 3.3. It was selected for inclusion in this thesis as it has been well studied in a number of papers (Ericsson et al., 2022)(Jindong Wang et al., 2022) which allows for an effective comparison between methods discussed in this thesis and methods from the wider literature.

3.5 Motivational Interview (MI) Data

The Motivational Interview (MI) Dataset is the primary target dataset of this thesis. It features audio recordings of patients and clinicians performing long form (approximately 30 minute) "motivational interviews" in order to therapeutically guide the patients towards better self-management of their conditions (in this case, diabetes). These interviews are then given a short post interview assessment by the clinician, and this is to be considered the labelling scheme for data analysis work.

This dataset is unpublished and was collected by the Institute of Psychiatry, Psychology and Neurosciences (IoPPN) at Kings College London. The transcriptions were generated in part by the work of (Wei, 2022).

As mentioned previously, this data has a number of additional difficulties, the data is recorded on a single (static noisy) microphone, and there is generally a significant amount of background noise. Coupled with the fact that it is a recording of a two party conversation, this make automatic transcription and speaker separation somewhat challenging.

3.5.1 Relevant Statistics

In all of the statistics below, a "sentence" in this context is considered to be a single complete utterance from one of the two parties in the conversation, this can be a single word such as "yes" or "no". Or could be multiple complete clauses.

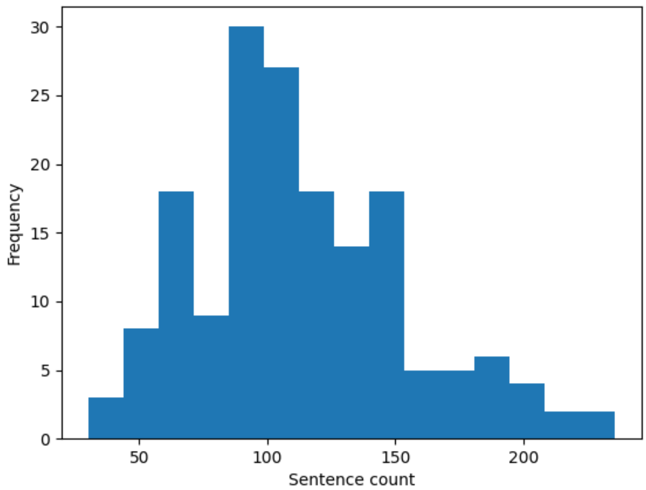


Figure 3.15: The distribution of sentence counts in the MI Transcripts

Sentences per transcript: The transcripts are extracts of full recordings, typically 10 minutes worth of conversation between 2 parties. As such the upper limit on number of sentences is approximately 200 (100 sentences from each party) with the peak at 100 (50 from each party).

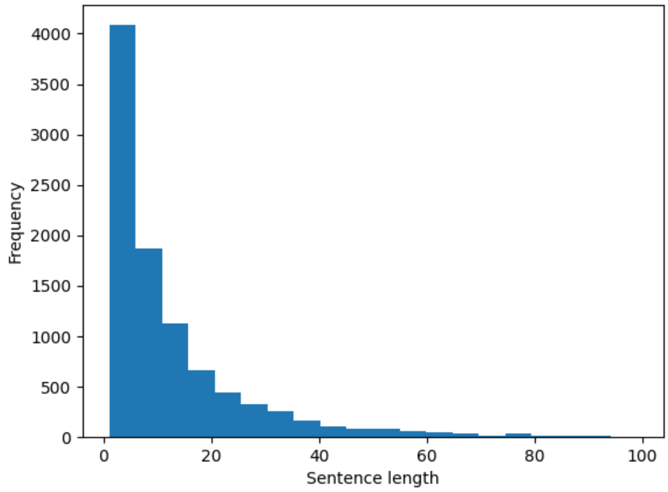


Figure 3.16: Histogram of the word count of patient sentences in the MI Transcripts

Word count: The distribution of word count making up a single "sentence" is almost identical for the patients and the clinicians across the transcripts. With a skew toward very short sentences (less than 5 words). The skew toward short sentences is typical of informal spoken communication. And the almost identical distribution of word counts reinforces that the intent of these clinical conversations is to be conversations rather than one-sided instructional lectures.

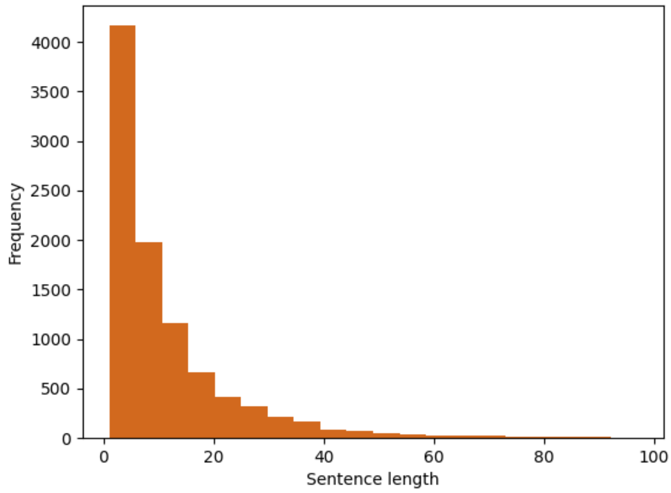


Figure 3.17: Histogram of the word count of clinician sentences in the MI Transcripts

Sentiment Labels

This dataset was not provided with sentiment labels. In order to allow experimentation in the context of this thesis, the author labelled the provided data for sentiment, at a whole document level as well as an individual sentence level. The statistics from that labelling are given below. These labels cannot be considered as gold-standard labels as the author is not an expert reviewer of human sentiment, and as there was only a single annotator, no cross-annotator agreement statistics can be generated. It is recommended that future work on this topic should have at least a statistically significant number of motivational interviews labelled for sentiment by experts to allow robust conclusions to be drawn.

Document-wise sentiment: The distribution of document-wise sentiment is heavily skewed toward neutral, followed by negative, with only 6 of the 169 documents being notably positive. This is an unsurprising distribution due to the inherent nature of medical conversations.

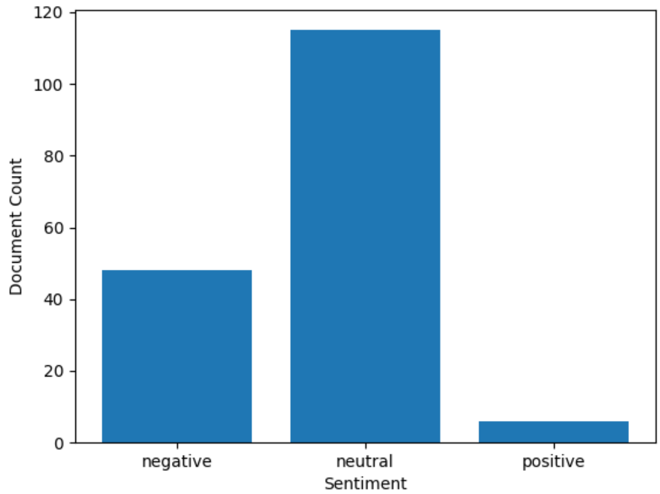


Figure 3.18: Distribution of document-wise sentiment in the MI data

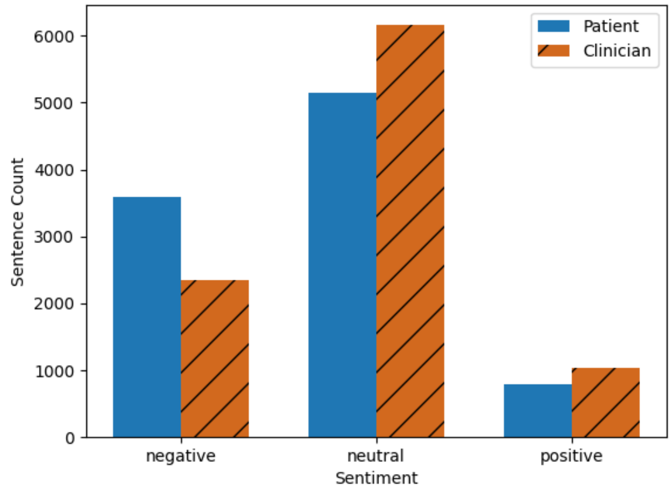


Figure 3.19: Distribution of sentence-wise sentiment in the MI data

Sentence-wise sentiment: The distribution of sentence level sentiment is skewed similarly to the document-wise, but with a slight increase in positive sentiment at the sentence level, this is mostly explained by greetings and other customary polite exchanges that can still be part of a generally negative or neutral-informative conversation. It is also noteable that the clinician sentences are slightly more positive, and signficantly less negative than the patient sentences on average.

3.5.2 Preprocessing Considerations

This dataset represents a significant preprocessing challenge. The raw transcript data is provided in `.doc` files featuring headers and preamble regarding the timestamp of the start time of the transcription. The data is then a set of lines separated by carriage-returns, where the patient sentences are at the base indentation, and the clinician sentences are indented by 1 level.

There transcripts feature some inline annotations from the reviewer(s) such as: `(info gathering)` or `(Affirmation; could add a summary here to reinforce pt about regular routine)`.

Sections of the transcriptions that were not understandable by the transcribers have been marked with `[...]` or `[____]` (this inconsistent standard is likely the result of different transcribers) with the timestamp of the inaudible section filled in red font. Occasionally this has then been modified with an updated transcription within the square brackets.

In order to preprocess this for use, the following steps were taken.

- extract all the text for a document between the `START TIME` and `END TIME` flags
- split the conversation into unique utterances on the `\n\r\n\r` tag (two carriage returns)
- identify whether the utterance is a patient or clincian based on the indentation
- extract any inaudible sections marked by `[...]` or similar, and replace them with the `<unknown>` token
- replace ... with the `<PAUSE>` token
- remove any additional whitespace generated by parsing from the `.doc` format
- extract the annotations, denoted by circular brackets from the text and store these separately in a linked data structure
- follow all the remaining steps listed in 3.3.3 in order to generate word-vectors that are compatible with the other datasets used in this thesis

3.5.3 Justification for Inclusion

This dataset is representative of the Motivational Interviews and other Clinical Conversations that are the primary motivator for the work conducted in this thesis. While the scope of experiments conducted on it are more limited (due to the lack of available gold-standard labelling), it is included and experimented on in Chapter 8 in order to demonstrate that the techniques discussed in this thesis are applicable to this style of data.

Chapter Four

Emotion Classification of Vocalised Speech

This chapter's primary aim is to introduce the parameter tuning methodologies that will be used throughout the thesis. We will introduce the methodology using a worked example, and then demonstrate how it can be applied to the task of creating an emotion classification system for spoken utterances. The secondary goal of this chapter is to demonstrate the effectiveness of the discussed methodologies in classifying emotion in spoken utterances by experimentation on the RAVDESS (3.1) dataset. As will be discussed in Chapter 9 this informs the recommendation that further works should explore multi-modal sentiment analysis using both the audio data and the transcription of spoken conversations.

4.1 Explanation of Parameter Tuning Methodology

In this section we will explain Bayesian Hyperparameter optimisation, and explore a simplified worked example so that the plots produced by hyperparameter optimisation in later sections of the thesis can be clearly interpreted.

4.1.1 Bayesian Hyperparameter Search

Bayesian hyperparameter optimization (Wu et al., 2019) is a probabilistic approach to finding the optimal set of hyperparameters for machine learning models. Unlike traditional methods such as grid search or random search, which sample hyperparameter values either exhaustively or at random, Bayesian optimization builds a model of the objective function to predict which hyperparameters are likely to perform well. This is done by treating the optimization process as a probabilistic inference problem. The method works by first selecting a small number of hyperparameter combinations and evaluating the model's performance for each. Based on these initial results, a surrogate model, typically a Gaussian process, is built to approximate the relationship between the hyperparameters and the model's performance. The surrogate model is then used to estimate the performance of untested hyperparameter combinations, guiding the selection of the next set of hyperparameters to explore.

The key advantage of Bayesian optimization is that it balances exploration and exploitation efficiently. It uses an acquisition function, to decide whether to explore areas of the hyperparameter space where performance is uncertain (exploration) or focus on areas that are predicted to yield the best results (exploitation). This allows Bayesian optimization to converge to the optimal hyperparameters more quickly than traditional methods. Over time, as more hyperparameter configurations are evaluated, the surrogate model becomes more accurate, and the optimization process homes in on the best hyperparameters, minimizing the number of evaluations required while still improving the model's performance.

4.1.2 Setup and hypotheses of worked example

In this section, we present a simplified worked example to demonstrate how Bayesian hyperparameter optimisation can be used to identify optimal parameters. The example consists of a model where the prediction is defined as a product of the input and two parameters, a and b, such that the prediction is given by:

$$\text{prediction} = \text{input} \times a \times b$$

The target label for the optimisation is simply the input itself. The goal of the hyperparameter optimisation is to find an appropriate pair of values for a and b such that their product, $a \times b$, is as close as possible to 1. In other words, the objective is to discover a set of parameters a and b that minimise the difference between the prediction and the target label, which would lead to a prediction of:

$$\text{prediction} = \text{input} \times 1 = \text{input}$$

This setup allows us to assess how Bayesian optimisation can efficiently explore the search space of a and b to discover values that satisfy $a \times b \approx 1$. The underlying hypothesis of this experiment is that the Bayesian optimisation process will converge to an optimal pair of parameters, leading to minimal prediction error. Additionally, we will demonstrate that the Bayesian process selects potential parameters to efficiently determine the boundaries of the distribution in a way that is considerably more efficient than alternate methods such as grid search. And can optimise an arbitrary number of parameters with a relatively low number of iterations.

Finally we will expand this problem to the objective function of $a \times b \times c \times d = 1$ to explore how this process responds with a higher-dimensional search space, and the impact of adjusting the number of search iterations on the performance.

4.1.3 Results and Discussion

Two dimensional search space optimisation

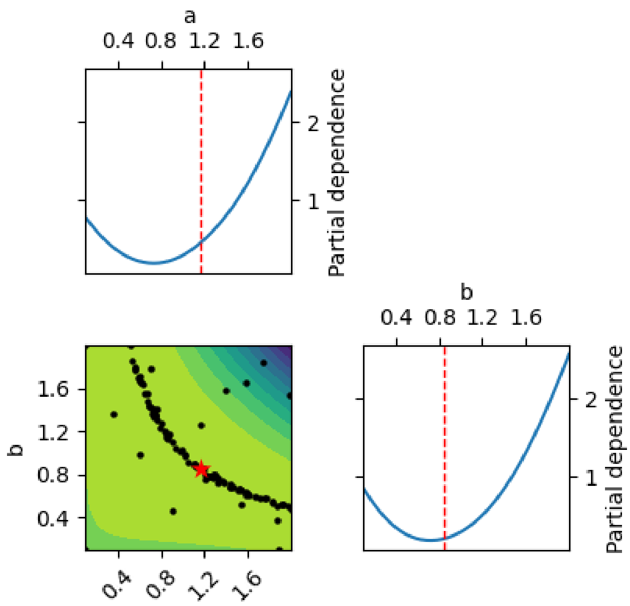


Figure 4.1: The partial dependence plot of example Bayesian Optimisation

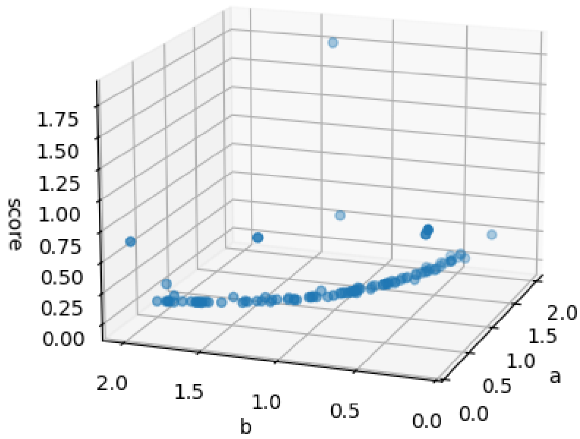


Figure 4.2: Visualisation of search space explored by example Bayesian Optimisation

Figure 4.2 shows the solutions explored by the system over the course of 100 optimisation iterations, with the Z axis showing the score of those solutions. While there are a small number of solutions that are not along the curve of $a \times b = 1$ (which mean their score is higher - with the task being score minimisation). The system was able to quickly discover a local minima cost and then devoted the majority of the search iterations to exploration of the bounds of that minima. This demonstrates that the Bayesian Optimisation process is a highly targeted method for parameter optimisation, that does not aim to explore the entire parameter search space, instead exploring the gradients within the search space.

Four dimensional search space optimisation

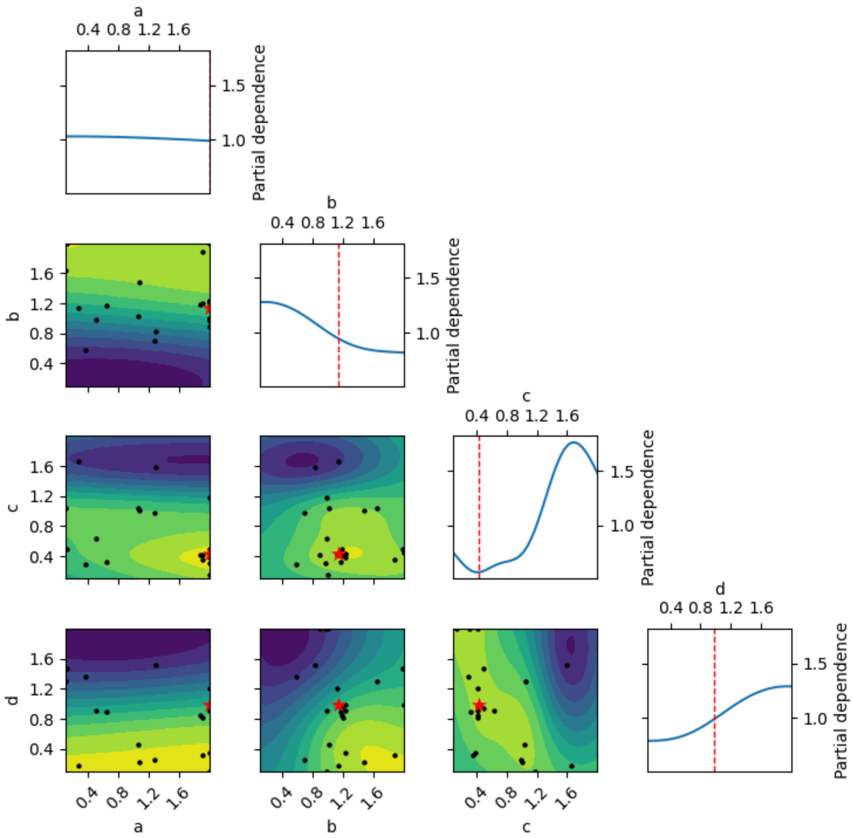


Figure 4.3: A partial dependence plot of 4 parameter Bayesian optimisation for 25 iterations

While the distributions of partial dependence do not perfectly represent the objective function of $a \times b \times c \times d = 1$ the system was able to discover multiple possible parameters that provide a local minima within 25 iterations of searching the space. The best discovered solution as per this plot is $2.0 \times 1.136 \times 0.436 \times 0.984 = 0.974$

This demonstrates the effectiveness of this system for highly dimensional optimisation problems even with a relatively low number of optimisation iterations.

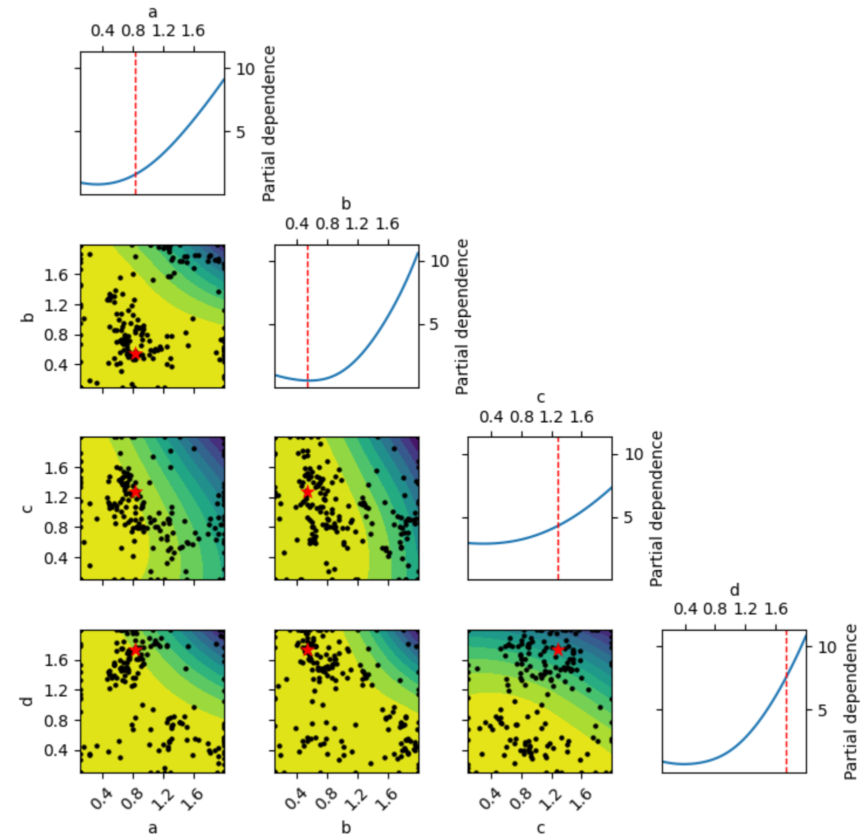


Figure 4.4: A partial dependence plot of 4 parameter Bayesian optimisation for 200 iterations

For completeness and comparison to the lower confidence solution. The figure 4.4 demonstrates the same optimisation problem of $a \times b \times c \times d = 1$ after running for 200 iterations. It can be seen that the distributions are a more true representation of the problem space, and the optimal solution discovered of $0.831 \times 0.544 \times 1.278 \times 1.729 = 0.999$ is a better solution than that discovered after 25 iterations. But this represents a marginal gain in confidence of the distributions, and optimality of the solution for 8 times the runtime.

It is important to recognise the potential for mistakenly selecting parameters based on this plot. Each of the 4 variables independently shows that its optimal (the value at which the partial dependence is lowest) value is approximately 0.5. However it is not appropriate to then simply select this value for all 4 variables and expect that this will yield an effective solution (in this case this would give a value of 0.0625 which is very far from the optimal value of 1).

Conclusions

From this worked example we have demonstrated the effectiveness of the Bayesian Hyperparameter optimisation technique even for optimising multiple interacting hyperparameters at the same time. We have shown that it is a focused technique that is able to discover close to optimal solutions with relatively low numbers of optimisation iterations. And we have explored the partial dependence plots in so that the conclusions drawn from these plots in later sections of this thesis can be confirmed by the reader.

When reading the partial dependence plots, the general rule is that we are aiming to select parameter values that minimise the partial dependence. However as mentioned, this cannot be applied as a sweeping rule across all parameters based on the result of a single optimisation. In practice when applying this process, we will aim to perform an initial run of a very large number of hyper-parameters and identify and remove parameters that have an almost flat partial dependence curve, meaning that their value has almost no impact on the performance of the system. Then subsequent runs of a lower dimensional search space can be performed in order to identify a locally optimal selection of hyper-parameters for the objective.

4.2 Classifier Methodologies

The primary preprocessing methodology used in this investigation is based on the use of mel frequency cepstral coefficients (MFCC) (Muda, Begam, and Elamvazuthi, 2010) vectors as an input vector to a classifier. MFCC vectors are a form of spectrographic vector constructed using spectrogram techniques, but have frequency bands optimized for use with human speech. Additionally it is common to include the delta (velocity) and often the delta-squared (acceleration) of the MFCC vectors as part of an input when using them for classification. This allows modelling of the dynamic features of the signal using a non-dynamic classification method.

