DOMAIN ADAPTATION APPLICATIONS TO COMPLEX HIGH-DIMENSIONAL TARGET DATA

A Dissertation Proposal
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by
Marija Stanojevic
May, 2023

Examining committee members:

Dr. Zoran Obradovic, Advisory Chair, Department of Computer & Information Sciences
Dr. Eduard Dragut, Department of Computer & Information Sciences
Dr. Slobodan Vucetic, Department of Computer & Information Sciences
Dr. Sudhir Kumar, Department of Biology, Institute for Genomics and Evolutionary Medicine

©
Copyright
2023

by

Marija Stanojevic

All Rights Reserved

ABSTRACT

In the last decade, machine learning models have increased in size and amount of data they are using, which has led to improved performance on many tasks. Most notably, there has been a significant development in end-to-end deep learning and reinforcement learning models with new learning algorithms and architectures proposed frequently. Furthermore, while previous methods were focused on supervised learning, in the last five years, many models were proposed that learn in semi-supervised or self-supervised ways. The model is then fine-tuned to a specific task or different data domain. Adapting machine learning models learned on one type of data to similar but different data is called domain adaptation. This thesis discusses various challenges in the domain adaptation of machine learning models to specific tasks and real-world applications and proposes solutions for those challenges.

Data in real-world applications have different properties than clean machine-learning datasets commonly used for the experimental evaluation of proposed models. Learning appropriate representations from high-dimensional complex data with internal dependencies is arduous due to the curse of dimensionality and spurious correlation. However, most real-world data have these properties in addition to a small number of labeled samples since labeling is expensive and tedious. Additionally, accuracy drops drastically if models are applied to domain-specific datasets and unbalanced problems. Moreover, state-of-the-art models are not able to handle missing data. In this thesis, I strive to create frameworks that can learn a good representation of high-dimensional small data with correlations between variables.

The first chapter of this thesis describes the motivation, background, and research objectives. It also gives an overview of contributions and publications. A background needed to understand this thesis is provided in the second chapter and an introduction to domain adaptation is described in chapter three. The fourth chapter discusses domain adaptation with small target data. It describes the algorithm for semi-supervised learning over domain-specific short texts such as reviews or tweets. The proposed framework achieves up to 12.6% improvement when only 5000 labeled examples are available. The fifth chapter explores the influence of unanticipated bias in fine-tuning data. This chapter outlines how the bias in news data influences the classification performance of domain-specific text, where the domain is U.S. politics. It is shown that fine-tuning with domain-specific data is not always beneficial, especially if bias towards one label is present. The sixth chapter examines domain adaptation on datasets with high missing rates. It reviews a system created to learn from high-dimensional small data from psychological studies, which have up to 70% missingness. The proposed framework is achieving 9.3% smaller imputation and 33% lower prediction errors. The seventh chapter discusses the curse of dimensionality problem in domain adaptation. It presents a methodology for discovering research articles containing evolutionary timetrees. That system can search for, download, and filter research articles in which timetrees are imported. It scans 5 million articles in a few days. The proposed method also decreases the error of finding research papers by 21% compared to the baseline, which cannot work with high-dimensional data properly. The last, eighth chapter, summarizes the findings of this thesis and suggests future prospects.

To my mother Snezana, my sister Ana, Robert, and all family and friends who supported me through my journey.

In loving memory of my father Goran and my grandparents.

ACKNOWLEDGEMENTS

I want to express my immense gratitude to my advisor, prof. Zoran Obradovic, for giving me the opportunity to join his laboratory and for his guidance, advice, and support. His mentoring approach, driven by rigorous examination and collaborative initiative for high-quality work, was essential to conducting the research throughout my Ph.D. and completing my dissertation. I am grateful to Temple University and prof. Obradovic for funding my research and education throughout my graduate studies.

I am deeply grateful to prof. Sudhir Kumar for his invaluable advice, supervision, and contribution to the work described in Chapter 6 of this thesis. I am beholden to prof. Eduard Draguut for his helpful counsel, unwavering support, and constant encouragement throughout my graduate studies.