For this chapter we will be comparing two neural network based classifier methodologies

4.2.1 Fully connected Network

The first system that we will optimize and evaluate will be an n-hidden layer fully connected neural network. For a given utterance that creates x number of frames of MFCC vectors, the network will be duplicated x times. Each MFCC frame will be inferenced, and the result of the output layer will be averaged across the x outputs in order to create a prediction. The number, and dimension of the hidden layers used by the network will be a tunable parameter available to the parameter optimisation system.

For example for a network with its configuration denoted by $(64, 32)$ would be constructed as in figure 4.5, this network would then be duplicated for frames t_0 to t_x for training or inference. At training time, it is assumed that each frame of an utterance has the classification of its parent (i.e. every frame of an "angry" utterance is labelled as "angry") so each MFCC vector forms a single complete input item.

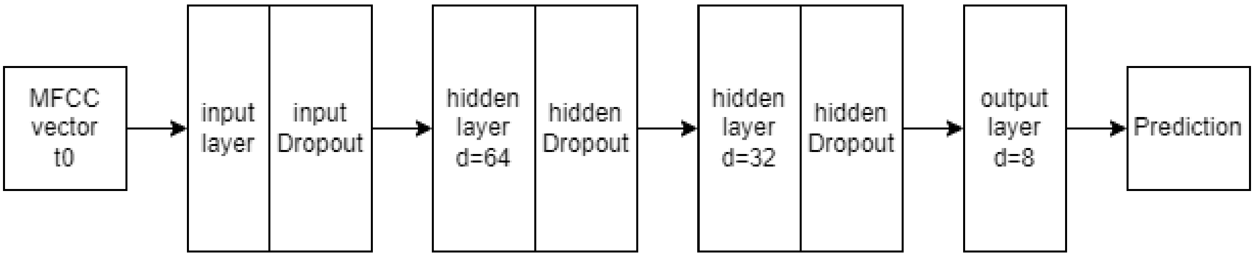


Figure 4.5: An example fully connected network configuration

The configurations available for the hyper-parameter tuning as as follows:

Table 4.1: RAVDESS Fully Connected Network Configurations

Configuration Number	**Configuration**
0	(32, 16)
1	(64, 32, 16)
2	(128, 64)
3	(512,)
4	(256, 128)
5	(64, 64, 64, 64, 64, 64, 64, 64, 64)
6	(512, 256, 128)
7	(1024, 512, 256)
8	(2048, 1024, 512)

4.2.2 Recurrent Neural Network (RNN)

A Recurrent Neural network implementation (Lee and Tashev, 2015) of a classifier has also been constructed for this task. Taking A sequence of MFCC based input vectors as input in order to produce a single output. At both training and inference time, the system takes all MFCC vectors from the utterance as a series of inputs and produces a single prediction of the emotion of the utterance. The RNN uses a multi-layer approach with the depth and number of hidden units in each layer available as a tunable parameter to the optimisation system. For example a network with the configuration $(64, 32)$ will be constructed as in figure 4.6. The available configurations are altered during the parameter optimisation experiments in

4.3.2 and will be listed separately during that section of the thesis.

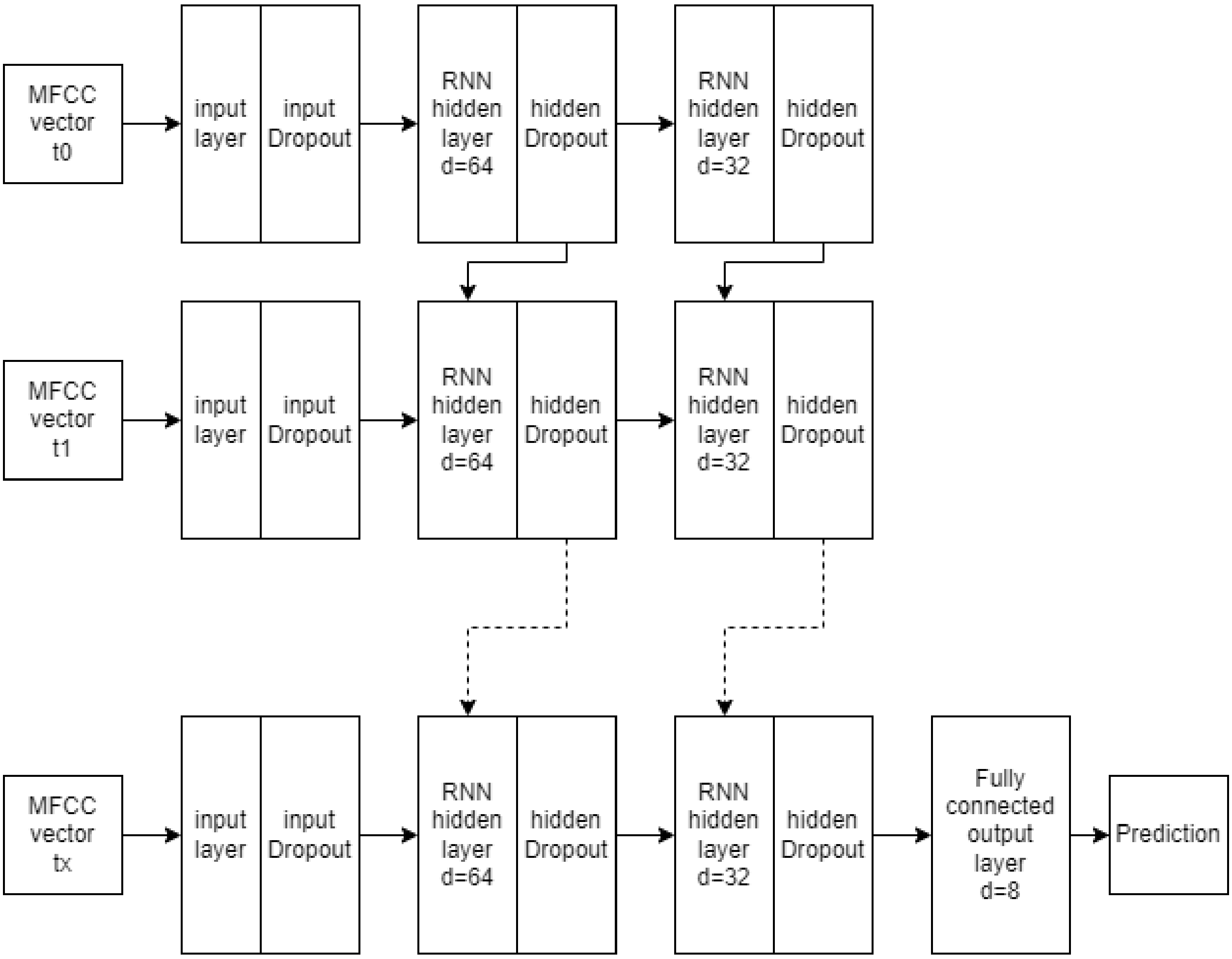


Figure 4.6: An example RNN classifier configuration

4.3 Hyperparameter tuning of classifiers for the RAVDESS dataset

In this section we will apply the Bayesian hyperparameter optimisation method discussed in 4.1 to the classifier systems defined in 4.2 in order to discover optimal configurations for application to this problem.

For all the hyper-parameter tuning in this section, the scoring metric used was "macro-averaged F1 score" (the mean of the F1 score of all classes for a given test set). While this is a balanced dataset with respect to emotional classes, so accuracy may have been an appropriate metric, the majority of datasets discussed in this thesis are unbalanced, and there is little downside of using F1 in balanced datasets. All of the hyperparameter tuning experiments below utilise 3-fold cross validation, using 20 speakers (10 male and 10 female) for training data, and 4 speakers (2 male and 2 female) for evaluation of the objective.

4.3.1 Hyper-parameter Optimisation of a Fully-Connected Neural-Network based system

Phase 1 search

For the first phase of the hyper-parameter optimisation process, a search space for the significant majority of selectable parameters was created, given in the table 4.2. It is important to note that while there are known "standard" values for FFT and MFCC generation for human speech, these parameters were made available to the optimiser to ensure that the problem space was fully explored.

Table 4.2: RAVDESS Fully Connected phase 1 search space

Hyperparameter	Type	Range / Values
n_mfcc	Integer	2 – 60
delta_order	Categorical	{0, 1, 2}
n_fft	Integer	128 – 2048
hop_length	Integer	256 – 1024
n_mels	Integer	16 – 256
network_structure	Categorical	{0, 1, 2, 3, 4, 5, 6}
input_dropout	Real	0.0 – 0.5
hidden_dropout	Real	0.1 – 0.75
hidden_activation	Categorical	{leaky_relu, sigmoid}
optimizer	Categorical	{adam, rmsprop}
batch_size	Integer	32 – 128

This optimisation was run for 100 iterations, the partial dependence plot output for this optimisation is given below in figure 4.7

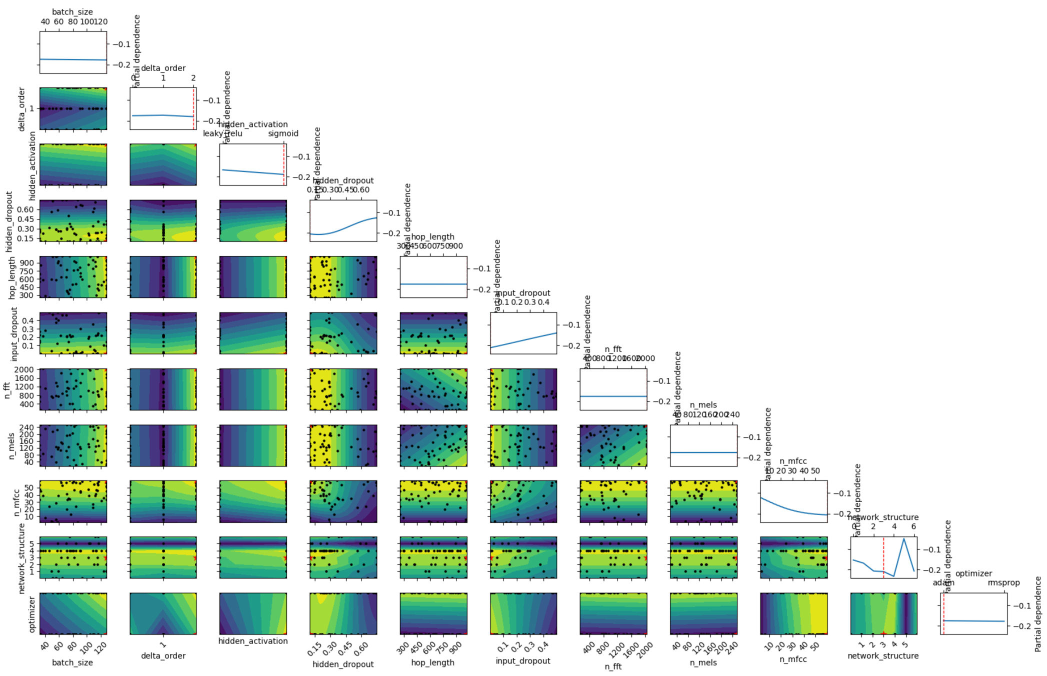


Figure 4.7: The phase 1 partial dependence plot for parameter optimisation of a Fully Connected NN system

Based on the output of the partial dependence plot 4.7. the following values were fixed for the next phase of parameter exploration

Table 4.3: RAVDESS Fully Connected phase 1 selected parameters

Parameter	Value
n_fft	1024
n_mels	64
optimizer	adam
batch_size	64

Phase 2 search

The reduced search space and output partial dependence plot based on 100 iterations of optimisation are given below in table 4.4 and figure 4.8 respectively.

Table 4.4: RAVDESS Fully Connected phase 2 search space

Hyperparameter	Type	Range / Values
n_mfcc	Integer	30 – 60
delta_order	Categorical	{0, 1, 2}
hop_length	Integer	256 – 1024
network_structure	Categorical	{2, 3, 4, 6}
input_dropout	Real	0.0 – 0.5
hidden_dropout	Real	0.1 – 0.75
hidden_activation	Categorical	{leaky_relu, sigmoid}

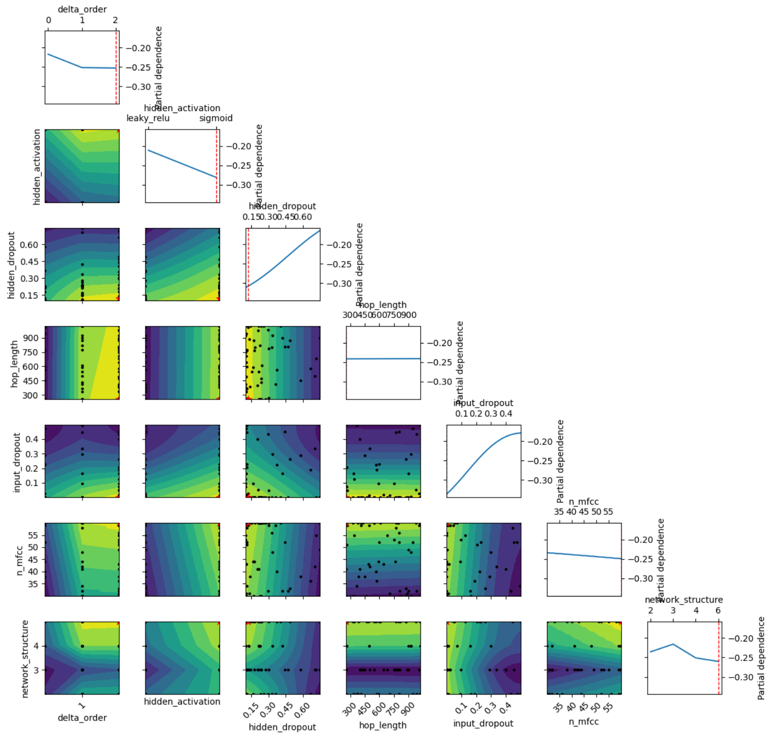


Figure 4.8: The phase 2 partial dependence plot for parameter optimisation of a Fully Connected NN system

Based on the output of the partial dependence plot 4.8, the following values were fixed for the next phase of parameter exploration.

Table 4.5: RAVDESS Fully Connected phase 2 selected parameters

Parameter	**Value**
`hop_length`	768
`delta_order`	2
`hidden_activation`	`sigmoid`

Phase 3 search

For the final phase of the search some changes were made to the process. As the results of phase 2 demonstrate an improved performance with increasing the network complexity, which does not appear to have reached a performance maxima, some additional network configuration options were added to the search space (denoted by configuration 7 and 8 in table 4.1. Additionally, The number of training epochs were given as an additional tunable parameter, this can be used to quantify over-fitting, and is useful to observe the 2D gradients of other parameters with respect to the optimal number of training epochs.

The search space used for phase 3 is given below:

Table 4.6: RAVDESS Fully Connected phase 3 search space

Hyperparameter	**Type**	**Range / Values**
`n_mfcc`	Integer	30 – 60
`network_structure`	Categorical	{6, 7, 8}
`input_dropout`	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4}
`hidden_dropout`	Categorical	{0.0, 0.2, 0.4, 0.6, 0.8}
`train_epochs`	Integer	10 – 100

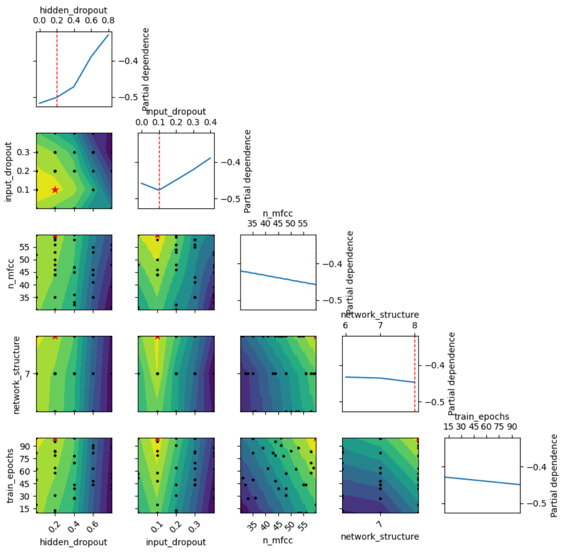


Figure 4.9: The phase 3 partial dependence plot for parameter optimisation of a Fully Connected NN system

Based on the partial dependence plot 4.9 the following final parameters were confirmed:

Table 4.7: RAVDESS Fully Connected phase 3 selected parameters

Parameter	**Value**
`hidden_dropout`	0.2
`input_dropout`	0.1
`n_mfcc`	60
`network_structure`	Configuration 8

4.3.2 Hyper-parameter Optimisation of a Recurrent-Neural-Network Based System

Phase 1 search

The process for hyper-parameter optimisation of the RNN is similar to that described in 4.3.1, however as there was more variance in the partial dependence based on network structure, the network structures available to the optimiser varies between the phases detailed below, the notation to understand the network structure tables is explaining in 4.2.2:

Table 4.8: RAVDESS RNN phase 1 network structures

Configuration Number	Hidden Layer Structure
0	(8,)
1	(32,)
2	(8, 8)
3	(8, 8, 8)
4	(16, 8, 8)

Table 4.9: RAVDESS RNN phase 1 search space

Hyperparameter	Type	Range / Values
`n_mfcc`	Integer	2 – 60
`delta_order`	Categorical	{0, 1, 2}
`n_fft`	Integer	128 – 2048
`hop_length`	Integer	256 – 1024
`n_mels`	Integer	16 – 256
`network_structure`	Categorical	{0, 1, 2, 3, 4}
`unit_type`	Categorical	{lstm}
`input_dropout`	Categorical	{0.0, 0.1, 0.2}
`hidden_dropout`	Categorical	{0.0, 0.1, 0.2}
`optimizer`	Categorical	{adam}
`batch_size`	Categorical	{16, 32, 64}
`train_epochs`	Integer	4 – 25

This optimisation was run for 100 iterations, the partial dependence plot output for this optimisation is given below in figure 4.10

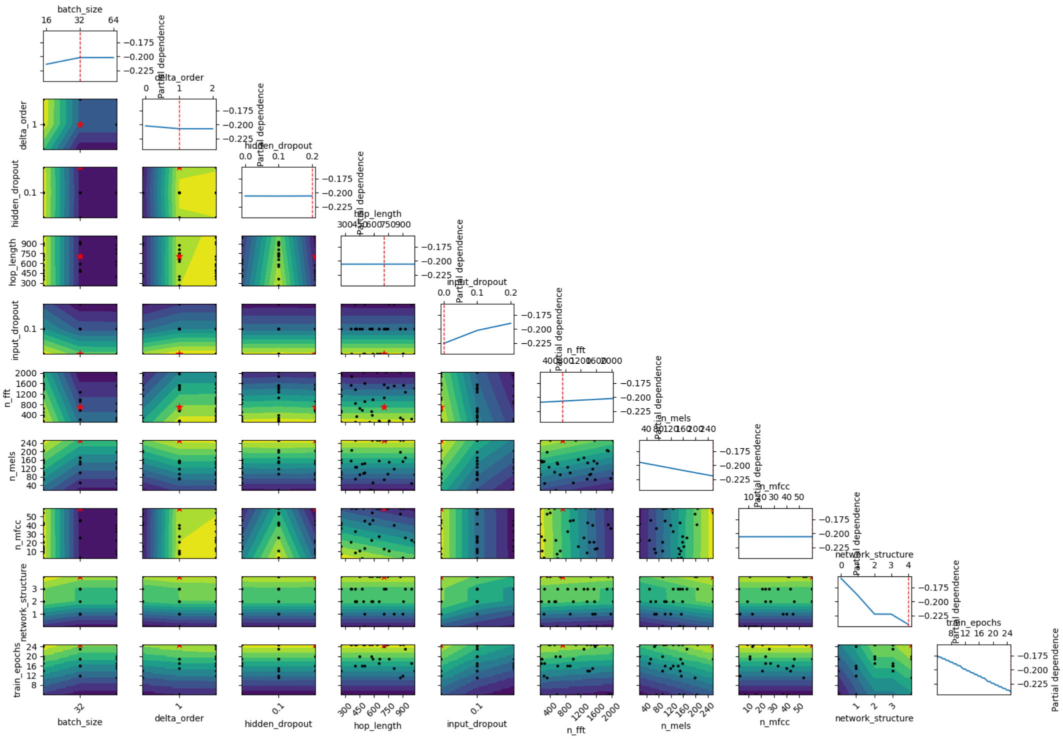


Figure 4.10: The phase 1 partial dependence plot for parameter optimisation of an RNN system

Based on the output of the partial dependence plot 4.10. the following values were fixed for the next phase of parameter exploration

Table 4.10: RAVDESS RNN phase 1 selected parameters

Parameter	**Value**
`n_fft`	1024
`n_mels`	64
`delta_order`	1
`input_dropout`	0.1
`optimizer`	`adam`
`batch_size`	64

Phase 2 search

The reduced search space and output partial dependence plot based on 100 iterations of optimisation are given below in table 4.4 and figure 4.8 respectively.

Table 4.11: RAVDESS RNN phase 2 network structures

Configuration Number	**Hidden Layer Structure**
0	(32,)
1	(64,)
2	(128,)
3	(256,)
4	(32, 16)
5	(64, 32)
6	(128, 64)

Table 4.12: RAVDESS RNN phase 2 search space

Hyperparameter	**Type**	**Range / Values**
`n_mfcc`	Integer	2 – 60
`hop_length`	Integer	256 – 1024
`network_structure`	Categorical	{0, 1, 2, 3, 4, 5, 6}
`hidden_dropout`	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4}
`train_epochs`	Integer	10 – 100

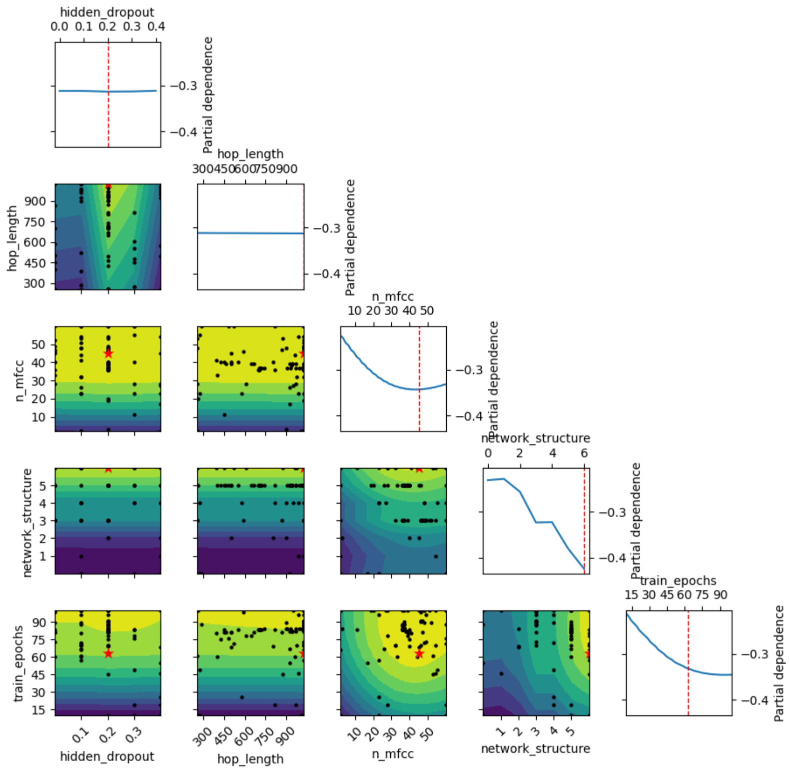


Figure 4.11: The phase 2 partial dependence plot for parameter optimisation of an RNN system

Based on the output of the partial dependence plot 4.11, the following values were fixed for the next phase of parameter exploration.