I am particularly thankful to professors Slobodan Vucetic, Eduard Dragut, and Sudhir Kumar for serving on the committee for my preliminary exams and my dissertation committee. Their feedback and insights significantly contributed to the improvement of my dissertation.

I want to thank the chair of the Department of Computer & Information Sciences, prof. Jamie Payton for her support and participation in many outreach activities I was involved with. Many thanks to the department's professors for creating a friendly and stimulating academic environment and delivering various courses to learn from. I want to thank prof. Rose McGinnis for advice about writing an appropriate resume and finding internships. I am also grateful for her support in organizing the 9th Mid-Atlantic Student Colloquium

on Speech, Language, and Learning (MASC-SLL 2022). Finally, my gratitude goes to the department and especially to the committee of professors who awarded me the Graduate Outstanding Teaching Assistant award in 2022.

Special thanks to the department's staff members Marah Minetola and Michelle Rambo for their invaluable help with the organization of the 9th Mid-Atlantic Student Colloquium on Speech, Language and Learning (MASC-SLL 2022). I am also grateful to the department's staff Julie Skrocki, Thomas Stauffer, Andrea McGady, Charles T. Rorke, and Kalee Marshall, for their time and assistance during my graduate studies. Finally, I want to commend Michele Young, the Executive Assistant to the Dean of the College of Science and Technology, for her support and help in organizing outreach activities, especially Research Mixer, and her whole family for making me feel a bit more at home during my studies.

My deep gratitude goes to Temple University for awarding me with a presidential fellowship that partially funded my research during my Ph.D. studies. I am also thankful to Barbara Fles, Director of special events at Temple University, for her help with organizing many events and activities I was involved with during my Ph.D.

I genuinely appreciate my dear friend and labmate Jumanah Alshehri for her companionship and the scientific and outreach work we have done during the Ph.D. I want to thank my previous flatmates, Tanaya Roy, Jovan Andjelkovic, and Daniel Saranovic, for their companionship and support. I was lucky to have many other friends during my graduate studies. I am very thankful to Saman, Sidra, Abrar, Alkhansa, Somayih, Hannah, Miu, Rafaa, Sarah, Shelley, Ameen, Francisco, and Avik.

My Ph.D. journey would not have been the same without my fellow labmates, visiting scholars, and postdocs at the Center for Data Analytics and Biomedical Informatics, listed here in alphabetical order: Ameen Abdel Hai, Andy Gnias, Benjamin Rapp, dr Branimir Ljubic, dr Branko Arsic, Brian Davis, dr Chao Han, Daniel Saranovic, dr Djordje Gligorijevic, Duncan Copeland, dr Dusan Ramljak, dr Fang Zhou, Hannah Kim, Hussain Otudi, dr Ivan Stojkovic, dr Jelena Gligorijevic, Jennifer Chao, dr Jesse Glass, Jovan And-

jelkovic, Jumanah Alshehri, dr Martin Pavlovski, dr Milan Basic, Mohammad Alqudah, dr Mohamed Ghalwash, dr Nima Asadi, Noor Albarakati, Nouf Albarakati, Parisa Khan, Quang Pham, Rafaa Aljurbua, Scott Glascott, Shelley Biagas, dr Shoumik Roychoudhury, Thomas John Mirowski, Wilson Diaz and dr Xi Hang Cao; many of which I collaborated and published joint papers with. I would also like to thank my external collaborators: Adrienne Kasprowicz, Alexandra Bhatti, Alison Fisher, Alison Thaung, Ashis Kumar Chanda, prof. Blair S. Hedges, Cindi Knighton, Jason Brinkley, Knighton Cindi, Lesley A. Norris, Louise A. Huuki, Pam Srivastava, prof. Philip C Kendall, Sarah Ball, Stacie Greby, Yoonjae Kang, and William Campbell.

During my Ph.D. studies, I had the opportunity of participating in several research projects. In that regard, I would like to acknowledge the support of the National Science Foundation, U.S. Army Research Laboratory, U.S. Army Corp of Engineers Engineer Research and Development Center, Temple University, National Institutes of Health, IQVIA, King Abdullah University of Science and Technology Center Partnership Fund, Pennsylvania Department of Health, and Center for Disease Control.