Table 4.13: RAVDESS RNN phase 2 selected parameters

Parameter	**Value**
`hop_length`	512
`n_mfcc`	40
`hidden_dropout`	0.2

Phase 3 search

For the final phase of the search a plateau of partial dependence with respect to network structure has not yet been discovered, as such all other parameters are fixed, and only additional network structure options will be explored by the optimisation system. The search space used for phase 3 is given below, it was then run for 40 iterations to produce the partial dependence plot:

Table 4.14: RAVDESS RNN phase 3 network structures

Configuration Number	**Hidden Layer Structure**
0	(128, 64)
1	(256, 128)
2	(512, 256)
3	(128, 64, 32)
4	(256, 128, 64)
5	(512, 256, 128)

Table 4.15: RAVDESS RNN phase 3 search space

Hyperparameter	**Type**	**Range / Values**
`network_structure`	Categorical	{0, 1, 2, 3, 4, 5}
`train_epochs`	Integer	10 – 100

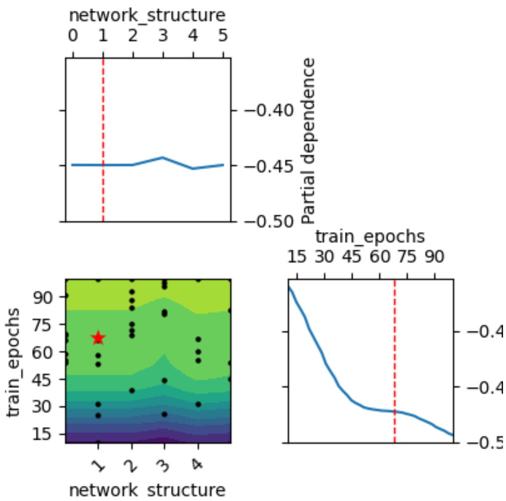


Figure 4.12: The phase 3 partial dependence plot for parameter optimisation of an RNN system

Based on the partial dependence plot 4.12 network structure 0 from table 4.14 `(128, 64)` was selected for use, as the network structure in this experiment had almost no impact on the performance of the system, therefore it is desirable to select the lowest complexity network of this performance level in order to minimize run-times, and minimize the risk of over-fitting due to excessive numbers of trainable parameters.

4.3.3 Discussion

The process for optimisation of the two networks was largely similar, the main notable differences were that the RNN system had a higher sensitivity to the network structure (and was tolerant of considerably larger numbers of trainable parameters). Additionally, the RNN was less sensitive to the MFCC construction parameters, and the final RNN configuration had a lower number of MFCCs per timestep, and included the Δ but not the $\Delta\Delta$ parameters.

These two classification systems have now been fully parameter tuned for the problem of emotion classification on this dataset.

This section provided a detailed view of how the hyper-parameter tuning was performed for 2 complete systems against the RAVDESS dataset. This methodology will also be used for hyper-parameter tuning in later chapters, however for the sake of compactness of this thesis, future descriptions of hyper-parameter tuning will give a truncated summary of the results.

4.4 Experimentation on the RAVDESS dataset

4.4.1 Further Investigation into Specific Hyper-parameters

In this section we will further explore the specific impact of a few notable hyper-parameters, and quantify precisely how the values of that parameter impact the performance of the system(s).

For these experiments we will be using 12-fold cross validation, each fold will utilise 22 (11 male and 11 female) speakers for training, and 2 (1 male and 1 female) speakers for evaluation of the objective. This means that after all folds are complete the entire dataset of 24 speakers will have been used for evaluation.

The impact of altering the number of MFCCs on a Fully Connected Network classifier

In this experiment we use the Fully Connected Neural Network optimised in 4.3.1, the only modification to the hyper-parameters is to set the input-dropout to 0.0 as this has the potential to severely impact the system at low numbers of MFCCs. The number of MFCCs generated from the raw data will be varied in order to understand the impact that this has on the performance of the system quantified in terms of the F1 score and the Accuracy (as this is a more easily understandable metric, and is appropriate to use on balanced datasets).

Discussion and Conclusions The FCNN system improves in performance as the dimensionality of the input vector is increased. This is an expected result as the MFCC is already

MFCC Count	**Accuracy$_{\text{mean}}$**	**Accuracy$_{\text{stddev}}$**	**F1$_{\text{mean}}$**	**F1$_{\text{stddev}}$**
5	0.443	0.107	0.342	0.116
10	0.508	0.072	0.466	0.068
20	0.516	0.067	0.492	0.069
30	0.531	0.075	0.503	0.073
40	0.539	0.071	0.521	0.064
50	0.542	0.067	0.533	0.065
60	**0.550**	0.071	**0.537**	0.059

Table 4.16: Results of varying MFCC vector size on emotion classification

a very compressed representation of a frame of audio, so providing additional information to the classifier is beneficial.

The impact of Δ and $\Delta\Delta$ features on both Fully Connected and Recurrent Classifiers

In this experiment we will compare the impact using the dynamic features of the MFCC vectors: the Δ - the change in MFCC values between timesteps; and the $\Delta\Delta$ - the change in the Δ value between timesteps. These features are appended onto the input vector that is passed into the classifier.

MFCC Dynamic Features	**Accuracy$_{\text{mean}}$**	**Accuracy$_{\text{stddev}}$**	**F1$_{\text{mean}}$**	**F1$_{\text{stddev}}$**
MFCC Only	0.395	0.102	0.378	0.094
MFCC + Δ	0.493	0.091	0.474	0.082
MFCC + Δ + $\Delta\Delta$	**0.550**	0.073	**0.537**	0.059

Table 4.17: Results of including dynamic features in fully connected classifier

MFCC Dynamic Features	**Accuracy$_{\text{mean}}$**	**Accuracy$_{\text{stddev}}$**	**F1$_{\text{mean}}$**	**F1$_{\text{stddev}}$**
MFCC Only	0.552	0.053	0.536	0.049
MFCC + Δ	**0.598**	0.037	**0.575**	0.041
MFCC + Δ + $\Delta\Delta$	0.594	0.041	0.571	0.050

Table 4.18: Results of including dynamic features in recurrent classifier

Discussion and Conclusions Both systems demonstrate improvement by having the dynamic features passed in. The Fully Connected based system demonstrates a much larger improvement than the RNN with the addition of these features however. This is expected as the RNN system is capable of modeling the temporal nature of the MFCC vectors internally, whereas the FCNN is not. The FCNN shows further improvement with the addition of the $\Delta\Delta$ features, whereas the RNN actually slightly degrades in performance. This is explainable by the RNN reaching saturation where the additional features are not providing useful input information, and they are consuming resources within the RNN that could be more usefully trained for other purposes.

4.4.2 Comparison of Fully-Connected and Recurrent Neural-Network Based Classifiers

For this experiment we will evaluate the fully parameter-optimised versions of the two systems experimented on throughout this chapter in order to determine the more effective methodology for classification tasks of this type.

Classifier Type	$\textbf{Accuracy}_{\textbf{mean}}$	$\textbf{Accuracy}_{\textbf{stddev}}$	$\textbf{F1}_{\textbf{mean}}$	$\textbf{F1}_{\textbf{stddev}}$
Fully Connected NN	0.550	0.073	0.537	0.059
Recurrent NN	**0.598**	0.037	**0.575**	0.041

Table 4.19: Comparing A fully connected classifier to an RNN

Discussion and Conclusions The RNN demonstrates slightly improved performance over the FCNN, and slightly lower variability across the multiple folds of test data. The implications of this will be discussed further in the chapter conclusion.

4.5 Conclusions and Discussion

In this chapter we have introduced a methodology for hyper-parameter tuning. We have applied this methodology to a static-classifier and a sequence-aware classifier for the task of classifying emotion in human speech. We have performed further experiments on these systems, and then finally compared the effectiveness of the two systems for the task of classification in this context.

The sequence-aware (RNN) system performed better than the static (FCNN) system for emotion classification, this was an expected result. However the fully connected system was able to perform very well considering its much simpler construction, and average-based final output. The performance of the FCNN demonstrates the power of the dynamic features of MFCC Δ and $\Delta\Delta$ in modelling the dynamics of human speech without a sequence classifier. However the RNN shows, by its better classification metrics, that there are some additional features to be modelled by classifying the sequence of the utterance more directly.

The systems evaluated in this Chapter have been demonstrably effective at classifying the emotional content of human spoken language. The dataset under test: RAVDESS (3.1) features emotional speech speaking the same sentences, from this we can clearly state that emotion and sentiment classification is possible completely independently of the content of the words being said. And this is therefore a viable technique to be combined with sentiment analysis of transcriptions of speech in order to provide meaningful new information.

Chapter Five

Sentiment Classification and Transfer Learning with Conventional Models

In this chapter we perform hyperparamter tuning of an LSTM model that will form a representative baseline for later works in the chapter. We conduct some additional experiments to investigate how the behaviour of the LSTM system changes in certain configurations. We then introduce a method of utilising an LSTM classifier to generate a metric for domain similarity, compare this to JS divergence as an established method and produce some results for similarity between domains that are used as a foundation for the work in later chapters.

5.1 Parameter Optimisation of Recurrent Neural Networks for Text Sentiment Analysis

In this section we will aim to use the parameter optimisation methods discussed in 4.1 to find optimal hyper-parameters for a sequence classification system based on RNN fundamentals. We will be exploring both LSTM and GRU (Chung et al., 2014) based network structures, and optimising a number of hyper-parameters of both the data preprocessing and the network itself. This process follows the same multi-step process as described in 4.3, however the presented results will be truncated to highlight the points of interest during the process. The full set of hyperparameters is given in table 5.1

Table 5.1: Parameter optimisation space for RNN models

Hyperparameter	**Type**	**Range / Values**
`vector_dim`	Categorical	{25, 50, 100, 200}
`max_sequence_length`	Categorical	{15, 28, 38, 50}
`network_structure`	Categorical	{0, 1, 2, 3, 4}
`unit_type`	Categorical	{"gru", "lstm"}
`input_dropout`	Categorical	{0.0, 0.1, 0.2}
`hidden_dropout`	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5}
`batch_size`	Categorical	{64, 128, 256}
`train_epochs`	Integer	3 – 15

The result of this some of the optimisation phases are given in figures 5.1, 5.2 in the form of partial dependence plots.

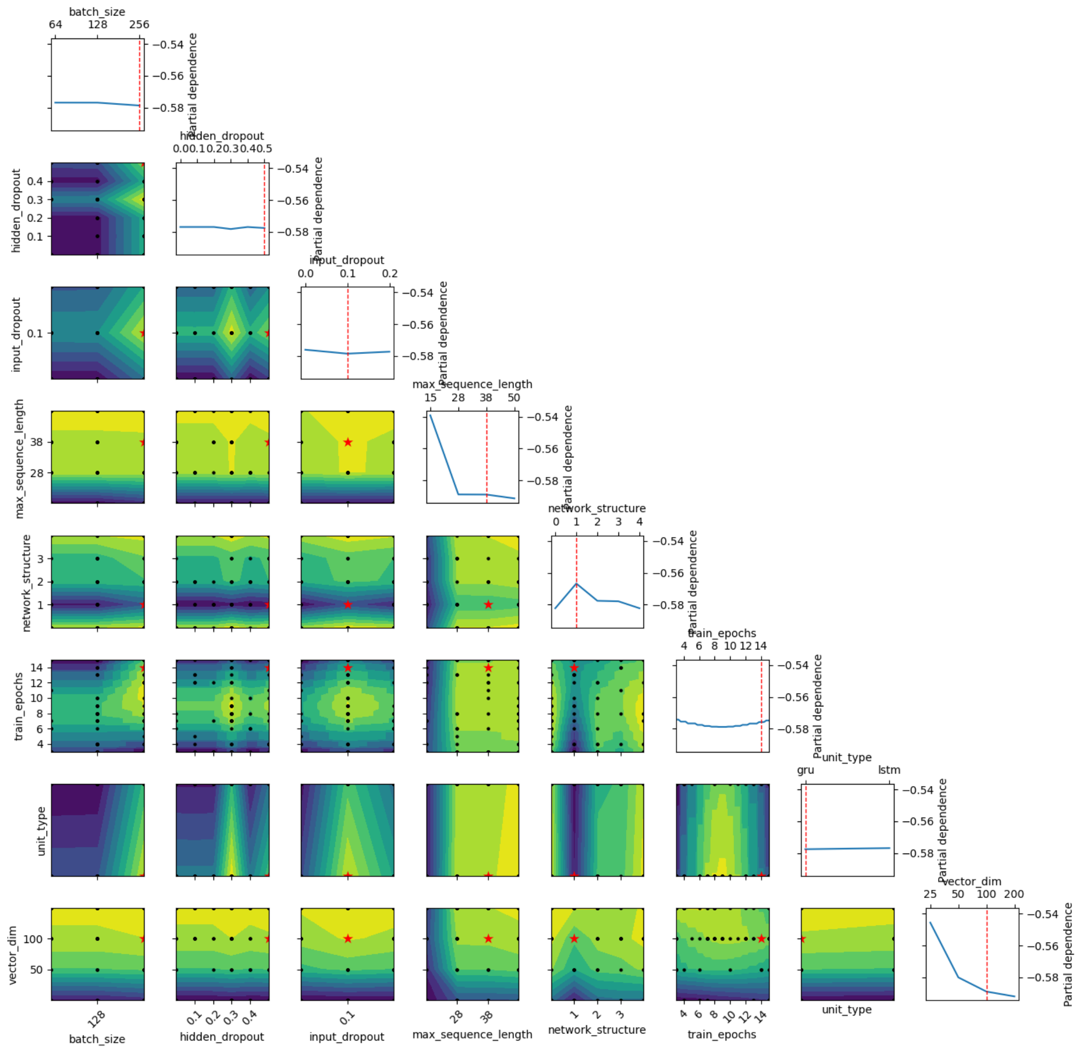


Figure 5.1: The phase 1 partial dependence plot for parameter optimisation of an RNN system

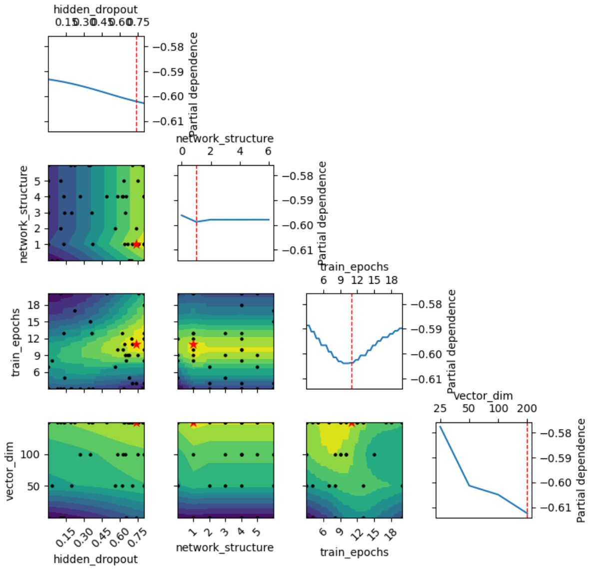


Figure 5.2: The phase 2 partial dependence plot for parameter optimisation of an RNN system

after multiple additional phases of optimisation, figure 5.2 demonstrates, by the valley shape of the `train_epochs` partial dependence curve, that the system is prone to over-fitting. Multiple additional experiments were run with alternate methods of regularisation, such as applying L1 and L2 regularisation to the hidden network layers. However none of these regularisation methods were able to completely prevent the over-fitting behaviour, and typically had a negative impact on the performance of the system. The recommendation is that in further uses of this system: A validation set should be utilised for early stopping of the training procedure when the validation loss stops decreasing.

5.1.1 Optimal Parameters

The discovered optimal parameters based on the search space are given in the table below. These optimal parameters are not used in their entirety for the experiments in the rest of this chapter, as the aim of those experiments is to determine how the performance of the LSTM changes under very constrained conditions. These optimal parameters are applied in later experiments that use this LSTM model style.

Table 5.2: SemEval RNN Optimal Hyperparameter selection

Parameter	**Value**
vector_dim	200
max_sequence_length	38
network_structure	(256, 128, 64)
unit_type	LSTM
input_dropout	0.1
hidden_dropout	0.2
batch_size	128

5.2 An Evaluation of LSTMs for Sentiment Analysis

The SE-2017-T4 data was published as part of a competition that featured almost 40 submissions, this makes this data an excellent candidate for exploration into general sentiment analysis. Not only are tweets an excellent source of real world data that feature many of the difficulties inherent in practical data analysis (such as shortened language and spelling errors), but the presence of a leader-board of example systems also provides a set of baselines against which any new system can be quickly compared.

Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber, 1997 networks are a form of Recurrent Neural Network (RNN) that have gaining increasing popularity in recent years for a variety of sequence processing tasks. Including Natural Language Processing. The advantage of an LSTM over a classical RNN is their ability to retain critical information from a "Long" distance in the past of the sequence. While an RNN can theoretically process sequences of any length, in practice the memory bus of an RNN is heavily biased towards information in the immediate past of the current point under observation. The LSTM solves this by introducing a dynamic memory as seen below in figure 5.3 that allows the each time unit to "decide" based on the current information whether information should be retained for future use or deleted.

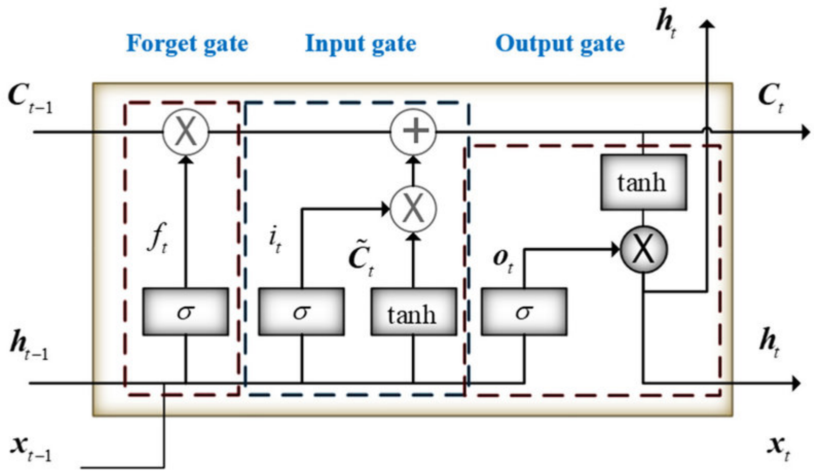


Figure 5.3: the structure of an LSTM block

Figure 5.3 shows the structure of an LSTM block. Where σ is notation for the sigmoid activation function of those NN units. tanh is the tanh activation function for those NN units. C is the long running memory bus. h is the output of the cell (h_{t-1} being the output of the previous cell). It features a unique gated structure. With 3 components:

- A "forget gate" that effectively determines which features of the long-running memory should be forgotten on a given timestep
- An "input gate" that determines which components of the current input timestep should be added to the memory bus.
- An "output gate" that generates an output from the unit based on the memory and input to the cell.

An example to understand the difference between the two network types is the following sentence "I am fluent in..." an RNN would have the capacity to know that the next word should be the name of a language. However an LSTM would be able to recall that many sentences ago the document contained "I lived in France for a year..." and therefore determine that the next word would likely be "French".

5.2.1 Managing class imbalance in RNN based systems

The SemEval dataset (3.2) Is a slightly imbalanced dataset, with a minority negative class, but a relatively balanced number of positive and neutral sentiment examples. In a number of machine learning problems, Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) is considered one of the most useful methods to synthetically augment data and re-balance classes in training data. However SMOTE can only be meaningfully applied if the data is in the form of a static, fixed-length form, such as a document vector. This technique cannot be applied to sequence classification tasks that use RNN based architectures. The typically recommended method of achieving this is to apply class-weights when training that effectively apply more priority to classes with a higher weight, and therefore allow assign more resource to defining clear decision boundaries for minority classes. For all experiments listed below. A scheme of weights inversely proportional to the representation of the class in the dataset.

$$\mathrm{w}_{\mathrm{class}} = \frac{\mathrm{count}_{\mathrm{all}}}{\mathrm{count}_{\mathrm{class}}}$$

to give the following weights: $\mathrm{w}_{\mathrm{neg}} = 6.4$, $\mathrm{w}_{\mathrm{neu}} = 2.1$, $\mathrm{w}_{\mathrm{pos}} = 2.6$

5.2.2 Testing simple LSTM capability on the data

The first task was to implement a simple LSTM system and evaluate its performance on the data, determining how it compares to the other system results presented in the leader-board. The LSTM implementation is detailed below. It should be noted that the word vector size and LSTM configuration used in these experiments does not align with the optimal parameters, as the goal of these experiments is to determine the behaviour of the LSTM in constrained conditions. But all other parameters are taken from the optimal parameters table.

First the tweets are parsed to remove abbreviations and unrecognizable characters. Then word-vectors (wv_size = 100) are generated using the Word2Vec method. The word vectors are then passed through LSTM blocks with n=54 hidden units. Finally the output is extracted from the final LSTM block and passed through 2 fully connected layers (20, 3) and soft-maxed to get a probability distribution across the 3 possible sentiment states [Positive, Neutral, Negative].

Results and Discussion Using this configuration the system was able to obtain a 66% accuracy (for a 3 class problem) on the test data, which would have made it rank #5/38 if it had been submitted at the time of the competition. This result suggests that the system performs very well for this data considering it's relatively simple design. Additionally, it is important to note that the top 10 places of the leader-board are quite contested between 61% and 69% implying that achieving performance beyond 70% may be very difficult or impossible. This could indeed be the upper ceiling if the labels for the tweets were applied by different annotators. This possibility is corroborated by Mozetic in "Multilingual Twitter Sentiment Classification: The Role of Human Annotators" Mozetič, Grčar, and Smailović, 2016 that suggests that, in 3 class sentiment classification on tweets, Human annotators have a self agreement of 0.75 (Measured by Krippendorf Alpha), and an inter-rater agreement of only 0.61. This translates to a classification accuracy of around 68% assuming balanced classes. The main source of this low agreement value is at the positive-neutral and negative-neutral boundaries. In 2 class problems [Positive, Negative] the inter-rater agreement is typically much higher.

5.2.3 Evaluating LSTM Trainable Parameters Against Performance

In order to further test the capability of LSTMs it is necessary to determine how adjustment of trainable parameters affects the performance of the system. It is important to note that the number of trainable parameters in this LSTM is governed by the following:

$$P_{trainable} = 4(nm + n^2) + [fc_1 n + fc_1 fc_2] \tag{5.1}$$

Where m is the size of the input vector, n is the number of hidden units of the LSTM, and fc_1 and fc_2 are the hidden units of the fully connected output layers. Given this, it is clear that adjusting the length of the Word2Vec word-vectors would also be desirable to evaluate the LSTM performance. The following word vector sizes were chosen wv_size = $[25, 50, 100, 200]$. Using the equation above, the following values for LSTM hidden units were chosen $n = [10, 20, 33, 45, 54]$. The choice of these values means that it is also possible to determine whether `wv_size` or `n` (the number of LSTM hidden units) has a greater effect on the performance of the system for a fixed number of raw trainable parameters, as there will be a diagonal in the results table where all systems on that line use approximately equal trainable parameters.

Results and Discussion The results of this experiment are given below in terms of classification accuracy on the tweets:

	lstm_hidden_units				
wv_size	10	20	33	45	54
25	0.60	0.61	0.61	0.60	*0.63*
50	0.61	0.63	0.63	*0.62*	0.65
100	0.62	0.64	*0.63*	0.65	**0.66**
200	0.61	*0.64*	0.63	0.64	0.65

Table 5.3: Evaluating change in performance of LSTM as a function of input and hidden unit size

From these results it can be seen that while the best results on the dataset are found with relatively high number of parameters: (wv_size $= 100, n = 54) \approx 35000$ trainable parameters, the system still maintains significantly better than average performance down as low as (wv_size $= 25, n = 10) \approx 1500$ trainable parameters. With the recorded classification accuracy of 60% being equivalent to place #14/38 in the competition for this data.

Additionally, the results marked in *italics* represent results that use an equal number of trainable parameters (≈ 17000). There doesn't appear to be a significant variation in the level of accuracy on the test data between these models. This is surprising in the lower word-vector sizes, i.e. (wv_size $= 25$) as it would be expected that the word vectors would begin to saturate at this dimension and lose a lot of important detail that the LSTM with additional units would be unable to recover. This may warrant further exploration.

5.2.4 Exploring the effect of word order on LSTM performance in short-format data

Given the extremely short nature of tweets (typically around 16 words), I hypothesised that the sentiment of the tweets may be extremely un-reliant on word ordering, as they are likely to only contain a single sentiment, and express this single sentiment with only a handful of words. This calls into question the true value of using LSTMs and other sequence models on this data type when it seems possible that a bag-of-words approach may be equally as effective.

A simple way to test this hypothesis is to randomly shuffle the words in the test and training data in order to see if the LSTM is exploiting some feature of the word order, or is in fact just remembering the sentiment critical words whenever it is passed them. This test will be done with the following LSTM configurations in order to observe if trainable parameters effect the results: $wv\_size = [25, 100]$, n(LSTM hidden units) $= [20, 45]$. Additionally, this experiment will be conducted in two parts: *shuffled test data only* in which the system will be trained on ordered data and evaluated on shuffled test data. And then *shuffled train and test data* in which both sets of data are randomly shuffled for word order. The purpose of this second experiment is to determine if the LSTM is being unfairly hindered in the first test by being given order to attempt to utilise and then having it removed.

Results and Discussion The results for these experiments are given below as well as their rank if they were submitted to the competition.

Experiment	Network_Structure	Acc	F1	Rank
Random_Test_Only	(wv_size $= 25, n = 20$)	0.60	0.55	#18
	(wv_size $= 100, n = 45$)	0.59	0.54	#19
Random_Train_Test	(wv_size $= 25, n = 20$)	0.58	0.54	#20
	(wv_size $= 100, n = 45$	**0.61**	**0.57**	#16

Table 5.4: Ranking of LSTM system against competitive systems

From these results it can be seen that while the decrease in performance from this shuffled approach is only around 4.5% (65% to 61%), this is however, a significant decrease w.r.t the position in the competition, which suggests that results in the 65-70% range may be "hard" results, which the results in the 60-65% range may actually be fairly easy to achieve. This could be a result of the difficulty of classification at the positive-neutral and negative-neutral boundary, where this bag-of-words approach loses the majority of it's performance. This suggests that while the tweets are of a short format, there is some intrinsic importance in the order of the words for sentiment classification. This is likely to be due to negations: "good" vs "not good", and other words such as "too" i.e. "this product is too big".

5.3 Defining a measure of similarity between domains

The primary focus of this thesis is to demonstrate techniques that allow for robust transfer learning of sentiment analysis between domains that have low similarity with each-other. In order to have said to have achieved that goal. It is first necessary to define domain similarity. In this section we will apply Jensen-Shannon divergence in order to measure the difference between domains. We then propose an alternate methodology using classifier mis-classification to measure domain similarity and discuss the differences between the two methods.

5.3.1 Applying Jensen-Shannon Divergence to Measure Domain Similarity

Jensen-Shannon Divergence (Menéndez et al., 1997) is used to measure the divergence between two distributions of data. In order to apply this technique to two domains of uncorrelated documents. We must first convert these domains of documents into comparable distributions.

Representing collections of documents as TF-IDF Distributions In order to compare two domains of documents. We first join both collections of documents and "train" a TF-IDF system (the document collections must be joined so that the TF-IDF representation of both domains are comparable). Each document in the domain then has its TF-IDF representation calculated. A domain-wise mean of the TF-IDF is calculated. Finally the domain TF-IDF is normalised such that the vector sums to 1. This vector can then be thought of as a probability distribution that represents the domain of documents. And this allows us to compare domain distributions to each other.

Applying JS Divergence Jensen-Shannon Divergence is an extension on the Kullback-Leibler Divergence method. The main notable difference in practice is that JS divergence is a symmetrical metrics (i.e. the JS Divergence of (A, B) is the same as that of (B, A)). While an asymmetric-metric may be useful in practice for the intent of this thesis (i.e. training on A to transfer learn to B may have different results than training on B to transfer learn to A); we cannot guarantee that the output of KL Divergence will correlate to this requirement. Therefore it is more practical at this time to focus on a symmetric metric.

The JS Divergence of two TF-IDF distributions will be calculated using the following (as an additional practical note, 0 values in the vectors are given a tiny ϵ value in order to prevent division by 0 errors.):

$$\mathrm{JS}(P \parallel Q) = \frac{1}{2}\left(D_{\mathrm{KL}}(P \parallel M) + D_{\mathrm{KL}}(Q \parallel M)\right) \tag{5.2}$$

$$\text{where } M = \frac{1}{2}(P + Q) \tag{5.3}$$

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{5.4}$$

JS Divergence on Amazon Review Dataset The result of applying the JS Divergence to the Amazon Review dataset is given below in Figure 5.4

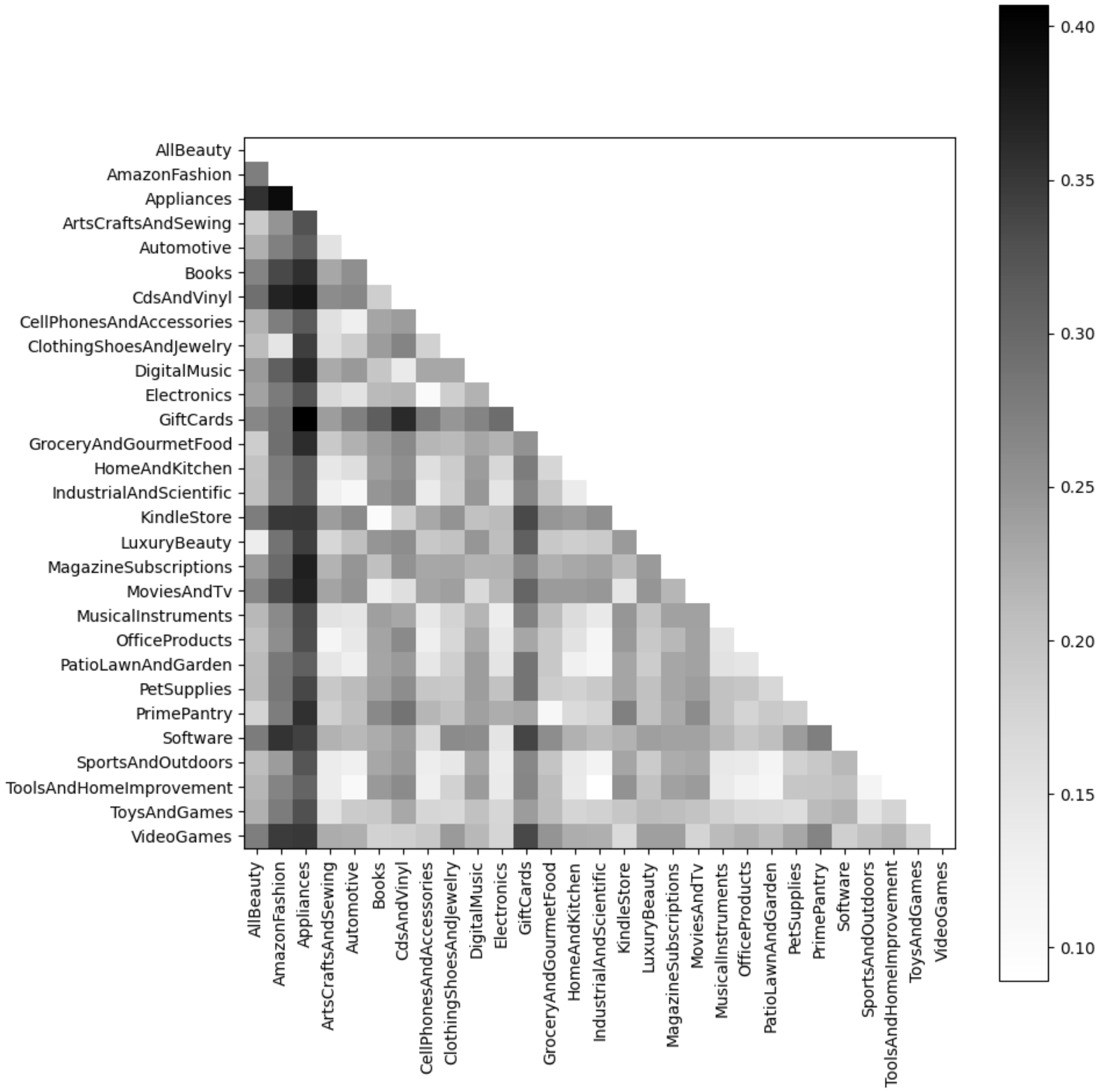


Figure 5.4: The result of JS Divergence on the Amazon Review Dataset

This result will be discussed in context in 5.3.4

5.3.2 Alternate Methodology

In order to define domain similarity, we can exploit a typically undesirable feature of classification systems: Mis-classification. A classifier neural network was created with a reduced number of trainable parameters (see implementation details below). This prevents the classifier from being able to fully model the specifics of each domain of target data, and instead to learn the general high-level features that are most salient for discriminating between domains. By using this method we capture the properties of each domain that makes it unique to other domains in the set.

Next we attempt to classify a test set of domain specific data. As with all classifier systems, some items of the test set are incorrectly classified as a different domain. Where this happens frequently, we consider these two domains to be "similar". Where this happens rarely, we consider these two domains to be "dissimilar".

In order to make this system a fairer test with limited data availability, and to allow it to be more easily extendable to new datasets. We run the test on individual pairs of domains one at a time, rather than running the test on a large group of several domains. This creates more reliable results, as the classifier is always training to find separable features between the two domains under test and therefore provides a better metric of similarity. In contrast, if the test had been run on many domains at once. The classifier may be forced to group features from several domains in order to get a better overall score, but this does not necessarily mean those domains would be fair to consider "similar".

It is also important to note that while this methodology is not guaranteed to produce a symmetric result. For example, it might be the case that test items from 'Domain B' are more likely to classify as 'Domain A' than test items from 'Domain A' classifying as 'Domain B'. However the goal of this experimentation was to establish a "similarity" score between domains that should be considered to be symmetric, and fully exploring the implications of asymmetric similarity between domains is outside of the scope of this research, but could be an avenue for further research in the future if it is considered of scientific interest. As such, the average of the two values was taken as the similarity score that these results are based on. (Additionally in testing it was noted that there was typically a strong symmetry between mis-classification).

5.3.3 Implementation and Experiment Design

The system used for this experiment is similar to the LSTM design discussed in 5.2. It uses a word-vector processed input of dimension 50, and a 2 layer LSTM with output sizes of [10, 5] respectively. Followed by a fully connected single classification layer of output size 2.

Layer number	**Type**	**Details**
0	Vectorisation	Word2Vec wv_size=50
1	LSTM	input = 50; output = 10
2	LSTM	input = 10; output = 5
3	Fully Connected	input = 5; output = 2;

Table 5.5: Network parameters used to define inter-domain distances

Results of Domain Similarity by Mis-classification The "matrix of similarity" created by this system is given below in Figure 5.5

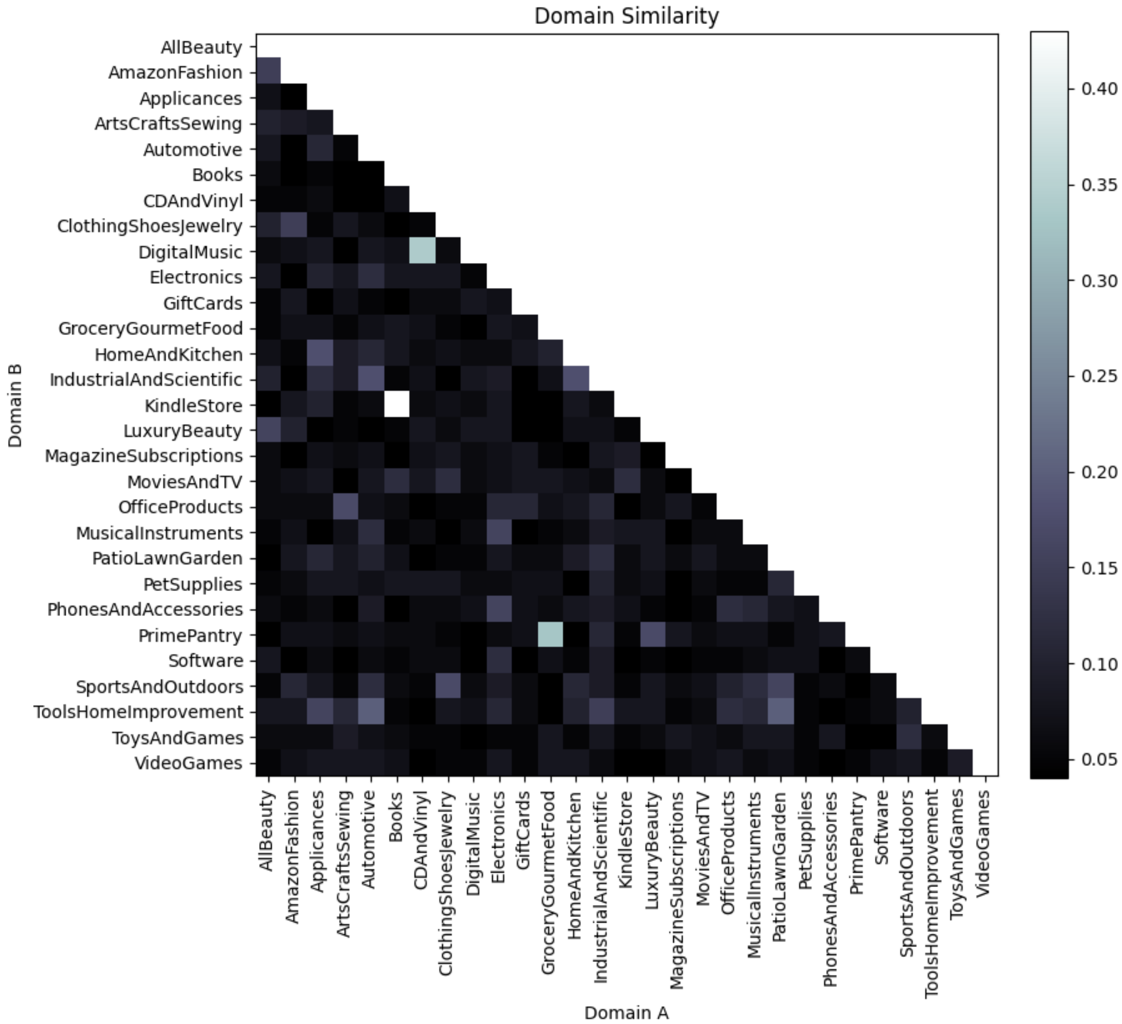


Figure 5.5: The "matrix of similarity" between domains in the Amazon review

Figure 5.5 shows a matrix of the mis-classification rate between pairs of domains. This metric is calculated by using $(\text{False}_\text{A} + \text{False}_\text{B}) \div (\text{False}_\text{A} + \text{False}_\text{B} + \text{True}_\text{A} + \text{True}_\text{B}) = (1 - \text{accuracy})$. The hypothesis proved by this figure is that the more similar the domains are, the less separable they will be, and therefore the higher their mis-classification score will be.

5.3.4 Discussion

Comparing domain mis-classification to JS Divergence

As discussed below, both the JS Divergence and the Mis-classification Method were able to identify pairs of expected similar domains, such as [Books, KindleStore]. However the JS Divergence provides a much less clear result with a large amount of noise that makes identifying similar and dissimilar domains using this matrix very challenging in practice. Additionally, the domains: AmazonFashion, Applicances, and GiftCards demonstrate a high divergence against almost all other domains, this is likely due to the low volumes of data available for those domains. This result implies that JS Divergence is not a good choice for comparing domains with low, or highly-variable volumes of data. In contrast, the mis-classification technique provides a very low noise result that clearly highlights domains in the set that are similar and those that are clearly separable.

This result is expected when considering that an ML classifier is fundamentally a tool for internally modelling the distribution of data. This means that the ML model is able to create a much more sophisticated representation of the two entire domains of data from learning on the individual document examples. The mis-classification metric is then a measurement of how much these distributions overlap. The high clarity of the result is due to the training intent of the model being to discover the optimal hyperplane for separating these distributions, so this metric is effectively correlated to the overlap in the most-separable hyperplane. Rather than a generic overlap of the distributions that is measured by standard statistical methods such as JS Divergence.

Analysis of similarity result

From the figure, we can a number of expected "highly similar" domain pairs, such as:

- 'KindleStore' (The Kindle is an Amazon e-reader device that allows for reading of eBooks) and "Books"
- 'CDAndViynl' and 'DigitalMusic'

As well as expected "somewhat similar" domain pairs such as:

- 'Automotive' and 'ToolsHomeImprovement'
- 'ArtsCraftsSewing' and 'OfficeProducts'

This result will be used as the basis when referring to domain level similarity with experiments that involve the Amazon Product Review dataset.

This experiment was also applied to Twitter and Motivational Interview datasets, to discover its similarity to the domains in the Amazon Review dataset. Multiple review domains were included, which allow the Twitter dataset to be base-lined in reference to the figure above. However the full range of product domains was not included in this result for readability.

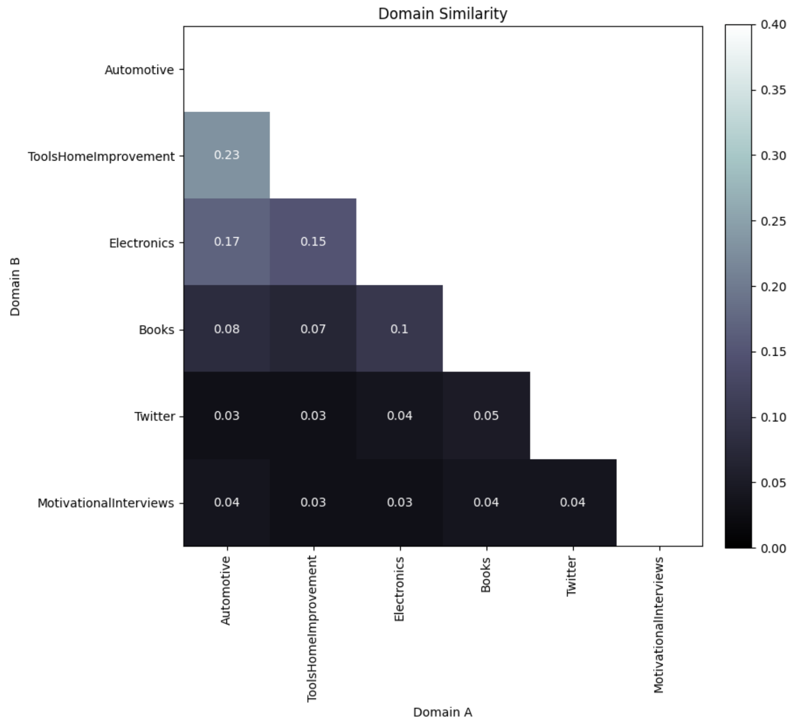


Figure 5.6: The "matrix of similarity" between between Twitter, MI Data, and select Amazon Reviews

As expected, the Twitter, and Motivational Interview datasets had a consistently lower similarity score with any domain of Amazon Review data, than those reviews did with each-other.

It is also critical to note that the Twitter and Motivational Interview datasets have **comparable** levels of similarity to the domains of Amazon Review data. This allows us to consider the Twitter data to be a strong analogue for "Highly dissimilar data" in further experiments that require significant quantities of labelled data. And allows reasonable confidence that the conclusions gathered from those experiments will also be applicable to the Motivational Interview data.

Chapter Six

Adversarial Learning for Domain Independent Sentiment Analysis

6.1 Introduction

This chapter aims to address Research Questions 3 and 4. By utilising a novel method of transfer learning, in order to either improve on the baseline results for "similar domain transfer learning" and "dissimilar domain transfer learning". Or demonstrate a lower rate of degradation of the model's capacity as the similarity of domain decreases (this degradation of performance can be seen in 6.6.2).

This work aims to extend upon the HAGAN system (Zhang, Miao, and Jiaqi Wang, 2019) Discussed in chapter 2. By both improving upon the design and implementation details of the system. As well as evaluating the improved system's performance across a range of transfer learning domain similarities.

6.2 Justification of the Use of Adversarial Learning for this Problem

As discussed earlier, in a typical transfer learning system it is common to use the main collection of out-of-domain training data in order to optimize some earlier layers, or feature processing section of the model, and then subsequently retrain later sections of the model in order to refine the performance on in-domain data. By utilising Adversarial Learning for this problem, we invert the process. Fine tuning a classification model on out-of-domain data, thereby creating a map from input-space to feature-space to classification-space. The system then attempts to map the in-domain data into the calculated feature-space without modifying the mapping from feature-space to classification-space.

This inversion of the process creates a critical unique difference from traditional methods: Because the in-domain data is not used to build the feature-space to classification-space mapping, it is possible to use this process to transfer learn classification onto completely unlabelled data which could yield an extremely potent advantage for this system over traditional methods in such applications where there is no labelled in-domain data.

Furthermore, Traditional transfer learning methods use the out-of-domain data in order to build feature learning layers, and then those layers are frozen regardless of the source that the in-domain data comes from. Therefore we can state that these methods rely on un-intelligently exploiting the similarities between the out-of-domain and in-domain data. For instance, by learning common patterns in a shared language. However, the Adversarial Learning approach attempts to dynamically map the in-domain and out-of-domain data into a shared feature space, and therefore can be considered to be intelligently exploiting the similarities between the two sets.

6.3 Abstract Explanation of System Function

As in the majority of adversarial learning applications. This system is separated into two distinct agents: The "Generator" (in this case it would be more accurate to refer to this agent as the "Embedder" as no data is generated.) And The "Discriminator" (or "Critic"). However unlike classic GAN tasks. The primary goal of the discriminator is not to distinguish between "real" and "generated" data. But to distinguish between "in-domain" and "out-of-domain" embeddings. The Discriminator agent also has the additional task of performing sentiment classification on the embedded data.

As the training progresses, the "Embedder" agent creates document embeddings of both in-domain and out-of-domain data. The "Discriminator" agent performs sentiment classification on the sentiment-labelled out-of-domain data, while simultaneously attempting to separate the in-domain from the out-of-domain data. The loss from both of these signals is then fed back to the "Embedder" agent.

The desired result from this training procedure is three-fold: First, the "Embedder" agent should be learning to generate document embeddings that are optimized for sentiment classification of the out-of-domain training data; Second, The "Discriminator" (Classifier) agent should be learning to identify an optimal hyperplane for classifying sentiment with respect to this embedding space; Third and finally, the "Embedder" should be learning to exploit similarities in order to map the in-domain data into the same feature space as the out-of-domain training data.

Unlike typical GAN and similar systems, there is an expectation that the mapping between the input space and the embedding space is coherent. The use of the additional classification task causes an anchoring effect that encourages this behaviour.

It should be noted however that there is no hard anchoring that forces the un-labelled in-domain data from being mapped into the space in a coherent manor. Or formally, there is no gradient that prevents positive sentiment examples from the in-domain set from being mapped onto negative sentiment examples from the out-of-domain dataset. However it is assumed that a combination of the similarity of the input features (i.e. both English language text data) and a careful tuning of the complexity available to the embedder network will make this undesirable behaviour unlikely.