I am thankful to my current boss Jekaterina Novikova, and Winterlight Labs for hiring me before the end of my Ph.D., for the excellent working and research environment, and for support and understanding during my immigration to Canada and during the finalization of my Ph.D. Finally, I am genuinely grateful to all my colleagues, especially my teammates Malikeh and Ali.

During my Ph.D. program, I joined multiple companies for summer internships, Alliance Data (2018), Facebook (2019, 2020), and LinkedIn (2021), during which I was privileged to work with a number of terrific researchers and engineers that influenced the way I thought about approaching real-world problems. I also learned a lot about working in large corporations and better understood the future perspectives. Particularly, I am tremendously thankful to my managers, Glenn Strycker, Boaz Super, Daniel Li, and Mahesh Joshi, and mentors Zachary Kann, Hailing Cheng, Chunfei Li, and Kai Yang, for being so welcoming

and supportive. Special thanks to Qiaochu Yuan for her valuable input during my second internship at Facebook.

I am also thankful to the organizers of ICML 2020, NeurIPS 2020, ACL 2020, ICML 2020, and ICLR 2020 for involving me in the organization of those conferences. As a result, I had the opportunity to meet some of the best ML scientists, discuss excellent machine learning research, and grow my network. Furthermore, I am incredibly grateful for the chance to be a virtual chair of ICLR 2021 and ICML 2021, where I led the global team of volunteers and learned a lot about the organization of big conferences. Special thanks to Lee Campbell for the great collaboration on all those events and to prof. Kyunghyun Cho for his guidance during the organization of ICLR 2020. I am also grateful to professors Mark Dredze and Hal Daumé III for their advice during the organization of the 9th Mid-Atlantic Student Colloquium on Speech, Language and Learning (MASC-SLL 2022).

I started my Ph.D. studies while finishing the master's program, and I am tremendously grateful to my master's thesis mentor, prof. Zeljko Djurovicm, other professors, and assistants of my master's program for support, understanding, and flexibility, which helped me finish my master's studies.

Finally, I would like to express my sincere gratitude to my parents, Snezana and Goran, my sister Ana, my boyfriend, and the rest of my family and friends for their continued love, encouragement, and support.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS vi

LIST OF TABLES xvi

LIST OF FIGURES xvii

LIST OF ABBREVIATIONS AND SYMBOLS xix

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Thesis Objectives 3

1.3 Methodological Contributions and Practical Implications 6

1.4 Dissertation Outline 7

1.5 Published work 9

2 BACKGROUND 13

2.1 Machine Learning 13

2.1.1 Linear Regression 15

2.1.2 Logistic Regression 16

2.1.3 Common Cost Functions 17

2.1.4 Gradient Descent 19

2.1.5 Bias-variance Tradeoff 20

2.1.6 Regularization 23

2.1.7 Evaluation Metrics 24

2.2 Effective Strategies for Data Handling in Machine Learning 28
2.2.1 Missing Data and Imputation Techniques 29
2.2.2 Strategies for Achieving Dataset Balance in Machine Learning . . . 31
2.2.3 Uncovering and Addressing Bias in Data 33
2.2.4 Curse of dimensionality 35
2.3 Neural Networks 36
2.3.1 Important Models 38
2.4 Transfer learning 43
2.4.1 Pretraining 46
2.4.2 Lifelong Learning 48
2.4.3 Meta-learning 48
2.4.4 Zero-shot and Few-shot Learning 49
2.4.5 Multi-task Learning 50
2.4.6 Sequential Transfer Learning 51
2.5 Natural Language Processing 51
2.5.1 Zipf Law 56
2.5.2 Traditional Word Representations 56
2.5.3 Word Embeddings 59
2.5.4 Deep Pretrained Representations 63
2.5.5 Universal Language Model Fine-tuning for Text Classification . . . 69
2.5.6 BERT and RoBERTa 72
2.5.7 GPT-2, GPT-3, and GPT-chat 77
2.6 Machine learning for medicine and healthcare 80
2.6.1 Machine Learning for Cognitive and Mental Health 81
2.7 Machine Learning for Biology 82
2.8 Conclusion 85