Additionally, it should be possible to mitigate this undesirable behaviour using a small amount of labelled examples of the in-domain training set. This possibility will also be explored further in this chapter in order to demonstrate how this system can be used to very effectively leverage minimal amounts of labelled training data.

6.4 Implementation Specifics

Similar to the Diagram below from Zhang, Miao, and Jiaqi Wang, 2019, the system is implemented as such: The documents are parsed and converted to word vectors using the Word2Vec system; These vectors are then passed through the Generator Network in order to produce a document-embedding for each document; Finally the document vectors are passed through the fully connected "Discriminator" Network in order to be classified for both **Sentiment** and **Domain**.

At training time, the generator is provided documents from both the *Source* and *Target* domains to embed into the document-embedding space. These embeddings are then passed to the Discriminator for classification. The *Source* data is classified for both Sentiment and Domain, while the *Target* data (that is not sentiment-labelled) is classified only for Domain.

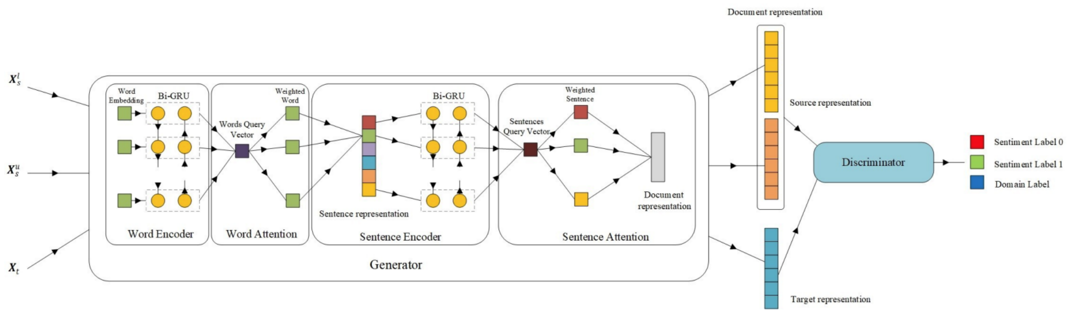


Figure 6.1: The System diagram for the HAGAN system

The Discriminator Network feeds back and improves its ability at both classification tasks. While the Generator network receives feedback for its *Source* Sentiment "classifiability", and its *Target* Domain **"inclassifiability"** so that it is able to improve both the **sentiment-quality**, and the **"domain-indistinguishability"** of its embedding system. This training was run for up to 200 epochs for these experiments; as with GAN systems there is no way of setting a loss threshold to stop at, as the losses are conditional on each other.

There were a number of challenges to overcome with implementing this using existing frameworks.

The TensorFlow framework Martín Abadi et al., 2015 has some difficulties implementing the GAN architecture as it requires the discriminator network to be calculated but not trained when training the generator section of the network. And then the discriminator must still be trainable during the same iteration. While this is somewhat complex to achieve, it is a relatively common goal and thus many resources are available to explain the steps needed to implement this form of network.

However, this particular system presents an additional difficulty, in that the loss function for the generator and discriminator are representing the combined loss for two separate tasks: The domain classification task, and the sentiment classification task. Therefore the loss must be calculated manually using `tf_backend` tools, and then the gradients must be applied to the network. This requires rewriting a number of the built in methods that are called during the training process so that they are able to accept multiple loss functions and combine them in the desired manner.

Additionally, in this system unlike the original HAGAN system, Word2Vec word-vectors (d=200) are used instead of BERT word-vectors (d=768), this was a decision made in order to complete experimentation in a reasonable time, as the BERT vectors take orders of magnitude longer than W2V to calculate at run-time, and they could not be stored ahead of time due to memory restrictions.

When performing experimentation on this system, a subset of the MDSD2 dataset will be used to ensure that it is functioning as expected and evaluate the results compared to the original HAGAN system Zhang, Miao, and Jiaqi Wang, 2019, then this dataset will be combined into a single, wider domain considered as "Amazon Reviews". This dataset will then be cross-evaluated with the SE-2017-T4 dataset and the LMRD1 Dataset.

6.4.1 Data usage strategy

For all experiments detailed in this chapters, the following experimental methods were applied:

- Each result was repeated 3 times in order to generate a mean result and a standard deviation of that results.
- The training datasets were split into 90% train and 10% validation (for early stopping modes where appropriate)

- no cross-fold style validation was used unless explicitly stated, as the datasets used for training and the dataset used for testing for the majority of these experiments are different domains. And therefore the concept of cross-fold validation is not appropriate.

6.5 Hyperparameter Tuning of the Networks

This system uses a GAN style training procedure, but the classification at inference time is performed by classifier networks similar to those seen in earlier sections of this Thesis. When performing hyper-parameter optimisation for these systems, we will limit this tuning to finding optimal hyper-parameters for the classifier networks on the task of sentiment classification of the Amazon Review (3.3) dataset. This is done primarily as the intent of the system is to demonstrate the ability to transfer learn against multiple datasets, including datasets without their own sentiment labels. As such it might corrupt the results if the hyperparameters are tuned against a specific target dataset.

This Chapter compares the effectiveness of two neural network classifier methodologies, which will be compared in more detail in 6.7: A CNN based system and a HAN based system. In this section we will use the methods described in 4.1 and used elsewhere in this Thesis to identify optimal hyper-parameters for the two classifiers.

6.5.1 Hyperparameter optimisation of the HAN network

The structure of the HAN network is marginally different to other networks seen in this Thesis thus far. It segments a document into sentences, and then segments those sentences into words. As such there are always two layers to the GRU network, and those parameters will be listed using the subtext $_{\text{words}}$ or $_{\text{sentences}}$ respectively. The evaluated search space for the system is given in the table below.

Table 6.1: Parameter optimisation space for HAN model

Hyperparameter	**Type**	**Range / Values**
$\texttt{max\_length}_{\texttt{words}}$	Categorical	{10, 20, 40, 60, 80, 100}
$\texttt{GRU\_units}_{\texttt{words}}$	Categorical	{25, 50, 100, 200}
$\texttt{GRU\_dropout}_{\texttt{words}}$	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4}
$\texttt{max\_length}_{\texttt{sentences}}$	Categorical	{10, 25, 50, 100}
$\texttt{GRU\_units}_{\texttt{sentences}}$	Categorical	{25, 50, 100, 200}
$\texttt{GRU\_dropout}_{\texttt{sentences}}$	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4}
`batch_size`	Categorical	{8, 16, 32, 64}

The optimal parameters discovered using Bayesian optimisation techniques are as follows:

Table 6.2: Discovered optimal parameters for HAN classifier network

Hyperparameter	**Value**
$\texttt{max\_length}_{\texttt{words}}$	80
$\texttt{GRU\_units}_{\texttt{words}}$	100
$\texttt{GRU\_dropout}_{\texttt{words}}$	0.2
$\texttt{max\_length}_{\texttt{sentences}}$	50
$\texttt{GRU\_units}_{\texttt{sentences}}$	100
$\texttt{GRU\_dropout}_{\texttt{sentences}}$	0.2
`batch_size`	32

6.5.2 Hyperparameter optimisation of the CNN network

The CNN network used in this chapter also features some parameters that have not been encountered in earlier sections of this thesis. it includes a number of parameters for the filter configuration (both the kernel-size of each filter in the CNN layer, and the specific configuration of the filters themselves). Additionally, it has a multi-layer fully-connected network to feed the final output layer, and this network configuration is also tune-able.

Regarding the filters, a `filter_sizes` parameter of (2, 3, 5, 7) means that there are 4 types of filter. that span the listed number of words, in this example there would be a 2-gram, 3-gram, 5-gram, and 7-gram filter that perform encoding on their respective spans of words to generate phrase information. The explored parameter space for the CNN classifier model, and the network configuration options are given below:

Table 6.3: Parameter optimisation space for HAN model

Hyperparameter	**Type**	**Range / Values**
`CNN_filter_configuration`	Categorical	{0, 1, 2, 3, 4, 5, 6}
`CNN_kernel_size`	Categorical	{10, 25, 50, 100}
`CNN_dropout`	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4}
`Dense_configuration`	Categorical	{0, 1, 2, 3, 4, 5, 6}
`Dense_dropout`	Categorical	{0.0, 0.1, 0.2, 0.3, 0.4}
`Batch_size`	Categorical	{8, 16, 32, 64}

Table 6.4: CNN Filter configuration options

Configuration Number	**Configuration**
0	(2, 3)
1	(2, 3, 4, 5)
2	(3, 5, 7)
3	(3, 5, 7, 9)
4	(3, 5, 7, 9, 11)
5	(3, 5, 7, 9, 11, 13)
6	(3, 5, 7, 9, 11, 13, 15)

Table 6.5: CNN dense layer configuration options

Configuration Number	**Configuration**
0	(32,)
1	(32, 16)
2	(64, 32)
3	(128, 64, 32)
4	(256, 128)
5	(256, 128, 64)
6	(512, 256)

The results of the optimisation process are given below:

Table 6.6: Discovered optimal parameters for CNN classifier network

Hyperparameter	**Value**
`CNN_filter_configuration`	(3, 5, 7, 9, 11, 13, 15)
`CNN_kernel_size`	50
`CNN_dropout`	0.2
`Dense_configuration`	(256, 128)
`Dense_dropout`	0.2
`Batch_size`	32

6.6 Performance Degradation with Decreasing Domain Similarity

6.6.1 Experimental Setup

In order to justify the work undertaken in this chapter, it is first necessary to demonstrate that the performance of a classifier system shows degradation as the similarity between the train domain and the test domain decreases.

For this experiment we will be using the CNN system with the parameters optimised in 6.5. Based on the domain similarity scores output in 5.6, we will utilise the Amazon "Automotive" review domain as our training domain, with the following domains being used in order of decreasing similarity: [Automotive, Amazon-ToolsHomeImprovement, Amazon-Electronics, Amazon-Books, Tweets (SemEval)]. In order to simplify the metrics, all training and test data will be an artificially balanced 2-classes of "positive" and "negative" sentiment, the classification accuracy is then a useful metric to measure to demonstrate the degradation of classification performance. This experiment will be run 3 times, and a mean accuracy and std-dev will be reported.

6.6.2 Results and Discussion

Table 6.7: Accuracy Statistics for Test Domains

Test Domain	**Accuracy$_{\text{mean}}$**	**Accuracy$_{\text{stddev}}$**
Automotive	0.87	0.021
ToolsHomeImprovement	0.81	0.017
Electronics	0.77	0.024
Books	0.75	0.027
Tweets	0.68	0.043

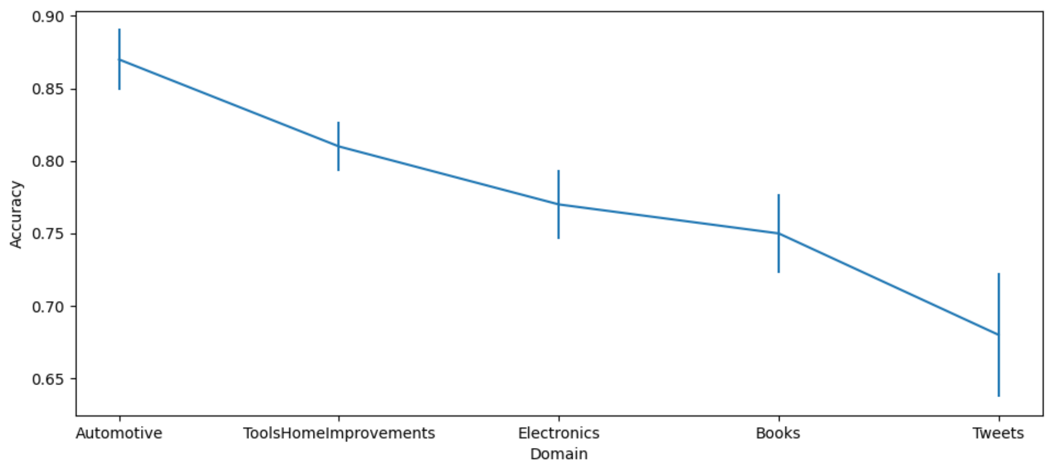


Figure 6.2: the performance degradation of sentiment classification with domain distance

From the results presented above, we have demonstrated the performance degradation of classification as we increase the inter-domain separation. We have also validated the methodology presented in 5.3, by showing that the performance of sentiment classification degrades in agreement with the order of domain similarity.

6.7 Further Optimization of Network Structure

The HAGAN paper demonstrates the use of a Hierarchical Attention Network (HAN) as the generator (embedding) network to transform the sequence of word vectors at the input into a fixed length document vector suitable for classification. However in order to justify this choice it is necessary to explore at least one other architecture capable of performing variable-to-fixed-length transformation. In this case a Convolutional Neural Network (CNN) of a specific structure can be used.

Hierarchical Attention Network (HAN) The HAN Yang et al., 2016 can be understood by breaking it down into it's two components: The "Hierarchical" component of the network uses two layers of Bi-directional Gated Recurrent Units (GRU) Cho et al., 2014, the first to convert the words of a sentence into "sentence vectors"; and the second layer to convert this sequence of sentence vectors into a document vector. The second component of the system, the "Attention" component means that the system uses a self-attention mechanism in both of the layers, which gives the system a mechanism by which it can "decide" that some words or sentences may be more important that others for performing it's classification. For a better understanding of the structure of the HAN system in this context refer back to figure 6.1 to see the implementation from the HAGAN paper.

Convolutional Neural Network (CNN) This structure of network (first proposed in Kim, 2014) is often used for sequence-to-fixed-length embeddings in natural language processing applications. The network can be designed such that it uses an "n-gram" style approach, where many (in my implementation n=50) filters would be passed over all the groups of 3, 5, 7 ... 15 words, these filters can then be max-pooled (over the time dimension) such that each filter contributes a single value to the overall document vector, thereby creating a fixed length output vector.

Put simply the system can use each one of its available $m * n$ filters to "search" for one specific feature in the document, by ensuring that this feature sums to a large value. A simple example might be "happy tri-grams": A 3-gram filter is passed over the whole document, calculating the happiness value of each of the tri-grams in the document, the maximum value for "happy tri-grams" is then passed to position x of the final document vector. (From this explanation it might appear that average-pooling the vectors i.e. "the average happiness of the tri-grams in the document" would be more appropriate however in practice max-pooling tends to yield better performance). refer to figure 6.3 for more information on the structure of this network

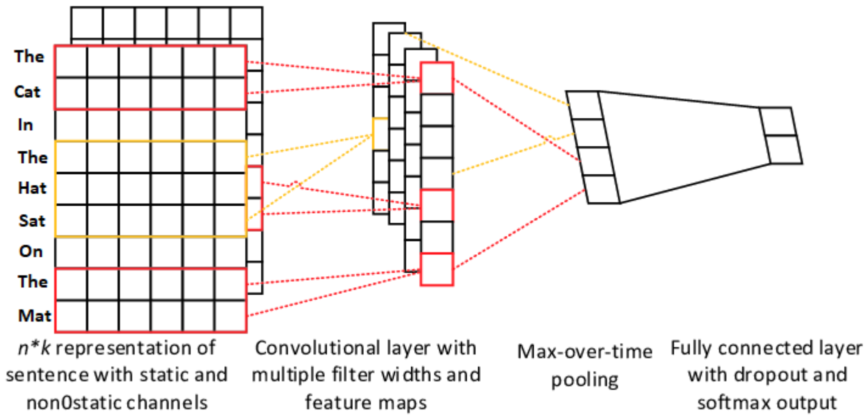


Figure 6.3: CNN for text classification

Results and Discussion The results of this experiment are as follows:

		HAN		CNN	
source_db	target_db	mean_acc	std_dev	mean_acc	std_dev
MDSD2_books	MDSD2_dvds	0.75	0.012	**0.76**	0.007
MDSD2_books	MDSD2_electronics	**0.76**	0.010	0.76	0.006
MDSD2_books	MDSD2_homeware	0.75	0.013	**0.77**	0.005
MDSD2_combined	SE-2017-T4	0.71	0.009	**0.73**	0.009
MDSD2_combined	LMRD1	**0.80**	0.010	0.78	0.007
SE-2017-T4	MDSD2_combined	**0.69**	0.018	0.67	0.006
SE-2017-T4	LMRD1	0.67	0.019	**0.70**	0.009
LMRD1	MDSD2_combined	**0.82**	0.009	0.77	0.008
LMRD1	SE-2017-T4	0.74	0.011	**0.78**	0.003

Table 6.8: Evaluating the performance of the HAN vs CNN based system against difference domains of data

While these results demonstrate that both systems seem to have very similar performance on the datasets tested, the CNN demonstrated more consistency in its classification accuracy. Additionally, when observing the accuracy on validation data during training, it was noted that the CNN system consistently optimized in fewer training epochs than the HAN system. Therefore the CNN system will be used for future experimentation.

Additionally, it can be seen that the performance in this experiment is marginally lower than that of the HAGAN system in literature. This is attributed to the use of Word2Vec word-vectors (d=200) rather than BERT word-vectors (d=768).

6.8 Further Optimising the Loss Functions for The System

The HAGAN loss functions The paper Zhang, Miao, and Jiaqi Wang, 2019 that suggested using GANs for this application proposes the following loss function, broken down into 4 parts detailed below. However it is necessary to validate that this loss function is appropriate and to experimentally determine how it compares to both baselines and normal best-practices for this type of problem.

The Sentiment loss (6.1) which refers to the loss generated by the inaccuracy between predicted sentiment and true sentiment. **This loss is only calculated for the "source" data set (the data with sentiment labels)**. However in experiments detailed below this loss may be used for labelled items from the "target" data set.

$$L_{sen} = -\frac{1}{N_s}\sum_{i=1}^{N_s} y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i) \tag{6.1}$$

The Domain loss (6.2) generates its loss based on the difference between the predicted and true labels for the domains of both the "source", and the "target" data sets. This loss is used to train the **Discriminator** only.

$$L_{dom} = -\frac{1}{N_s + N_t}\sum_{i=1}^{N_s+N_t} d_i \ln \hat{d}_i + (1 - d_i)\ln(1 - \hat{d}_i) \tag{6.2}$$

The "Masked" Domain loss (6.3) its based on the difference in prediction between the domain labels for the **"target"** set only, and "masked" labels that are set to the label value of the "source" set. This allows this loss component to measure how effectively the Generator is "confusing" the Discriminator into incorrectly predicting the wrong domain labels for the "target" set. This loss is used to train the **Generator** only. Where $\tilde{d}_i$ is the "masked" label

$$L^c_{dom} = -\frac{1}{N_t}\sum_{i=1}^{N_t} d_i \ln \tilde{d}_i + (1 - d_i)\ln(1 - \tilde{d}_i) \tag{6.3}$$

Finally, the "Entropy Loss" (6.4, which the authors claim helps to maximize the margins between the target domain data and decision boundaries thereby increasing prediction confidence.

$$L_{ent} = -\frac{1}{N_t}\sum_{i=1}^{N_t}\sum_{j=1}^{C} p_{ij}^{sen} \ln p_{ij}^{sen} \tag{6.4}$$

These losses are then combined in the following ways for the Discriminator and Generator respectively: where λ_D, λ_G^1, and λ_G^2 are controllable parameters that determine the blending weights of the loss components.

$$L_D = L_{sen} + \lambda_D L_{dom} \tag{6.5}$$

$$L_G = L_{sen} + \lambda_G^1 L_{dom}^c + \lambda_G^2 L_{ent} \tag{6.6}$$

MSE loss In order to serve as a baseline to compare the other results to, I felt it was necessary to include the mean-squared-error (MSE) loss (6.7), for both the domain-classification task and the sentiment-classification task.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{6.7}$$

Where Y_i and $\hat{Y}_i$ are the predicted and true labels for each data point. While MSE is not always a good choice for classification and is generally more suited to regression problems, it is acceptable in this case as the domain-classification task can be thought of as a regression of the similarity between the two domain embeddings. However MSE loss may prove unsuitable for use on the sentiment-classification task

Cross-entropy losses For this test I utilised tensorflow's built in cross-entropy loss functions: For the domain classification task the `binary_crossentropy` loss was used; and for the sentiment classification task, the `categorical_crossentropy` loss was used. Explaining the exact functionality of these losses is outside the scope of this paper however. But the critical thing to note is that `binary_crossentropy` is suitable for classification tasks where a single output will be either 0 or 1 for the two possible options; and `categorical_crossentropy` is used for classification task where multiple categories will be expressed using a one-hot (softmax) output.

Wasserstein GAN loss Finally, I implemented the WGAN loss discussed in section 2.5, in order to implement this, the main critical changes to the system involve altering the domain labels to $[-1, +1]$ instead of the original $[0, +1]$, the loss function for the domain loss is given below (6.8. There is one other critical addition that is a core part of the WGAN implementation: It is necessary to perform some form of gradient clipping in order to prevent the gradients from exploding and leading to mode collapse of the GAN, there are a number of methods of doing this, however in this implementation I introduce an additional loss during training time that penalises the system for excessively high gradients, this is an additional parameter that can then be weighted to adjust it's importance to the training procedure. In this format, the WGAN loss is used for the domain task, and the `categorical_crossentropy` loss discussed above is used for the sentiment loss.

$$L_{wasserstein} = \frac{1}{N}\sum_{i=1}^{N}(d_i\hat{d}_i) \tag{6.8}$$

Results The mean classification accuracy when using the loss functions described is given below:

Source_db	Target_db	MSE	HAGAN	Crossentropy	WGAN
MDSD2_books	MDSD2_dvds	0.51	0.76	0.75	**0.77**
MDSD2_books	MDSD2_electronics	0.52	0.76	**0.77**	0.76
MDSD2_books	MDSD2_homeware	0.51	0.77	0.76	**0.78**
MDSD2_combined	SE-2017-T4	0.39	0.73	**0.75**	0.75
MDSD2_combined	LMRD1	0.53	0.78	0.76	**0.80**
SE-2017-T4	MDSD2_combined	0.49	0.67	**0.69**	0.68
SE-2017-T4	LMRD1	0.51	0.70	0.68	**0.71**
LMRD1	MDSD2_combined	0.50	0.77	0.76	**0.78**
LMRD1	SE-2017-T4	0.61	0.78	0.79	**0.79**

Table 6.9: Evaluating the effect of modifying loss functions in the TransferGAN system

It is clear from these results that as predicted the MSE loss is unsuitable for this application, as it causes the GAN to collapse to predicted a single sentiment. Of the other evaluated loss functions however, there is relatively little difference in the final classification accuracy, although the WGAN method seems to show a slight consistent improvement. Additionally, while not reflected in these results, the WGAN algorithm also displayed more robustness to minor changes in parameters in tests that were run when working on the functionality of the system. For these reasons, the WGAN training method will be utilised in further experimentation.

The improvement shown in in the WGAN configuration (Wassertstein earth-mover distance function to measure domain loss) is likely due to the smoother and more consistent representation of the Wassertstein loss. It is able to create a smoother gradient in order to transition data from one domain to the other. Which is likely even more impactful in the TransferGAN task than in a typical GAN, as the transfer GAN is more reliant on encouraging 2 domains of significantly different data into a single distribution.

6.9 Comparison of Optimization Methods for The System

When training any type of neural network based machine learning system, there are a number of optimization algorithms available to use during training, and depending on the structure of the network, this can dramatically affect the performance. This is doubly important in a GAN system as the training loss for both the Generator and Discriminator will generally not significantly decrease during the training. Therefore It is necessary to ensure that the training algorithm works in such a way as to maintain the balance of "power" between the two adversarial networks. Because of this it is necessary to compare how the "typical" state of the art optimizer `Adam` compares to the more standard optimization method of the `RMSProp` (standard root mean squared back-propagation) algorithm.