3 INTRODUCTION TO DOMAIN ADAPTATION 86

4 ENHANCING PERFORMANCE ON TARGET DATA WITH LIMITED SAMPLES THROUGH DOMAIN ADAPTATION 94

4.1 Introduction 95

4.2 Related Work 97

4.3 Methodology 99

4.3.1 Seed Labeling 99

4.3.2 Semi-supervised Label Prediction (SLP) 100

4.3.3 Handling very small seeds 102

4.3.4 Exit criteria 102

4.3.5 Streaming 103

4.3.6 Model testing 103

4.4 Experimental Evaluation 103

4.4.1 Experimental Environment 104

4.4.2 Datasets 104

4.4.3 Twitter guns advocacy dataset 105

4.4.4 Parameters settings 106

4.4.5 Models used for training step 107

4.4.6 Baseline models 108

4.4.7 Time performance 108

4.5 Results and Discussion 108

4.6 Summary 112

5 EXPLORING THE IMPACT OF UNFORESEEN BIAS ON DOMAIN ADAPTATION 114

5.1 Introduction 115

5.2 Related Work 117

5.3 Methodology 118
5.4 Experimental Evaluation 119
5.4.1 Experimental settings 120
5.5 Results and discussion 121
5.6 Summary 124
6 IMPROVING DOMAIN ADAPTATION PERFORMANCE ON DATASETS WITH HIGH MISSING DATA RATES 125
6.1 Introduction 126
6.2 Related work 128
6.2.1 Imputation state of the art 128
6.2.2 Machine learning in psychological studies 130
6.3 Methods 132
6.3.1 Imputation methods 132
6.3.2 Prediction methods 134
6.3.3 Joining multiple studies 134
6.3.4 Domain adaptation 135
6.4 Experimental Evaluation 135
6.4.1 Data 135
6.4.2 Baseline methods 137
6.4.3 Dataset Cleaning 138
6.4.4 Imputation evaluation 138
6.4.5 Comparison of the imputated datasets on prediction task 138
6.4.6 Domain adaptation task 139
6.5 Results 140
6.5.1 Imputation results 141
6.5.2 Performance comparison between joined dataset and single studies . 141

6.5.3 Domain adaptation - withdrawn patients 144

6.6 Summary 145

7 MITIGATING THE CURSE OF DIMENSIONALITY IN DOMAIN ADAPTATION 146

7.1 Introduction 147

7.2 Related Work 149

7.3 Methodology 150

7.3.1 The ground truth TimeTree (TT) dataset 150

7.3.2 Processing the corpus of scientific literature 150

7.3.3 Representation learning 156

7.3.4 Representation enhancing 156

7.3.5 Classification 157

7.3.6 Evaluation 157

7.3.7 Generation of text excerpts datasets 158

7.3.8 TimeTreeFinder (TTF) tool 159

7.4 Experimental Evaluation 159

7.4.1 Experimental Settings 160

7.5 Results and Discussion 161

7.5.1 Most common words in the TimeTree Dataset 162

7.5.2 Properties of collected and filtered datasets 163

7.5.3 Classification models and their predictions 164

7.5.4 Improving predictions using text excerpts 166

7.5.5 Classification results and discussion 169

7.5.6 The proportion of additional timetree articles predicted 172

7.6 Summary 173

8 CONCLUSION 175

8.1 Future Directions 178

BIBLIOGRAPHY 182

LIST OF TABLES

4.1 Large-scale text classification datasets 104

4.2 The label prediction accuracy (LP) was evaluated for a seed size of 5,000, and the best ratio for achieving the highest LP for each proposed model is indicated in parentheses. 108

4.3 The accuracy of label prediction (LP) is reported for a seed size of 50,000, with the best ratio for each value of the proposed model indicated in parentheses. 109