Results The mean classification accuracy when using the optimization methods described is given below:

Source_db	Target_db	`RMS_prop`	`Adam`
MDSD2_books	MDSD2_dvds	**0.77**	0.52
MDSD2_books	MDSD2_electronics	**0.76**	0.51
MDSD2_books	MDSD2_homeware	**0.76**	0.51
MDSD2_combined	SE-2017-T4	**0.75**	0.61
MDSD2_combined	LMRD1	**0.80**	0.52
SE-2017-T4	MDSD2_combined	**0.68**	0.49
SE-2017-T4	LMRD1	**0.71**	0.52
LMRD1	MDSD2_combined	**0.78**	0.51
LMRD1	SE-2017-T4	**0.79**	0.61

Table 6.10: Evaluating the effective of modifying optimization methods for the TransferGAN system

These results clearly show that using the `Adam` optimizer is a poor choice for GANs of this type, this is quite obvious in hindsight, but still needed to be evaluated. This optimizer features internal mechanisms similar to a variable learning-rate, that adjusts the speed of optimization adaptively based on the gradient of the loss. However in GAN systems, the final goal of the loss is not to be minimized as it is an adversarial game. Use of the `Adam` optimizer caused the GAN to become very unstable, and the results seen above are the result of the classifier simply predicting all one class (i.e. the prior probabilities).

6.10 Comparing this System to State-Of-The-Art Sentiment Analysis

In order to get an understanding of the performance of this system, it is necessary to determine how it compares to the state of the art and baseline systems when they are run on the following tasks:

1. Sentiment classification trained on the *Source* dataset and tested on the *Target* dataset. This test is following the same rules that the GAN works under, and this should be considered as the quality of classification that we should be aiming to beat if the system is to be considered and improvement on existing technologies.

2. Sentiment classification trained on the *Target* dataset and tested also on the *Target* dataset. This is classic sentiment analysis, and this should reflect the ceiling of the possible classification accuracy for the *Target* dataset, the GAN system would ideally approach this value, but reaching it or getting very close is likely an unrealistic goal.

The baseline system under test here which will be given the notation of **GAN_nd**, will be the same network structure as the GAN: An encoder with identical parameters as the GAN system, followed by a **sentiment classification phase** but with no output for domain classification. And during training the system will only be optimised for sentiment classification, with no aspect of the loss being derived from the domain separability task (that is performed by the decoder phase of the GAN) In this way it should be possible to precisely identify the additional contribution that the GAN training regime provides.

Results The results of experiment 1 from this section are given below, evaluating transfer learning ability from a *Source* domain to a *Target* domain

		BERT		GAN_nd		GAN	
source_db	target_db	mean_acc	std_dev	mean_acc	std_dev	mean_acc	std_dev
MDSD2_books	MDSD2_dvds	**0.85**	0.003	0.76	0.010	0.77	0.013
MDSD2_books	MDSD2_electronics	**0.84**	0.007	0.75	0.007	0.76	0.011
MDSD2_books	MDSD2_homeware	**0.86**	0.018	0.75	0.010	0.78	0.002
MDSD2_combined	SE-2017-T4	0.71	0.014	0.74	0.006	**0.75**	0.013
MDSD2_combined	LMRD1	0.79	0.004	0.77	0.003	**0.80**	0.004
SE-2017-T4	MDSD2_combined	**0.72**	0.014	0.72	0.004	0.68	0.016
SE-2017-T4	LMRD1	**0.77**	0.012	0.64	0.026	0.71	0.006
LMRD1	MDSD2_combined	**0.80**	0.006	0.79	0.003	0.78	0.007
LMRD1	SE-2017-T4	0.68	0.013	0.74	0.009	**0.79**	0.002

Table 6.11: Comparing GAN based systems agains the BeRT system

These results demonstrate the incredible power of the BERT network. With no additional tuning, across the majority of domains tested, it demonstrates superior sentiment transfer-learning capability to the GAN system explicitly designed for this task. This alone is interesting, but possible unsurprising as the BERT system is a significantly more powerful network (140 million parameters) that has the advantage of pre-training it's language model on a very expansive language dataset. The results from experiment 2 (given below) demonstrate the difference in power of the two networks by showing how they perform in a standard sentiment classification task on the datasets provided.

From experiment 2 we can see that the BERT network holds a significant advantage in standard sentiment classification over the network we are using as our GAN. And these

	BERT		GAN_nd	
source_db	mean_acc	std_dev	mean_acc	std_dev
MDSD2_books	**0.87**	0.002	0.76	0.011
MDSD2_combined	**0.83**	0.003	0.76	0.006
SE-2017-T4	**0.86**	0.006	0.82	0.004
LMRD1	**0.88**	0.009	0.87	0.003

Table 6.12: Further comparison of the GAN-nd system against BeRT

figures should be considered to be the maximum possible accuracy (that can be achieved using current technologies) on these datasets.

Referring back to the results from experiment 1, it is interesting to note that results for the network with no GAN functionality are very similar to those from the GAN running normally. This result implies that the GAN method is not significantly contributing to the overall result (but nor is it detracting from it). This is a very surprising result that will be discussed in more detail in section 6.12

6.11 Leveraging The GAN for Label-Assisted Transfer Learning

In order to fairly assess how this system compares to other systems that would be used in a realistic scenario. It is important to determine how effectively this system is able to utilise varying amount of sentiment-labelled target-domain data.

In order to try and maintain consistency in the architecture of the system, the baseline transfer learning tests will be performed with the same network structure as the GAN system. A CNN document embedding system as described above, and a fully connected network for the sentiment classification system.

Source and Target Data Mixed The first method under test will be to include some percentage of labelled "target" domain data shuffled in with the main training set of "source" domain data. This allows the system to see some examples that are representative of the test set during training time, but does not give any additional weight or importance to those examples. This will be considered as a baseline, which should be surpassed by intelligently designed systems.

Source Pre-Train Followed By Target Tuning This method is "classic" transfer learning. The system will be fully pre-trained to classify sentiment for the "source" data. After this training is complete, the weights for the first half of the network (up to the document embedding) will be frozen. Finally the system will then be retrained on a small amount of labelled "target" domain data. This allows the system to learn how to generate sentiment specific document embeddings from its experience with the "source" set. And then the re-training of the classifier allows it to find a sensible separation that is valid for the "target" domain data.

GAN with Sentiment-Labelled Target Data In order to make use of labelled target data with the GAN system, it was necessary to add an additional component to the loss functions for the generator and the discriminator training. L^t_{sen} is the sentiment loss for only the sentiment-labeled target data. This allows it to be mixed into the loss functions with its own blending weight λ^t_{sen} and can therefore have its relative importance adjusted with respect to the other elements of the loss functions, similar to the losses described in (6.5) and (6.6)

Testing Methodology The method for running this test is as follows: Each system is provided the full amount of available training data from the "source" domain, then the experiment involves presenting a varying amount of sentiment-labeled "target" data, then fully training the network as described above, and testing the sentiment classification accuracy for a constant test set of "target" domain data. In this way it is possible to see how the systems make use of different quantities of "target" training data. In order to simplify the analysis of this experiment, only the results for Source: MDSD2 (Amazon Reviews) and Target: SE-2017-T4 (Twitter Data) will be included, as this is arguably the most difficult task available to us.

Results and Discussion The Classification accuracy of the system in the described experiments is given below (in both tabular and graph format):

	percentage_labelled_target				
Experiment	0%	5%	25%	50%	100%
Source and Target Mixed	0.69	0.74	0.76	0.79	0.79
Classic Transfer Learning	0.69	0.74	0.76	0.77	0.78
GAN with Target Injection	0.70	0.72	0.77	0.80	**0.81**

Table 6.13: Evaluation of GAN with limited access to labelled target data

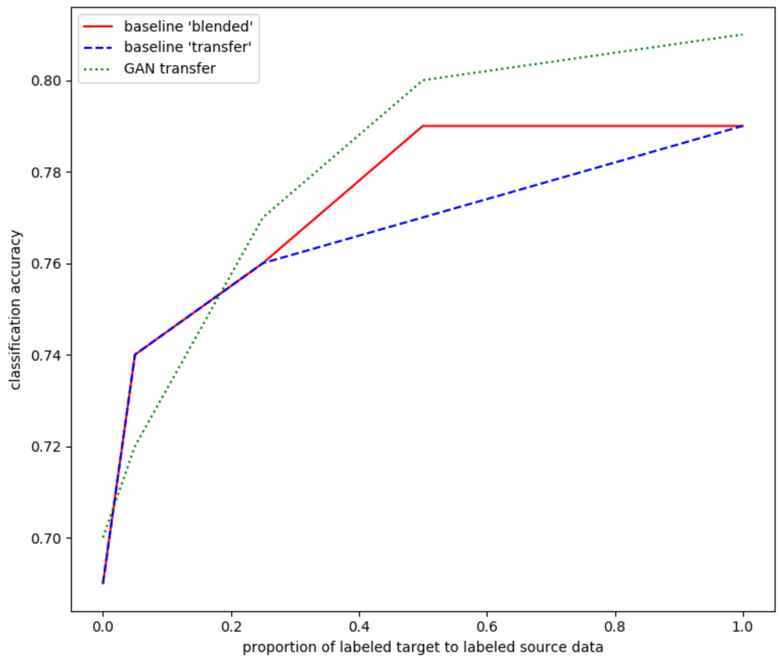


Figure 6.4: Graph of results for transfer learning experiment

From these results we can see the while the GAN with injection of labelled target data performs the best of the three systems, as we would expect. The results are again surprisingly similar. Additionally the expectation was the the GAN would be able to make the most value out of small amounts of labelled target data (5%) but of the three systems this is the only point where it actually performs below the baseline. While this is unexpected, the fact that the majority of these results are within 1% of each other makes this result less significant. The full implication of these results will be discussed in detail in section 6.12. This hypothesis was assumed because it was suspected that the GAN would need only a small amount of data in order to correctly learn the alignment between the sentiment axis of the different domains of documents, but these results suggest the hypothesis is incorrect.

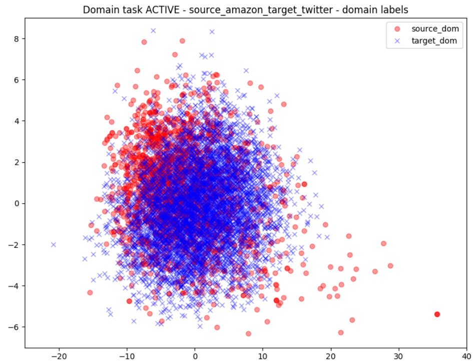
6.12 Additional Discussion of Results

The LSTM system evaluated in section 5.2 demonstrates the relative effectiveness of sequence focused systems on even short-document sentiment analysis such as on tweets. While this is not a novel result, it was important to verify how such systems react to various parameter changes and data format changes in order to move forward and conduct a proper tuning and evaluation of a more complex system such as the one described and evaluated in section 6.11.

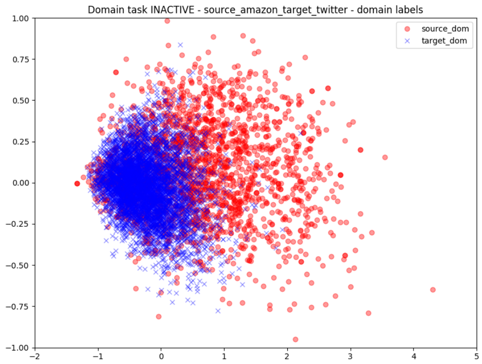
The experimentation on the GAN transfer learning system (section 6.11) yielded a number of surprising results that warrant further investigation. Primarily, in section 6.10 it was noted that the GAN system performed almost identically to the GAN_nd system (an identically parameterised encoder+classifier model without a domain-discriminator component present during training). There are two possible ways of interpreting this result:

- The GAN is not operating as expected, or the domain-indistinguishability task is contributing very little to the embedding of the documents.
- The GAN is operating correctly, however any benefit that might be gained by this method is being balanced out by some negative aspect.

Let us evaluate those two options: The first option is relatively simple to investigate. Given that the net result of the domain task is that the distribution of the two datasets should be close to identical. Therefore we can generate a compressed representation of all of the data-points in the space and make observations about the separability:



(a) PCA with domain task active



(b) PCA with domain task inactive

Figure 6.5: figures showing PCA of the document embedding space

The figures above suggests that the system is operating as expected and projecting the *Target* domain data into the same representation space as the *Source* domain data, and it seems unlikely that the domain task would be not contributing to performance while simultaneously not actively hindering performance. Therefore we must consider the second option: There is some negative aspect to this GAN embedding that is wiping out any gain we get by performing this mapping.

There are a number of hypothetical negatives aspects to the system that could be responsible for negating any performance gains in the system:

- The system is not properly exploiting shared sentiment information, and therefore is mapping **Positive** Target items to be neighbours with **Negative** Source items. This seemed likely the most plausible explanation, however this hypothesis is potentially refuted by the results presented in 6.4 that demonstrate that adding a small quantity of labelled data of the target domain does not cause a large increase in performance, which would be the expected result if this scenario was the case.

- It is also possible that the process of making the embedding domain-indistinguishable is causing a compression or an expansion of the embedding space in order to make it hard for the *Discriminator* to "keep up"; rather than the *Generator* "winning" by "fair" means, it exploits a feature of its network to adjust weights faster than the *Discriminator* is able to match, and this is actively hindering the ability to perform sentiment classification.

It is necessary to perform further analysis and experimentation of these issues if the GAN system is to be taken further and applied to real-world usages. And understanding this at a theoretical level could be an avenue for future work on this subject.

Finally, the significant improvement in classification accuracy granted by simply using the BERT classifier in its default setting suggests that it would be practical to investigate the effectiveness of a BERT-GAN system, that utilises the pretrained BERT language model as the generator stage of the GAN.

As demonstrated in Section 6.7 the GAN super-structure should be considered as more of a "training regime" than a "network" in its own right. Fundamentally for a system to be GAN, it simply means that the loss function is automatically optimised by the decoder phase during the training of the encoder. As such it is theoretically possible to use any model that can generate an encoding as the encoder phase of the GAN pair. It is then necessary to identify an appropriate model to use as a decoder that ideally is similarly powerful to the encoder network, this is also possible with transformer architecture like the BERT model. Therefore it would be a practical extension to this thesis to evaluate the performance of a state of the art encoder model such as BERT or GPT while still generating a loss based on domain indistinguishability as we have discussed in this chapter.

Experiments in this chapter have demonstrated that the GAN training regime allows it to exceed the performance of its underlying encoder in a number of transfer learning scenarios. Applying this system as a complement to an established state of the art encoder such as the BERT system could provide demonstrable improvements in the versatility of these systems for transfer learning. This would make a strong candidate for further research based on this thesis.

Chapter Seven

Meta-optimization of Transfer Learning using Grammatically Evolved Preprocessing

This chapter explores the idea of utilising meta-optimization to identify and refine preprocessing techniques that can improve classification results. In keeping with the theme of these thesis: transfer learning of sentiment analysis is considered as the primary proving ground for this methodology. However, as discussed at the end of this chapter, this research could be highly extensible to a wide range of different problem domains, and preprocessing methodologies.

In order to demonstrate the effectiveness of the concept, we choose to focus on a single style of preprocessing, that we will refer to as "sentence segmentation". In the first part of this chapter. We will introduce the concept of sentence segmentation, and then demonstrate provable effectiveness with a selection of designed rules.

In the next part of this chapter, we will discuss the potential weaknesses of manually designing rules for the sentence segmentation based preprocessing system. And introduce the use of Grammatical Evolution for generating rules simple rules that are provably optimized for the problem of sentiment transfer learning.

Finally we will discuss potential improvements to the grammar defined in the previous section; alternative preprocessing methodologies that this meta-optimization could be applied to; and other extensions to this work.

This chapter explores the idea of classifying sentences into a number of categories using grammatical rules. From there these sentences can be individually classified for some target label, and then recombined to classify a target document as a whole. This should allow us to create a model of sentence wise classification that is more suited to transfer learning.

Consider the following simple example: During training, our documents are made up for 50% sentences of type “A” and 50% sentences of type “B”. But the domain of documents that we wish to test on is an average of 90% type “A” and 10% type “B”. This chapter will discuss the idea that attempting to transfer learn using traditional methods between these domains will perform less effectively than if we are able to train a “type-A classifier” and a “type-B classifier” and then use a voting system to classify the document.

7.1 Sentence Segmentation

In order to demonstrate the effectiveness of the system discussed in the latter part of this chapter, we introduce a simple preprocessing methodology: "Sentence Segmentation".

This method relies on an assumption that while the content, tone, writing style, and specific vocabulary will vary greatly across documents of different domains. Therefore considering the documents as a whole can lead to degradation of performance. But there are likely to be components of documents that can be grouped into categories that are ubiquitous across domains.

Consider the following extracts from two significantly different domains of the Amazon Product Review data:

'[...] This book has something to offer everyone. I know I will be revisiting it many times in the future.'

'[...] It has a sleek modern design and it toasts bread evenly and quickly. It has all of the features I need and none of the ones I don't.'

While these two extracts both demonstrate domain specific vocabulary that would be hard to transfer learn in isolation, there are some stylistic similarities, which are what we hope to exploit with this method. Note that both examples feature one sentence that discusses features of the "product" in a passive third person voice. And one sentence that speaks from the point of view of the reviewer about their interaction with the "product".

The next step in this methodology is to group sentences into categories like the two described above. Individual classifiers can then be trained on these categorised sentences. This allows us to significantly reduce the complexity of the problem space, and, as we will see later in this chapter, train effective classifier systems using less available data. This also allows us to introduce a classifier voting system in which sentences that have high confidence of a particular classification can outvote low confidence sentences. Per the example above, if trained on reviews of books, the sentence *'[...] it toasts bread evenly and quickly.'* may be hard to accurately assign a positive sentiment to. This should however lead to a low confidence score that allows the final document classification to assign more weight to the classification of the sentence *'...] It has all of the features I need'* which would likely to be easier to learn as positively sentimental just using training data from book reviews.

This method's primary advantage comes from allowing the system to give priority to well-learned features in a robust rule based way. Consider two example domains: A and B, and we create a rule-set that defines two sentence types x and y. Our training domain (A) is primarily (90%) *sentence-x's* and our target domain (B) is primarily (90%) *sentence-y's*. If we made no attempt to apply the above method, then our classifier would be attempting to apply features learned on x type sentences to a dataset of y type sentences, which would likely lead to poor performance. Whereas using sentence segmentation, the y type sentences have their own dedicated classification system that has been optimized for the use-case. It is clear that the y classifier will have had access to a smaller training set, and therefore be a potentially weaker system; however it will not have been biased toward the classification of type x sentences, and should therefore have still learned some pertinent features for classification. This should lead to a better final result for classification of documents in domain B.

7.2 Demonstrating the Effectiveness of Sentence Segmentation

In order to justify that the idea is worth further exploring. We will demonstrate that at least one set of simple human-designed rules can perform better than the baseline for the task of sentiment-analysis transfer-learning.

7.2.1 Defining a rule-set

In order to demonstrate the effectiveness of this method, and show that the idea is worth further exploration and optimization, two rule-sets are used: A simple one, similar to that described above, and a more complex one.

The first rule-set is as follows:

- Sentences that Include the word 'I'
- All other sentences.

This first set is extremely simplistic, but we should see that even with such a set, there is still a demonstrable improvement in transfer learned classification over the baseline.
The second rule-set used is:

- Sentences that include:
  - Personal pronouns ('I', 'you', 'he', 'she', etc.)
  - Relational nouns ('Wife', 'Husband', 'Sister', etc.)
  - Subjective indicators ('Think', 'Feel', etc.)
- All other sentences.

When used on the product review dataset, both of these rule sets are designed to accomplish a similar outcome: Avoid creating one complex sentiment classifier, and instead break the task down into two simpler classifiers: The first classifies sentiment in majoritively subjective descriptions of **internal** sentiment: *I think, I feel, I want, I like, I hate.* And the second classifies sentiment in majoritively objective descriptions of products: *It is fast, The book* *was long, The toaster is sturdy.* This second category of sentences is likely to have difficulty with domain independence, as some adjectives that are **good** in one domain are **bad** in another domain, however this will be a minority of cases. And with the voting scheme that incorporates information from category 1 sentences, we should still see improvement from the baseline system.

7.2.2 Classifier implementation

The classifier we are using for this evaluation will be a multi-layer LSTM similar to the system used in 5.3 featuring a multi-layer LSTM in order to model the sequencial nature of the data. For the experiments detailed below, two configurations were considered for the baseline. One featuring twice as many trainable parameters as the first. This was done to avoid the potential error of concluding that a system that features two separate classifiers performs better than a system with only one classifier without considering that it would present it with more available resources in order to optimize the problem. The results from these two classifier modes were similar enough to be statistically insignificant. But the results detailed below use the baseline classifier that has twice as many parameters as the individual classifiers used in the segmentation based system.

7.2.3 Experimental Design

The hypothesis under test in this experiment can be stated as the following:

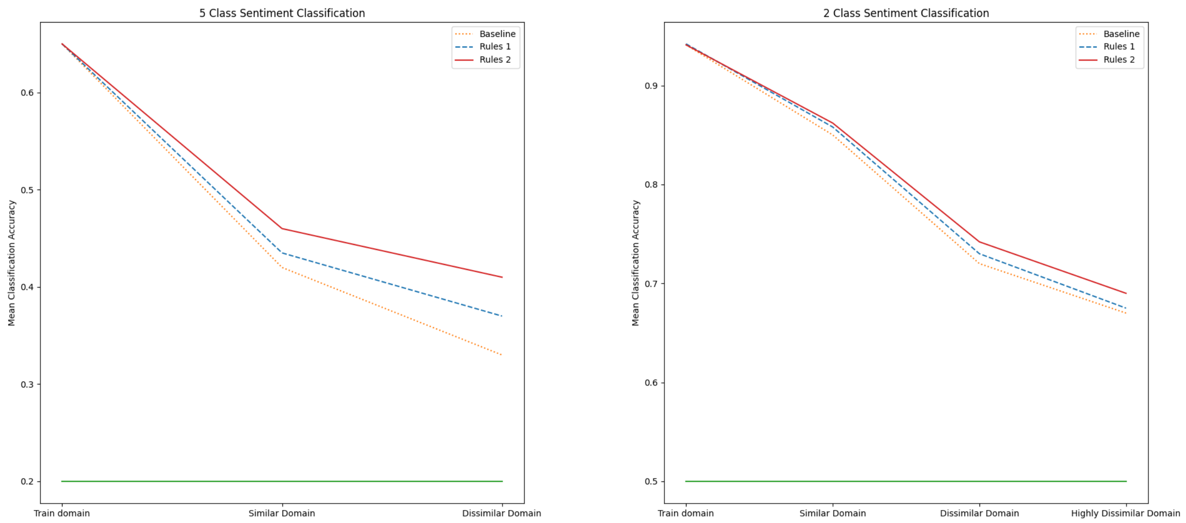
When using sentence-segmentation based preprocessing, classification performance will decrease at a slower rate than without the preprocessing as the domain similarity decreases.

This hypothesis will be tested in two different scenarios.

- 5 class classification (based on star rating) of three different domains of Amazon, using the following domain aliases:
  - Train Domain: Automotive
  - Similar Domain: ToolsHomeImprovement
  - Dissimilar Domain: Books
- 2 class sentiment classification (positive vs negative), and will also include Twitter as a "Highly Dissimilar" Domain
  - Train Domain: Automotive
  - Similar Domain: ToolsHomeImprovement
  - Dissimilar Domain: Books
  - Highly Dissimilar Domain: Tweets (SemEval)

7.2.4 Results and Discussion

The results of the two scenarios are given below:



(a) 5 Class classification

(b) 2 Class classification

Figure 7.1: Change in classification accuracy with decreasing domain similarity

These results show that, as expected, the ability of all systems to classify sentiment decreases as the similarity between the train and test domains is reduced. However it can be seen that both of the sentence-segmentation preprocessing systems showed notably better performance in the out-of-domain testing. Additionally, the more sophisticated rule-set demonstrates better performance than the simple rule-set. This provides a strong justification to continue this avenue of exploration in order to discover a provably optimal rule-set for this problem.

It can be seen from figure 7.1b, that while this same trend is present, it is significantly less pronounced. This can likely be attributed to the the relative simplicity of this task compared to the 5-class classification task. As a result all models are able to approach optimal performance for the available data, and the unclassifiable elements of the test set rely on features that cannot be learned from the training domain. However the primary goals of this experiment were to demonstrate the effectiveness of the sentence-segmentation based pre-processing; and to show that better definition of the rule-sets can lead to a demonstrable improvement in performance of the system. Both of these goals have been met by the results of the experiment. However this work could be extended by further exploration into the reduced performance of the system on the 2-class classification problem.

7.3 Grammatical Evolution of Sentence categorization rules

In the previous section we have demonstrated that this method can produce an improvement over the baseline, however the rules used above were discovered experimentally and are in no way provably optimal for this task. In this section we present a method of meta-optimization of these categorization rules for the task of transfer learning sentiment analysis. Using Grammatical Evolution (and techniques discussed in 2.7), we create simple algorithms to group sentences into categories. We then evaluate these algorithms by their effectiveness at the task of transfer learning sentiment using the methodology described above.

7.3.1 Defining a Grammar

Defining an appropriate grammar is the first critical problem that must be addressed when creating a system that relies on Grammatical Evolution. As, formally speaking, the Grammar will define the bounds of our available search space, and the GE algorithm simply identifies optima within that space. The Grammar below represents what we consider to be the minimum viable set of rules in order to approach this problem. By remaining small we hope to restrict the search space somewhat for run-time reasons. This Grammar could be expanded in future experiments if this methodology is demonstrated to be a viable avenue for progress in this field.

<start>	::=	<expression>
<expression>	::=	RETURN *category-id* | IF <condition> THEN <expression> ELSE <expression>
<condition>	::=	NOT <condition> | <condition> <bool-comb> <condition> | STARTS WITH <pos-string> | CONTAINS <pos-string> | LENGTH GREATER THAN <min-len> WORDS
<pos-string>	::=	<pos-string> <pos-tag> | <pos-tag>
<bool-comb>	::=	AND | OR
<min-len>	::=	5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 | 233
<pos-tag>	::=	*all of the possible part-of-sentence tags*

Where a POS tag is as described in 2.1.1, see appendix A for a full list of the POS tags and their meanings.

This grammar relies on simple boolean combinations of the following rules:

- contains *some continuous string of (1 or more) POS tags*
- starts with *some continuous string of (1 or more) POS tags*
- sentence length greater than x

While this grammar appears simplistic, it is based on robust principals, and can be used to define highly generalised rules (such as "Contains POS-TAG[PRP]" as we saw above). Or extremely sophisticated rules, by using boolean combinations of longer 'POS strings' that range from extremely simple to moderately complex. The simplicity of the grammar also reduces the size of the potential solution space that our optimizer will need to search in order to discover effective rule-sets. This makes it an ideal tool for demonstrating the effectiveness of this methodology.

It is appropriate to use an author defined grammar in this case as the goal of this chapter is to demonstrate the potential effectiveness of a solution involving Grammatical Evolution. The goal is not to present an optimal grammar for this particular problem space. The hypothesis is that if this chapter is able to demonstrate that a simple author-defined grammar is able to meaningfully improve the performance of a system over the baseline then it validates the potential of this method and opens an avenue for further study optimising the grammar.

7.3.2 Implementation of the Grammatical Evolution Framework

As with all experiments presented in this thesis, the grammar framework was implemented in python. It utilises a "gene-string" which is a list of integer numbers of arbitrary length in order to perform the evolutionary algorithm component of the optimisation. The grammar construct is then implemented as a tree where each node of the tree can have a number of potential children, i.e. the <expression> node could have RETURN <id> or IF <condition> THEN <expression> ELSE <expression> nodes as children, where the *terminal* nodes then contain values such as 5 without children. In order to build a tree of a specific program, the list of integers is iterated through and used to select a choice of child nodes with a simple modulus of the number of available options (i.e. if the value is 6, and there are 2 options, $6\%2 =$ option-0. This tree can then be rendered into a program, and that program is used to process the input data into groupings based on the conditions and expressions. Networks are then trained with their respective groups of data. Finally at inference time, the programs are executed on the test data to segment it into groups and assign it to its group-wise classifier for inference before a final vote is conducted to give a global score.

7.3.3 Grammatical Evolution using Simulated Annealing

With a context-free grammar defined, a search algorithm can be created to explore the problem space. For this we will use Simulated Annealing, as discussed in 2.7.2. Simulated Annealing was selected over a classical Evolutionary algorithm primarily because, as discussed below, the process for scoring each individual involves the initialisation, training, and testing of a full neural network based sentence classification system. This is an extremely expensive cost function for an EA system, as for a typical EA we might define an initial population size of 50-200 individuals (meaning 50-200 full train/test runs of a neural network) just for the first generation of the EA, each subsequent generation would then need a similar number of evaluations, and this would likely continue for 20 or more generations to begin to notice significant convergence. Meaning a total of 500-4000 full system evaluations. This was not feasible with the resources available for this project. Therefore a simulated annealing approach that would require closer to 100 evaluations in order to approach a (less globally optimal) local optimum solution was chosen.

7.4 Hyperparameter Tuning

Due to the nature of this system being a meta-optimisation process already applied around a neural network training regime, the scope of hyper-parameter tuning presented in this chapter will be less than in other chapters of this thesis. The network being utilised is the same network optimised in 5.1 which has been optimised for general sentiment analysis tasks. There are a number of hyper-parameters of the GE process that also needed optimisation. In order to generate a score, the GE process was run for 20 generations and then the produced system was evaluated against the SemEval sentiment classification task using an accuracy metric. This was run for the minimum 10 generations on the Bayesian Hyper-parameter optimisation system in order to discover optimal parameters. The search space for parameter evaluation was as follows:

Table 7.1: Parameter optimisation space for EA Individuals

Hyperparameter	**Type**	**Range / Values**
`gene_mutate_method`	Categorical	{random, incremental}
`gene_length`	Categorical	{10, 25, 50, 100}
`mutation_prob_start`	Categorical	{1.0, 0.75, 0.5}
`mutation_prob_end`	Categorical	{0.5, 0.25, 0.1, 0.0}

where the parameters are explained as follows:

- `gene_mutate_method`: when a single item within a gene string mutates, it can either randomly select a new value ("random") or increment by 1 to iterate through all the options eventually via mutations ("incremental")
- `gene_length`: the length (number of integers) in the gene. A longer value means that the system has more capability to completely explore the search space, but implies a longer time to reach an optimal solution due to the impact of mutations late in the gene string (if a late value mutates, and the program tree is built before that value is consumed, then that is a useless mutation)
- `mutation_prob_start`: the probability of each item in the gene mutating decreases linearly as the gene is iterated through (to mitigate the problem in the item above). this is the multiplier of the first integer in the gene
- `mutation_prob_end`: The probability multiplier of the last integer in the gene mutating.

Additionally there are a number of hyperparameters of the simulated annealing process, for the simulated annealing process a linear cooling schedule was used, so the simulated annealing parameters can be simplified

- **P at** t_0: The probability that a lower scoring solution is accepted over the current solution at time-0
- **generations until** $P = 0$: The number of generations until the probability of accepting a worse solution is 0.0

Table 7.2: Parameter optimisation space for Simulated Annealing

Hyperparameter	**Type**	**Range / Values**
`P at` t_0	Categorical	{1.0, 0.75, 0.5, 0.25}
`generations until` $P = 0$	Categorical	{20, 40, 60, 80}

The selected parameters using Bayesian optimisation for the systems utilised in this chapter are as follows:

Table 7.3: SemEval RNN Optimal Hyperparameter selection

Parameter	**Value**
`gene_mutate_method`	"random"
`gene_length`	50
`mutation_prob_start`	1.0
`mutation_prob_end`	0.1
`P at` t_0	0.5
`generations until` $P = 0$	80

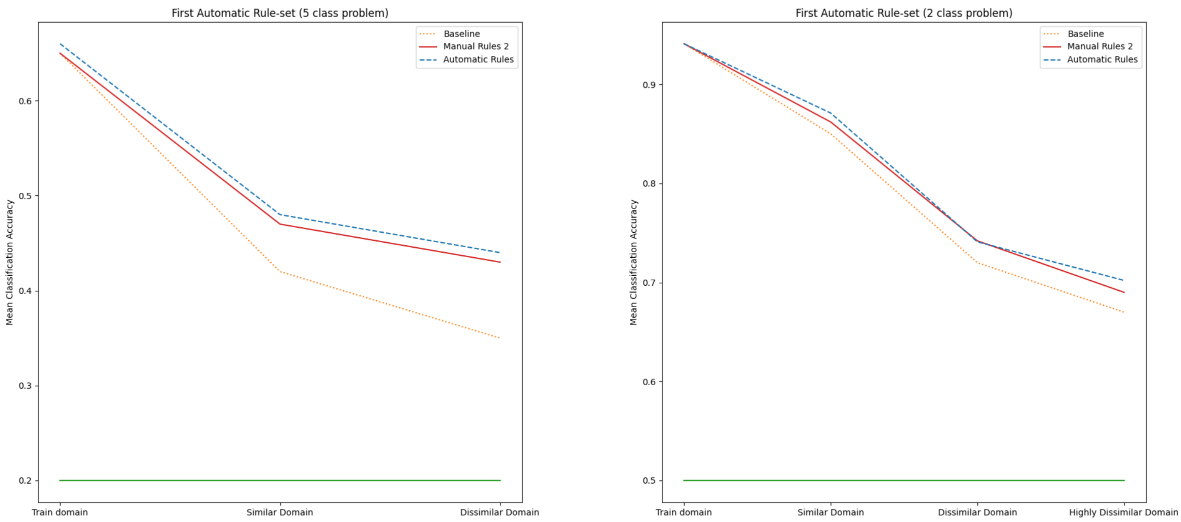
7.5 Results and Discussion

7.5.1 Generated Rule-sets and their performance

In this section we will show some rule-sets generated by the system, and show how their performance compares to the manually created rules discussed earlier.

```
if {inp_pos_tags} starts_with [NNS]:
    return: <class 1>
else:
    return: <class 2>
```

This first program is highly simple, and easy to understand both semantically and intuitively. The two classes are sentences that start with singular nouns. Intuitively it seems this rule will be somewhat effective as it allows the system to differentiate between sentences that use a significant amount of plural words and tenses, versus those that use primarily singular, for example “[this] has” vs “[these] have”.



(a) 5 Class classification

(b) 2 Class classification

Figure 7.2: Evaluation of first automatically generated rule-set

It can be seen from the results that this rule-set showed marginal improvement over the manually generated rules. While this is positive, it is not enough of an improvement in performance to be considered significant.

The next program is a stark contrast to the one shown above and highlights some of the deficiencies of this system approach. As it is extremely challenging to understand for a human interpreter without simplification:

```
if NOT (NOT (NOT (len({inp_str}) > 5))):
    if NOT ((NOT (({inp_pos_tags} starts_with [UH] OR {inp_pos_tags} contains [IN]))
        AND len({inp_str}) > 8)):

        return: <class 1>
    else:
        if {inp_pos_tags} starts_with [VBZ]:
            return: <class 2>
        else:
            return: <class 3>
else:
    if NOT (len({inp_str}) > 144):
        if ((NOT (len({inp_str}) > 5) OR NOT (NOT (len({inp_str}) > 5)))
            AND NOT ((NOT (len({inp_str}) > 8)
            OR NOT (len({inp_str}) > 233)))):

            return: <class 4>
        else:
            if len({inp_str}) > 21:
                return: <class 5>
            else:
                return: <class 6>
    else:
        if {inp_pos_tags} starts_with [VBZ]:
            return: <class 7>
        else:
            return: <class 8>
```

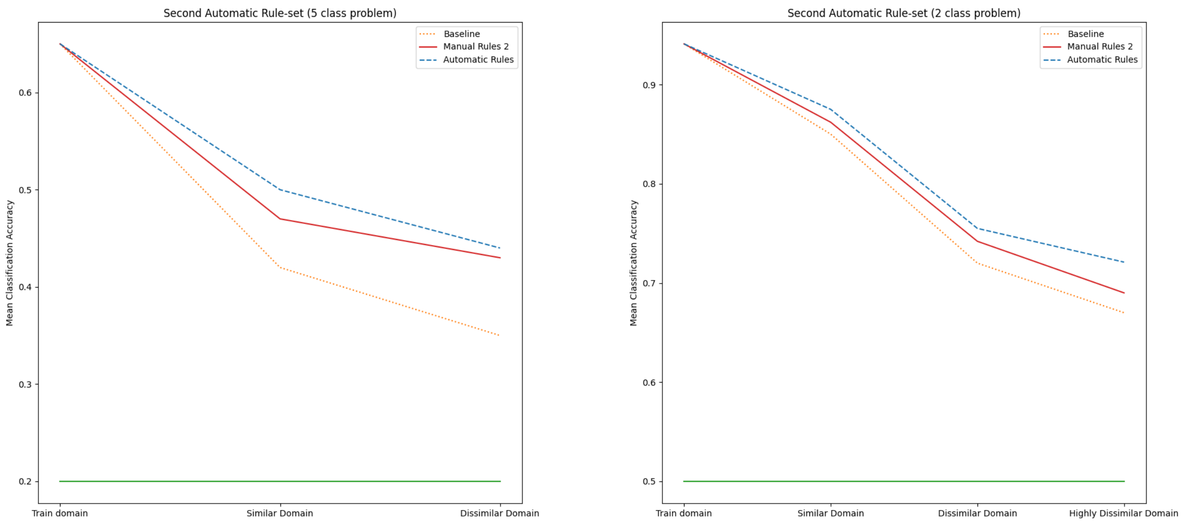
it should be noted that only classes 1, 5, and 6 are shown to be used by items in the data, the simplification of this program demonstrates that several of the classes are theoretically unreachable, and others are just unreached in practice. This returns to the statement made earlier about it guaranteeing syntactically valid programs, but not necessarily semantically valid ones. But while it damages the readability of the program, these closed branches of the program are incredibly healthy to have from an evolutionary algorithm perspective as it allows single flipped genes to close or re-open large components of the program to the advantage of the individual.

The simplified program (as it is used by the data) is given below:

```
if len({inp_str}) <= 5:
    return: <class 1>
elif len({inp_str}) > 21:
    return: <class 2>
else:
    return <class 3>
```

After simplification it can be seen that while there are some POS tag based rules included in the program, they are either unreachable or not present in real data, and the effectiveness of this rule-set is based on grouping sentences by length into 3 categories: short, medium, and long.

The result of this program against the test domains can be seen below:



(a) 5 Class classification

(b) 2 Class classification

Figure 7.3: Evaluation of second automatically generated rule-set

These results demonstrate a significant improvement over the baseline and manually created rule systems, especially in the 2-class classification problem that includes the twitter data. This result is interesting as the most effective system demonstrated made no use of the available POS tag based rules available for optimization. There are a number of possible explanations for this:

- Especially with reference to the twitter data (although this was not used for scoring candidates in the GE process) focusing on sentence length is likely a very effective method to transfer from a longer format document such as a product review, onto tweets which are renowned for their brevity.

- This is a 'perverse' optimization such that rather than optimizing for the problem set, it optimized against some exploitable weakness in my classification system. Such that training against sentences of more discrete length produces better results. But in further investigation I was not able to prove such weakness.

- Because of the incredible computational cost of this process, and therefore the relatively weak training procedure using simulated annealing, this could be a hyper-local optima that is much easier to generate than an effective system that makes use of precise combinations of POS tags.

This process of meta-optimization using Grammatical Evolution has, in this chapter, been shown to be an effective method of preprocessing for transfer learning applications. There are a number of possible extensions that could continue this work:

Further refinement of meta-optimization process As mentioned above, this work relies on an extremely computationally expensive cost function in order to try and discover optima in the search space. As such, further work could be done to apply more resource and attempt to streamline the process, or using parallel processing in order to use a more traditional form of Grammatical Evolution that involves EAs.

Investigating other pre-processing or processing tasks to optimize This work focused on a single style of preprocessing: 'sentence segmentation'. There is no reason that this methodology should be limited to this one style however, and further work could be done to investigate other preprocessing methodologies that this could be applied to, or meta-optimization of the system or network itself using either neuro-evolution style optimization or a grammar based approach as shown in this chapter.

Further refinement of grammar This work also applied an extremely basic grammar to the task in order to demonstrate that this system was effective at optimizing the available tools and applying them to the task of transfer learning. There is significant room for further development of the grammar in order to improve it for the 'sentence segmentation' style preprocessing, or to apply it to another method as mentioned above.

Comparison against, and integration with state of the art systems This work primarily uses a simplistic network based classifier (when compared to state of the art). This was done for computational complexity reasons in order to keep the run-time of the system withing acceptable bounds. In order to truly evaluate it's effectiveness as a methodology it should be compared against state-of-the-art systems in sentiment transfer learning. However as this is a pre-processing methodology, the way to test it fairly in that scenario would be to attempt to integrate it with those state-of-the-art technologies and determine if their effectiveness is improved with it's inclusion. This would require significant computing resource, but may be a route worth pursuing in future

Chapter Eight

Application of Novel Transfer Learning Techniques to the Motivational Interview Dataset

In this chapter, we demonstrate how the systems discussed previously, primarily those in chapter 6 and 7 can be used to generated labels on the previously unlabelled Motivational Interview data, we perform an evaluation of the quality of these generated labels. And Discuss the system performance with respect to this task.

8.1 Traditional Classifier Performance on the MI Data

In order to contextualise the results of transfer learning systems, it is important for us to understand the classifier performance of a non-transfer learned system, in order to understand the complexity of the problem space.

For these experiments, an LSTM based classifier, similar to that explored in Chapter 5 is used. Due to the heavily unbalanced nature of the data, specifically when looking at the document-level statistics. The data was split into 3 folds, where each fold contains 2 (of the available 6) positive documents, and the remaining data was randomly sampled into the folds. The classification experiments in the section were run with training on 2/3 of the data, and evaluation on the last 1/3 - for all 3 folds. This data was then combined into a single confusion matrix of classification counts in order to generate statistics. Due to the very low number of positive items, generating the statistics on a per-fold basis and producing a mean and stddev was not appropriate (as the f1 score of a class that does not get predicted at all is undefined, which happened in some individual folds, but was did not an issue with the summed results).

For all experiments in this chapter, the results will be given in the form of a confusion matrix with a normalised colour-map to aid interpretation. This tells a more complete story than simply providing metrics such as accuracy or F1-score, as it clearly visualises which part(s) of the 3 class problem a particular configuration excels or struggles at.

8.1.1 Document and Sentence Sentiment Classification

A sentiment classifier system as described above was trained to classify the sentiment of the motivational interview dataset at first a document level, and then at an individual sentence level. The raw results of the evaluation of those classifiers is given below.

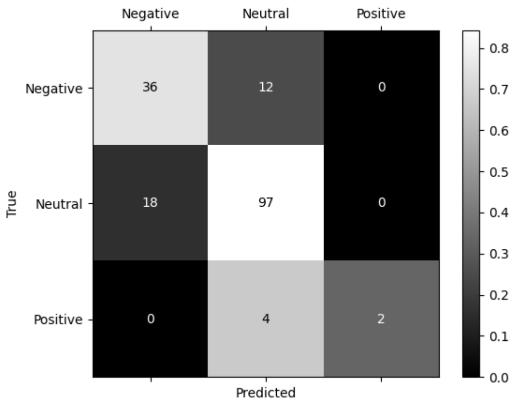


Figure 8.1: The confusion matrix of traditional document classification of the MI data

Document-wise sentiment classification: A traditional sentiment classification system has a macro-f1 score of 0.686 at the whole-document level.

Sentence-wise sentiment classification: A traditional sentiment classification system has a macro-f1 score of 0.687 at the sentence level.

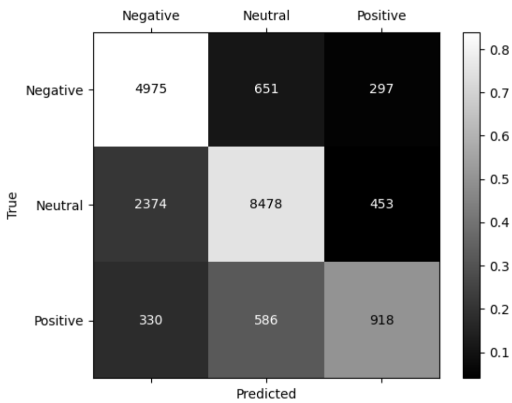


Figure 8.2: The confusion matrix of baseline sentence classification of the MI data

8.1.2 Managing Class Imbalance

As this dataset is highly unbalanced away from the *positive* sentiment class label, data augmentation was tested, for this experiment the strategy of class weighting used in 5.2.1 was applied when training the LSTM classifier. The results of that classification are given below:

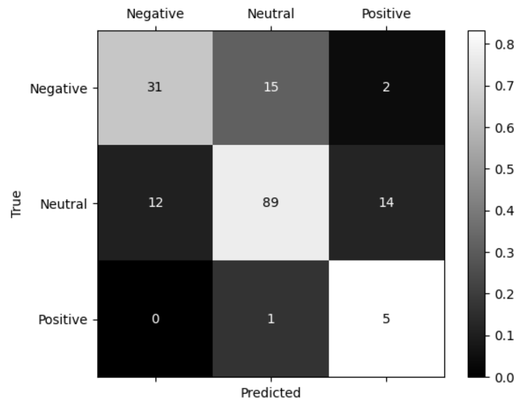


Figure 8.3: The confusion matrix of baseline document classification of the MI data with class weights applied

Class weighted sentiment classification: With class weighting applied, the classifier performed worse by the macro-f1 metric. With a total score of 0.620

8.1.3 Traditional Classification Results

In this section, we have generated some classification metrics, with which we can evaluate the performance of our transfer learning systems in the remainder of this chapter. Additionally we briefly explored the use of class-weighting in order to re-balance the decision boundaries of the MI data to observe the impact of minority class classifications. While this improved the recall rate of the *positive* sentiment class. It reduced the precision, and overall had a negative impact on the F1. It is possible that using a more sophisticated data augmentation strategy would yield better results, and could be an avenue for further exploration. However, the goal of this thesis is to demonstrate transfer learning methodologies for data with no, or few labels, and therefore an unknown distribution of classes. Therefore further pursuit of more sophisticated augmentation tools for this data would not significantly alter the conclusions presented in this chapter.

8.2 Application of Transfer Learning Methodologies

In this section we will apply 3 of the systems discussed in this thesis to the Motivational Interview data. The data we will be using as training for each of them will be the sentiment-balanced variant of the Amazon Review data discussed in 3.3.