5.1 Outlets 120

5.2 Classification results 121

6.1 Statistical overview of youth anxiety data. Data comes from studies S1-S9 in the order referenced in introduction. 136

6.2 Imputation evaluated on joined dataset (Full RMSE) and average RMSE of imputation on separate studies (Avg study RMSE) 140

6.3 Performance of imputation (Bagging with ElasticNet) and prediction (Bayesian Regression) on imputed data for each study. 142

7.1 Results from analysis. Average F1 scores and their standard are shown. . . 170

7.2 Results from analysis. Average F1 scores and their standard are shown when models are trained with enhanced representation. 171

LIST OF FIGURES

2.1 Recurrent neural network architecture: signal flow (left) and unfolded view (right) (Goodfellow et al. (2016)) 39

2.2 LSTM cell (Graves et al. (2013)) 41

2.3 Transformers architecture (Vaswani et al. (2017)) 42

2.4 Transfer learning approach (Ruder (2019)) 44

2.5 Transfer learning categorization (Ruder (2019)) 45

2.6 Architectuures of BERT, GPT, and ELMO (Devlin et al. (2019)) 72

2.7 BERT input representation (Devlin et al. (2019)) 75

2.8 BERT fine-tuning (Devlin et al. (2019)) 75

4.1 Architecture of the proposed semi-supervised label prediction (SLP) framework. 101

4.2 The label prediction accuracy of the SLP model was evaluated for the Yelp Review Polarity dataset using multiple language modeling algorithms and various ratios for two different seed sizes: (a) 5,000 and (b) 50,000. 111

5.1 ULMFiT model training-flow overview 115

5.2 ULMFiT model training details 116

5.3 Percentage of predicted labels for each class when the model is fine-tuned with ten different combinations of news outlet texts. The chart on the left (a) corresponds to the balanced dataset, while the chart on the right (b) corresponds to the unbalanced dataset. 122

6.1 This figure displays the percentage of missing features per patient, where a total of 108 features were considered. The patients are separated by vertical lines to indicate the different studies they belong to. 127

6.2 RMSE of the best, an average of all predictors, and worst over different imputation methods on test data. **Prediction full** lines describe prediction made on a joined dataset, and **prediction avg study** lines show average prediction RMSE when a different model is trained on each study. 143

6.3 RMSE of the best, an average of all predictors, and the worst predictor over different imputation methods on data from withdrawn patients. Lines have the same meaning as in Figure 6.2. 144

7.1 Overview of the proposed TimeTreeFinder (TTF) system, including searching, collecting, and analyzing research articles to discover time tree-containing articles. 151

7.2 Frequencies of most common words in (a) TimeTree Dataset, Google Scholar Datasets from (b) FL-simple query and (c) FL-complex query, (d) PMC-OA, (e) PMC Historical, and (f) bioRxiv dataset. 162

7.3 Comparison of the distributions of the most common words among papers labeled as a) positive and b) negative in TimeTree Dataset. There is quite a bit of overlap between the two sets. 165

7.4 Scatter plot of words from articles in original datasets. The horizontal axis represents word frequency in negatively labeled documents and the vertical in positively labeled. The top positive and negative words are displayed on the right. 165

7.5 Comparison of distributions of most common words between whole texts (left) and selected texts (right) from two research papers datasets: original (top) and bioRxiv (bottom). 166

7.6 Scatter plot of words in selected texts from articles in original datasets. The horizontal axis repressents word frequency in negative documents and the vertical in positive. The top positive and negative words are displayed on the right. 167

7.7 Comparison of projected 100-dimensional embeddings of words in contexts of words mega (left), and tree (right) learned on the whole papers (top) and on selected texts (bottom). 168

7.8 The precision per year of BERT, trained on 800-character excerpts and evaluated on articles from Molecular Phylogenetics and Evolution, Systematic Biology, and Zoologica Scripta, reveals that some articles containing information about timetrees are not present in the TimeTree database. . . . 172

List of Abbreviations and Symbols

Symbols

In this thesis, we adopt the following conventions: Vectors are represented by bold lowercase letters ($\boldsymbol{x}$), matrices are represented by bold uppercase letters ($\boldsymbol{X}$), and scalars are represented by regular weight letters (x). Subscripts with bold letters ($\boldsymbol{x_i}$) are used to refer to entire rows or columns, while subscripts with regular weight letters (x_{ij}) refer to specific elements. A feature space $\mathbb{X}$, a domain $\mathbb{D}$, and a task $\mathbb{T}$ are represented with blackboard bold letters.

P	Probability mass function
p	Probability density function
$\hat{P}$	Empirical distribution function
σ	Standard deviation
σ^2	Variance
ϱ	Pearson's r
N	Gaussian distribution
η	Learning rate
$\mathbb{T}$	Task
$\mathbb{D}$	Domain
$\boldsymbol{W}$	Weight matrix
$\boldsymbol{X}$	Set of training examples

$\boldsymbol{y}$	Set of training labels
$\boldsymbol{\theta}$	Model parameters
J	Cost function
L	Loss function

Abbreviations

e.g.	for example
i.e.	that is
et al.	and others
eq.	equation
KL	Kullback-Leibler
JS	Jensen-Shannon
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
CV	Computer Vision
i.i.d.	independent and identically distributed
EM	Expectation Maximization
LASSO	Least Absolute Shrinkage and Selection Operator
ML	Machine Learning
SSL	Semi-Supervised Learning
SGD	Stochastic Gradient Descent
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MBE	Mean Bias Error
MAPE	Mean Absolute Percentage Error

ROC	Receiver Operating Characteristic
AUC-ROC	Area under the ROC Curve
SVM	Support Vector Machine
HMM	Hidden Markov Model
LDA	Latent Dirichlet Allocation
FFNN	Feed-forward Neural Network
DNN	Deep Neural Network
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Network
BPTT	Back-Propagation Through Time
NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
tf-idf	Term Frequency-inverse Document Frequency
GloVe	Global Vectors for Word Representation
BOW	Bag-Of-Words
CBOW	Continuous Bag-Of-Words
SGNS	Skip-Gram with Negative Sampling
LM	Language Model
LSA	Latent Semantic Analysis
SRL	Semantic Role Labelling
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Pretraining Approach
GPT-2	Generative Pre-trained Transformer 2
GPT-3	Generative Pre-trained Transformer 3

CHAPTER 1

INTRODUCTION

1.1 Motivation

Over the past decade, deep neural networks (DNN) (Goodfellow et al. (2016)) have become the preferred method for learning from data. Their success is caused by their ability to automatically learn a hierarchy of features in an end-to-end manner, reducing the need for manual feature engineering.

The standard approach to train these models is supervised learning, where many annotated examples are provided to the model, which then learns to map inputs to outputs. However, this approach can become infeasible in real-world scenarios where annotating examples for every language, task, and domain is impossible. In addition, acquiring labels is a time-consuming and expensive process.

Semi-supervised and self-supervised pretrained models have advanced in the last five years in machine learning. They aim to improve the efficiency and performance of training models. Semi-supervised learning involves using a small amount of labeled data and a large amount of unlabeled data to train a model. Self-supervised learning, on the other hand, uses only unlabeled data and involves designing tasks that can be solved using the data without labeling them, allowing the model to learn valuable representations. Pretraining refers to

training a model on a large amount of general data before training it on specific target task data (fine-tuning). Semi-supervised and self-supervised pre-trained models have proven more effective than traditional supervised learning methods and can be applied to a wide range of tasks.

The goal of transfer learning is to overcome the inadequacies of supervised learning by transferring knowledge from similar domains, tasks, and languages in the source environment to the target environment. Transfer learning has been a crucial aspect of machine learning systems, with many critical advancements like latent semantic analysis (Deerwester et al. (1990)), Brown clusters (Brown et al. (1992)), pretrained word embeddings (Mikolov et al. (2013b)), and pretrained language models (Devlin et al. (2019); Liu et al. (2019); Brown et al. (2020)).