- An LSTM that has been trained on Amazon Review data but has not had a training procedure altered for the purpose of transfer learning. This system will serve as a baseline in order to demonstrate that the transfer learning specific training regimes of the other two systems materially improve the performance of those systems.
- The TransferGAN system developed and discussed in Chapter 6.
- The Grammatical Evolution of pre-processing rules developed and discussed in Chapter 7

8.2.1 Document Classification using Transfer Learning

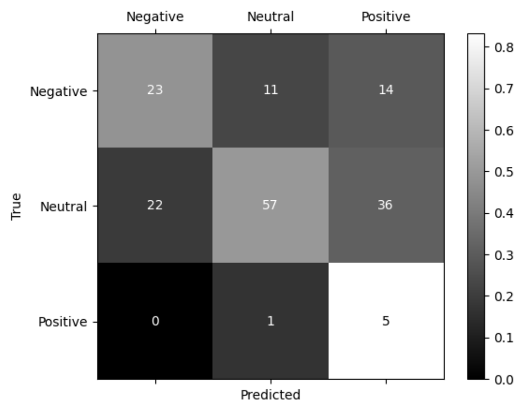


Figure 8.4: The confusion matrix of of transfer learned document classification using a standard LSTM

LSTM Performance: The standard LSTM system produced a macroaverage F1 score of 0.426 on the classification of Motivational Interview data full transcripts. This system shows a notable bias toward predicting positive sentiment, similar to the data-augmented classifier in Figure 8.3, likely due to the underlying prior probabilities in the Amazon Review data.

TransferGAN performance: The TransferGAN system produced a macroaverage F1 score of 0.561 on the classification of Motivational Interview data full transcripts

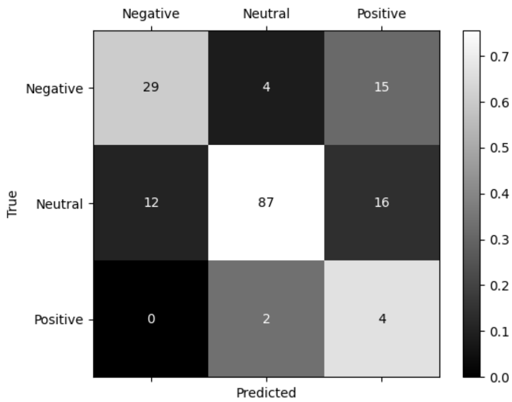


Figure 8.5: The confusion matrix of transfer learned document classification using TransferGAN

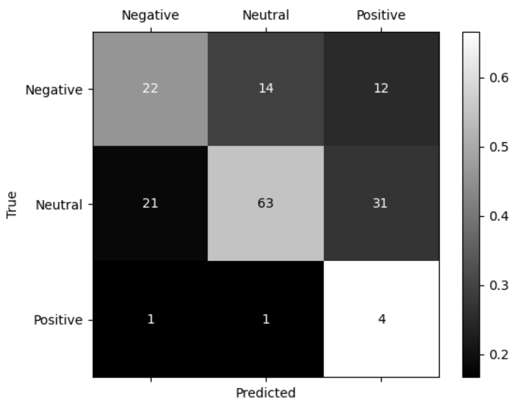


Figure 8.6: The confusion matrix of transfer learned document classification using Grammatically Evolved Preprocessing

GE Preprocessing performance: The Grammatically Evolved Pre-processing system produced a macroaverage F1 score of 0.427 on the classification of Motivational Interview data full transcripts. Like Figure 8.4, this system demonstrates a bias toward false prediction of the *positive* sentiment class.

8.2.2 Sentence Classification using Transfer Learning

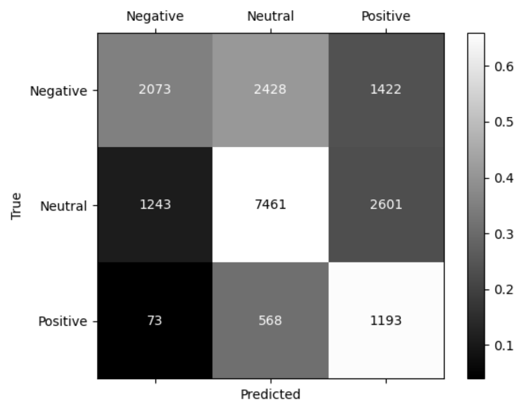


Figure 8.7: The confusion matrix of of transfer learned sentence classification using a standard LSTM

LSTM Performance: The standard LSTM system produced a macroaverage F1 score of 0.490 on the classification of Motivational Interview data at a sentence level

TransferGAN performance: The TransferGAN system produced a macroaverage F1 score of 0.594 on the classification of Motivational Interview data at a sentence level

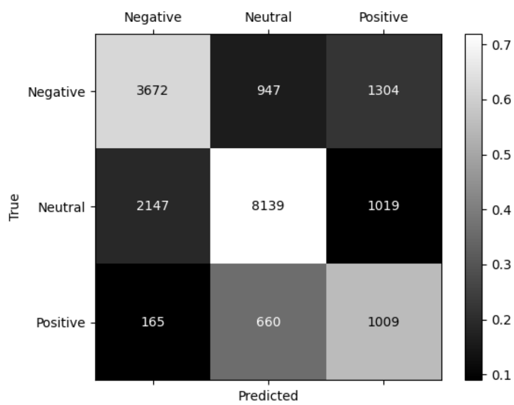


Figure 8.8: The confusion matrix of transfer learned sentence classification using TransferGAN

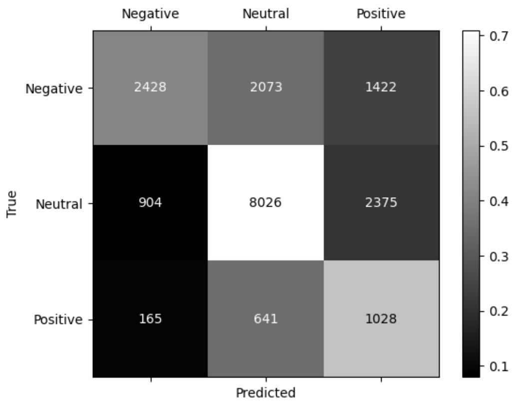


Figure 8.9: The confusion matrix of transfer learned sentence classification using Grammatically Evolved Preprocessing

GE Preprocessing performance: The Grammatically Evolved Preprocessing system produced a macroaverage F1 score of 0.517 on the classification of Motivational Interview data at a sentence level

8.2.3 Consolidated Results

The tables below show the result of the baseline experiment in 8.1 compared with the classification performance of the transfer-learning systems as trained on the Amazon Review dataset.

System	**Document-wise macro-F1**	**Sentence-wise macro-F1**
Trained on MI data	0.686	0.688

Table 8.1: Sentiment Classification Results on MI Data

System	**Document-wise macro-F1**	**Sentence-wise macro-F1**
Standard LSTM System	0.426	0.490
Transfer GAN	**0.561**	**0.594**
GE Preprocessing	0.427	0.517

Table 8.2: Sentiment Transfer Learning Results on MI Data

From this table, we can see that the TransferGAN system significantly outperforms the baseline LSTM system that was used as a reference. Coming close to the performance of the sentiment classification system trained directly on the data. This will be discussed further in 8.3

8.3 Discussion

In this chapter, we have performed a sentiment analysis task on the Motivational Interview data, according to the labels assigned by the author as part of this Thesis. We Evaluated a baseline model that is trained on our training set but has no transfer learning component. And finally we evaluated the two primary systems developed in this thesis.

The GE Preprocessing system developed in Chapter 7 demonstrated a minor improvement over the baseline for sentence-wise classification. But was not able to achieve a significant improvement over the baseline for document-wise classification of sentiment. This indicates that the rules learned for transfer learning between the domains of the Amazon Review dataset do not provide a material advantage to transferring between significantly dissimilar domains such as between the Amazon Review dataset and the MI dataset. This result disagrees with the results presented in Chapter 7 where the system demonstrates an improvement over the baseline when transfer learning onto the SemEval dataset. This difference is likely explained by the significant increase in complexity of classification in the medical domain and on transcribed conversation rather than the controlled text of Product Reviews. Perhaps by using a domain more comparable to the MI data for training the preprocessing system, a viable rule-set could be learned. This is a potential avenue for further research on the subject.

The TransferGAN system developed in Chapter 6 demonstrated a significant improvement over the baseline for both sentence and document-wise sentiment classification of the MI data. The classification metrics were close to that of the system trained directly on the MI data, which suggests that the TransferGAN was able to very effectively model the internal structure of the MI data in order to map the sentiment learning from product reviews onto it. This is a very promising result that validates the use of this system for low-domain-similarity transfer learning applications such as this.

Chapter Nine

Conclusions and Closing Remarks

9.1 Discussion of Research Questions

In this section, we will examine the research questions and discuss how this work contributes toward answering those questions:

9.1.1 Research Question 1

"To what extent can we classify emotion in conversational speech? And can this be used as an analogy for the classification of sentiment and conversational engagement?"

This research question is primarily addressed by Chapter 4. In that chapter we examine a dataset of emotional speech, and apply a number of well-tested techniques and models to it to classify the emotional content of the vocalised speech. The final classification results can be found in that chapter. The secondary part of this question was not sufficiently answered by chapter 4, and would be, as discussed in several parts of this thesis, a strong candidate for further research.

9.1.2 Research Question 2

"Can we define a measure similarity between different domains of text based data?"

This research question is addressed in chapter 5, we establish a a system in which we can quantify the **relative** similarity between any two domains of data. It is highlighted that this is a relative meaning, and typically requires at least 3 domains of data in order to create context for what this similarity metric means in practice. We then demonstrate this system on a collection of Amazon review data, as well as other datasets relevant to this thesis and establish a spectrum of domains that range from similar to highly-dissimilar in order to support further work.

This is still a highly open research question however, and as noted this metric is both relative and likely to be non-linear in nature so it is hard to use in practical applications where the specific value of similarity score might be more relevant.

9.1.3 Research Question 3

"In the scenario of having little or no labelled data for a sentiment analysis problem, what performance can we achieve using current transfer learning methodologies on similar data?"

This research question is first encountered in chapter 5, where some baseline models for sentiment analysis and transfer learning are considered and evaluated. We also demonstrate these baseline models in the subsequent chapters where they are compared against the novel methodologies under test.

9.1.4 Research Question 4

"Can the performance of the above scenario be improved using novel transfer-learning methodologies?"

This is one of the core research question of this thesis. Chapters 6 and 7 demonstrate two novel approaches to the transfer learning problem: 'TransferGAN' which leverages adversarial learning techniques in order to exploit any available similarities between the source and target domain. And a Grammatical Evolution based pre-processing system, that uses an exemplar preprocessing method to demonstrate that many aspects of the sub-components of sentiment analysis could be further optimized for the task of transfer learning.

9.1.5 Research Question 5

"To what extent does the performance degrade as we decrease similarity between the source and target domain datasets? and can this degradation be mitigated?"

Chapter 6 and 7 include results that demonstrate the degradation of performance as we decrease domain similarity (according to the method defined in chapter 5). The primary goal of the systems developed in these chapters was to work toward minimizing the reduction in performance of the the system on the "Dissimilar" and "Highly Dissimilar" domain classes presented. The results of both of these systems demonstrate that this degradation can be mitigated to an extent by using technologies better optimized for the task of transfer learning.

9.1.6 Research Question 6

"Can these novel methodologies be applied to the Motivational Interviews in order to gain new understanding of the data and apply meaningful labels?"

As discussed, the Motivational interview dataset was one of the primary sources that this work was hoping to contribute toward improving. At the start of this chapter it was demonstrated that the systems developed in this thesis are able to apply meaningful labels to the data with decent accuracy, and this could possibly be used as a first pass in order to "gain new understanding of the data". However without robustly manually labelled data to evaluate against it is hard to state conclusively whether the labels applied to the data can be considered "Meaningful". As such I believe that further work should be done in applying these tools to the motivational interview data, that includes working more closely with clinicians, and on-site data gathering. This would allow the work to progress toward more significant levels of automated assessment and triage and allow clinicians time to be utilised as effectively as possible, and to allow them to continuously learn and improve with meaningful feedback.

9.2 Recommendations for Future Work

In this thesis we have briefly mentioned a number of avenues for further study. These will now be formally presented below to summarise how the work of this thesis could be continued:

Multi-modal analysis of Motivational Interviews In the authors opinion, this is the primary route to most effectively further the work presented in this thesis. We have covered in detail methods for performing sentiment analysis on textual data, which can be applied to the MI data, even in the absence of meaningful sentiment labels. And it was demonstrated in Chapter 4 that classifying the emotional content of the speech signal directly could provide significant additional data to augment the assessing of Motivational Interviews.

Implementation of TransferGAN technique with more sophisticated encoder networks. The encoder presented in Chapter 6 is used to demonstrate the effectiveness of the GAN training methodology to improve performance over an equivalently powerful encoder. However both this baseline and the GAN improved system are outperformed by more modern encoder technologies. It is therefore a promising avenue of research to determine if the improvement in performance that the GAN demonstrated in this thesis would still occur when applying the technique to other encoder systems such as BeRT, or GPT systems.

Gaining a deeper theoretical understanding of the mechanics of adversarial learning for TL. This thesis presents the use of a GAN for transfer learning between domains. It's findings on the effectiveness of this system in Chapter 6 are inconclusive, and some potential hypotheses for the reasons behind these results are presented. This work could be extended by gaining a deeper mathematical understanding of the transformations that are occurring in the system in order to enable transfer learning. In order to robustly justify the validity or invalidity of the method for this application.

Developing a more robust grammar for GE Preprocessing In Chapter 7 we introduce a novel method of transfer learning that involves using Grammatical Evolution to develop pre-processing methodologies that prime a system for transfer learning between domains. This was proved to be effective on a simple problem using a Grammar that was defined by the author but was not supported by significant research. A further research project could test the limits of this methodology by using a more robust grammar, as well as using pairs of domains with lower similarity as training data to generate the TL rules.

Data Augmentation for MI Data In Chapter 8 we demonstrate use of the class-weighting strategy to improve classifier performance on the minority class (positive sentiment). This lead to an overall reduced performance in the system to due to higher rate of false positive predictions, reducing both the precision of the positive class, and the recall of the majority classes. A useful avenue for further study might be to explore alternative methods of data augmentation to improve performance on minority classes in the MI data without significantly impacting majority class predictions.

Appendix One

POS Tags

The full list of Part of Sentence tags, and their meanings:

Tag	Meaning
CC	conjunction, coordinating
CD	numeral, cardinal
DT	determiner
EX	existential
IN	preposition or conjunction, subordinating
JJ	adjective or numeral, ordinal
JJR	adjective, comparative
JJS	adjective, superlative
LS	list item marker
MD	modal auxiliary
NN	noun, common, singular or mass
NNP	noun, proper, singular
NNS	noun, common, plural

Tag	Meaning
PDT	pre-determiner
POS	genitive marker
PRP	pronoun, personal
PRP$	pronoun, possessive
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	particle
TO	"to" as preposition or infinitive marker
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle or gerund
VBN	verb, past participle
VBP	verb, present tense, not 3rd person singular
VBZ	verb, present tense, 3rd person singular
WDT	WH-determiner
WP	WH-pronoun
WRB	Wh-adverb

References

Aizawa, Akiko (2003). "An information-theoretic perspective of tf–idf measures". In: *Information Processing & Management* 39.1, pp. 45–65.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein generative adversarial networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 214–223.

Batliner, A, S Steidl, and E Nöth (2008). "Releasing a thoroughly annotated and processed spontaneous emotional database: the FAU Aibo Emotion Corpus". In: *Proc. of a Satellite Workshop of LREC*. Vol. 2008, p. 28.

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Blitzer, John, Mark Dredze, and Fernando Pereira (June 2007). "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 440–447. URL: https://www.aclweb.org/anthology/P07-1056.

Brill, Eric (1992). *A simple rule-based part of speech tagger*. Tech. rep. Pennsylvania Univ Philadelphia Dept of Computer and Information Science.

Brown, Tom B. et al. (2020). "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165. arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

Bu, Ke et al. (2020). "Adversarial Transfer Learning for Deep Learning Based Automatic Modulation Classification". In: *IEEE Signal Processing Letters*.

Chan, Jireh Yi-Le et al. (2023). "State of the art: a review of sentiment analysis based on sequential transfer learning". In: *Artificial Intelligence Review* 56.1, pp. 749–780.

Chawla, Nitesh V et al. (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16, pp. 321–357.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734.

Chung, Junyoung et al. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv: 1412.3555 [cs.NE]. URL: https://arxiv.org/abs/1412.3555.

Dehak, Najim et al. (2010). "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, pp. 788–798.

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL].

Dinh, Thi Thu Huong, Thi Huong Chu, and Quang Uy Nguyen (2015). "Transfer learning in genetic programming". In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1145–1151.

Donahue, Chris, Julian McAuley, and Miller Puckette (2018). "Adversarial Audio Synthesis". In: *International Conference on Learning Representations*.

Dragoni, Mauro (2019). "An evolutionary strategy for concept-based multi-domain sentiment analysis". In: *IEEE Computational Intelligence Magazine* 14.2, pp. 18–27.

Ericsson, Linus et al. (2022). "Self-supervised representation learning: Introduction, advances, and challenges". In: *IEEE Signal Processing Magazine* 39.3, pp. 42–62.

Espejo, Pedro G, Sebastián Ventura, and Francisco Herrera (2009). "A survey on the application of genetic programming to classification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40.2, pp. 121–144.

Floridi, Luciano and Massimo Chiriatti (2020). "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30, pp. 681–694.

Ghosh, Monalisha, Shashank Awasthi, and Goutam Sanyal (2019). "A Deep Learning Strategy to Sentiment Analysis on Low-volume Dataset". In: *2019 International Conference on Innovative Sustainable Computational Technologies (CISCT)*. IEEE, pp. 1–6.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems* 27, pp. 2672–2680.

Gulrajani, Ishaan et al. (2017). "Improved training of wasserstein gans". In: *Advances in neural information processing systems*, pp. 5767–5777.

Gupta, Rahul (2019). "Data augmentation for low resource sentiment analysis using generative adversarial networks". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7380–7384.

Hoang, Mickel, Oskar Alija Bihorac, and Jacobo Rouces (2019). "Aspect-based sentiment analysis using bert". In: *Proceedings of the 22nd nordic conference on computational linguistics*, pp. 187–196.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Ives, Laurel (Nov. 8, 2018). *Diabetes prescriptions now cost NHS £1bn, figures show.* https://www.bbc.co.uk/news/health-46139595, accessed 2021-01-01. (Visited on 01/01/2021).

JW, Backus (1959). "c The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference". In: *Conference, on,, Inform&on, Processing*, pp. 125–131.

Karras, Tero et al. (2018). "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*.

Kim, Yoon (2014). *Convolutional Neural Networks for Sentence Classification*. arXiv: 1408.5882 [cs.CL]. URL: https://arxiv.org/abs/1408.5882.

Koza, John R et al. (1994). *Genetic programming II*. Vol. 17. MIT press Cambridge.

Kupiec, Julian (1992). “Robust part-of-speech tagging using a hidden Markov model”. In: *Computer speech & language* 6.3, pp. 225–242.

Le, Quoc V. and Tomas Mikolov (2014). *Distributed Representations of Sentences and Documents*. arXiv: 1405.4053 [cs.CL].

Lee, Jinkyu and Ivan Tashev (2015). “High-level feature representation using recurrent neural network for speech emotion recognition”. In: *Interspeech 2015*.

Liu, Ruijun et al. (2019). “A survey of sentiment analysis based on transfer learning”. In: *IEEE Access* 7, pp. 85401–85412.

Livingstone SR, Russo FA (2018). “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English.” In: *PLoS ONE* 13(5), e0196391.

Lundahl, Brad et al. (2013). “Motivational interviewing in medical care settings: a systematic review and meta-analysis of randomized controlled trials”. In: *Patient education and counseling* 93.2, pp. 157–168.

Maas, Andrew L. et al. (June 2011). “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

Martín Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/.

Menéndez, María Luisa et al. (1997). “The jensen-shannon divergence”. In: *Journal of the Franklin Institute* 334.2, pp. 307–318.

Mikolov, Tomas et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv: 1301.3781 [cs.CL].

Mozetič, Igor, Miha Grčar, and Jasmina Smailović (2016). “Multilingual Twitter sentiment classification: The role of human annotators”. In: *PloS one* 11.5, e0155036.

Muda, Lindasalwa, Mumtaj Begam, and I. Elamvazuthi (2010). *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques*. arXiv: 1003.4083 [cs.MM]. URL: https://arxiv.org/abs/1003.4083.

Munikar, Manish, Sushil Shakya, and Aakash Shrestha (2019). "Fine-grained sentiment classification using bert". In: *2019 Artificial Intelligence for Transforming Business and Society (AITB)*. Vol. 1. IEEE, pp. 1–5.

Muñoz, Luis, Leonardo Trujillo, and Sara Silva (2020). "Transfer learning in constructive induction with genetic programming". In: *Genetic Programming and Evolvable Machines* 21.4, pp. 529–569.

Ni, Jianmo, Jiacheng Li, and Julian McAuley (2019). "Justifying recommendations using distantly-labeled reviews and fine-grained aspects". In: *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434*.

Ratnaparkhi, Adwait (1996). "A maximum entropy model for part-of-speech tagging". In: *Conference on empirical methods in natural language processing*.

Ringeval, Fabien et al. (2013). "Introducing the RECOLA multimodal corpus of remote collaborative and affective interactions". In: *2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG)*. IEEE, pp. 1–8.

Rosenthal, Sara, Noura Farra, and Preslav Nakov (Aug. 2017). "SemEval-2017 Task 4: Sentiment Analysis in Twitter". In: *Proceedings of the 11th International Workshop on*

Semantic Evaluation (SemEval-2017). Vancouver, Canada: Association for Computational Linguistics, pp. 502–518. DOI: 10.18653/v1/S17-2088. URL: https://www.aclweb.org/anthology/S17-2088.

Russel, James A. (1980). “A Circumplex Model of Affect”. In: *Journal of Personality and Social Psychology* 39(6), pp. 1161–1178.

Ryan, Conor, J. J. Collins, and Michael O’Neill (14-15 4 1998). “Grammatical Evolution: Evolving Programs for an Arbitrary Language”. In: *Proceedings of the First European Workshop on Genetic Programming*. Ed. by Wolfgang Banzhaf et al. Vol. 1391. LNCS. Paris: Springer-Verlag, pp. 83–96. ISBN: 3-540-64360-5. DOI: doi:10.1007/BFb0055930. URL: http://citeseer.ist.psu.edu/ryan98grammatical.html.

Seki, Hiroyuki et al. (1991). “On multiple context-free grammars”. In: *Theoretical Computer Science* 88.2, pp. 191–229.

Soderlund, Patricia Davern (2018). “Effectiveness of motivational interviewing for improving physical activity self-management for adults with type 2 diabetes: A review”. In: *Chronic Illness* 14.1, pp. 54–68.

Straka, Milan and Jana Straková (2017). “Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe”. In: *Proceedings of the CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies*, pp. 88–99.

Sun, Xiaofei et al. (2023). “Sentiment analysis through llm negotiations”. In: *arXiv preprint arXiv:2311.01876*.

Trigeorgis, George et al. (2016). “Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network”. In: *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5200–5204.

Wang, Jindong et al. (2022). “Generalizing to unseen domains: A survey on domain generalization”. In: *IEEE transactions on knowledge and data engineering* 35.8, pp. 8052–8072.

Wei, Xizi (2022). "Automatic assessment of motivational interview with diabetes patients". PhD thesis. University of Birmingham.

Wu, Jia et al. (2019). "Hyperparameter optimization for machine learning models based on Bayesian optimization". In: *Journal of Electronic Science and Technology* 17.1, pp. 26–40.

Yang, Zichao et al. (June 2016). "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://www.aclweb.org/anthology/N16-1174.

Yuan, Zhigang et al. (2018). "Domain attention model for multi-domain sentiment classification". In: *Knowledge-Based Systems* 155, pp. 1–10.

Zhang, Yuebing, Duoqian Miao, and Jiaqi Wang (2019). *Hierarchical Attention Generative Adversarial Networks for Cross-domain Sentiment Classification*. arXiv: 1903.11334 [cs.LG].