Domain adaptation is a subset of transfer learning methodologies concerned with problems where source and target tasks are the same, but labeled data are available only for source data, and domains of source and target data are different. In real-world scenarios, models trained on the source domain may perform poorly when applied to the target domain, leading to reduced accuracy and limited usefulness. To overcome this, domain adaptation techniques aim to adapt the model to the target domain by transferring the knowledge learned from the source domain to the target domain. These techniques enable the model to perform well on the target domain, even if the data distribution differs. The motivation behind domain adaptation is to improve machine learning models' robustness and generalization ability in real-world applications, where data distribution can vary greatly.

This dissertation proposes using domain adaptation techniques to learn better representations in domains with small (few-shot learning) or no label data (zero-shot learning). We argue that learning through domain adaptation is better than supervised learning because it allows to learning representations of domains and tasks for which it is impossible to get all the labels. We explore the domain adaptation learning process in NLP, medical and biological applications, which all contain different challenges, such as small target data,

imbalanced labels in source and target datasets, different distributions of labels between source and target datasets, high missing rates, and a high number of features which is usually combined with a small number of samples. Our systems show superior performance compared to existing methods.

1.2 Thesis Objectives

This thesis examines the issue of producing representations and predictions for NLP, medical and biological data using machine learning methods that can be transferred across tasks and domains. Neural networks (NNs) take advantage of end-to-end learning and existing related information from domains and tasks and perform better than models that do not utilize this information across various tasks. In most of the thesis, we are focusing on NNs, and deep learning (DL) based methods, but in some cases, it was not possible to utilize DL due to the small amount of data in both source and target domains.

We claim that domain adaptation is superior to supervised learning in most cases, with the following exceptions:

1. Target domain already has a sufficient number of labeled training examples available.

2. There is no relevant data source that can be utilized to improve learning on the target data.

To address the first aspect, we analyze the zero-shot and few-shot capabilities of domain adaptation models, and we experimentally evaluate other models to understand the smallest number of labeled target samples needed to achieve good performance. The second point hints at a recurring theme in this thesis: The success of domain adaptation is dependent on the similarity of the source to the target, including bias of source and target, the similarity between labels and features distribution, and missing rate.

The proposed approaches in this thesis will target six specific objectives:

1. **Optimizing source data modeling to enhance performance on target data**: It is essential to carefully model the source data to ensure that it captures the essential features of the target domain. This can be achieved through various techniques such as fine-tuning pretrained models on the target domain or using generative learning. The optimization of source data modeling can also be done through unsupervised learning methods such as clustering or deep unsupervised learning. Our approach involves developing a technique to extract maximum information from source data to achieve superior performance on target data while using minimal labeled examples. We investigate various learning methodologies and a required number of labeled samples to learn the best representation. We evaluate these methods on NLP tasks, specifically the classification of short personal opinion texts (Chapter 4).

2. **Analyzing the effects of unexpected bias on domain adaptation**: Most current methods are only effective under conditions where the source and target settings exhibit similarity. Ideally, the method should overcome discrepancies between the source and target settings. In this objective, we explore the hidden biases of the fine-tuning datasets and show that they can deteriorate the performance of the model on target data instead of helping the domain adaptation. We use news and social media data on the US elections to test this behavior and analyze the settings influencing domain adaptation success (Chapter 5).

3. **Investigating the impact of imbalanced class distribution on domain adaptation performance**: One of the challenges faced in domain adaptation is the issue of imbalanced class distribution, where the frequency of certain classes in the target domain may be significantly different from that of the source domain. This can lead to a degradation in the performance of the adapted model, as it may be biased toward predicting the majority class. We investigate how pretrained DNN models handle class imbalance between fine-tuning, target news, and social media data on

US elections (Chapter 5).

4. **Enhancing domain adaptation performance on datasets with a high rate of missing data**: In real-world applications, especially in medical data, missingness is a common challenge that can significantly impact the performance and availability of predictive models. When both source and target domains contain a high rate of missing data, it can be difficult to accurately adapt the model trained on the source domain to the target domain. To enhance domain adaptation performance in this case, we suggest imputing the missing data with iterative imputer, where in each step we use an ensemble of machine learning models to improve imputation. We also suggest combining data from multiple studies into one source in order to achieve better performance in the target domain. We apply these techniques to the youth anxiety clinical dataset (Chapter 6).

5. **Understanding influence of multiple data sources in domain adaptation**: One of the key challenges in domain adaptation is dealing with biases between the source and target domains. When adapting a model to a target domain composed of data from multiple sources, understanding and mitigating biases becomes even more complex. These biases can arise from a variety of factors, including differences in data collection methods, cultural and social differences, and biases in the algorithms used to collect or label the data. By improving our understanding of the biases of multiple data sources in domain adaptation, we can develop more effective methods for adapting models to new domains and improving the accuracy of predictive models in real-world applications. As mentioned in the previous objective, we combined data from different studies into one source. Then, we investigated the difference between domain transfer with a single study and a joint studies of anxiety clinical source dataset (Chapter 6).

6. **Alleviating the curse of dimensionality in domain adaptation**: In domain adaptation, curse of dimensionality can be particularly challenging, as the source and target domains may have different feature spaces, leading to a further increase in dimensionality. This is the case in our application where scientific phylogenetics articles are used as a source and biology articles are used as a target. Given that words and phrases are input features, those differ between source and target domain. We examine this difference, and its influence on domain adaptation performance. We propose a method to alleviate the curse of dimensionality and drastically improve the performance of the domain adaptation (Chapter 7).

1.3 Methodological Contributions and Practical Implications

This thesis investigates various challenges in domain adaptation and proposes solutions for these challenges. The contributions of this thesis are as follows:

- **Chapter 4:** This chapter proposes an algorithm for semi-supervised domain adaptation of machine learning models on domain-specific short texts such as reviews or tweets. The proposed framework achieves up to 12.6% improvement in accuracy when only 5000 labeled examples are available. This contribution is significant as it allows for better adaptation of models to domains with a small number of labeled samples.

- **Chapter 5:** This chapter explores the impact of unanticipated bias on domain adaptation performance. It outlines how the bias in news data influences the classification performance of domain-specific text, where the domain is U.S. politics. The findings suggest that fine-tuning with domain-specific data is not always beneficial, especially if bias towards one label is present. In addition, imbalanced data can further hurt the learning of properties of smaller categories. This contribution is important as it highlights the need for bias mitigation and dataset balancing techniques in domain

adaptation with pretrained models.

- **Chapter 6:** This chapter proposes a framework for domain adaptation on datasets with high missing rates. The framework is designed to learn from high-dimensional small data from clinical anxiety studies, which have up to 70% missingness. The proposed framework achieves 9.3% smaller imputation and 33% lower prediction errors. This contribution is significant as it allows for better adaptation of models to datasets with high missing rates, which are common in real-world applications, especially in medical datasets.

- **Chapter 7:** This chapter addresses the curse of dimensionality problem in domain adaptation. It presents a methodology for discovering research articles containing evolutionary timetrees. The proposed system can search for, download, and filter research articles in which timetrees are imported, scanning 5 million articles in a few days. The proposed method also decreases the error of finding research papers by 21% compared to the baseline, which cannot work with high-dimensional data properly. This contribution is significant as it allows for better learning from large documents, such as news articles, research articles, reports, and books.

In summary, this thesis makes significant contributions to the field of domain adaptation, addressing various challenges such as small labeled samples, unanticipated bias, missing data, and high-dimensional data. The proposed solutions are shown to improve the performance of machine learning models in real-world applications. Future research can build upon these contributions to address other challenges in domain adaptation and related fields.

1.4 Dissertation Outline

Chapter 2 of this thesis presents a detailed summary of the foundational knowledge required to comprehend the material covered in this work. This includes reviewing the fundamentals

of machine learning, neural networks, and transfer learning. We discuss relevant methods and tasks that can help the reader form the basic knowledge for understanding other chapters of this thesis. Finally, we discuss three application fields focusing on subtopics relevant to this thesis:

1. natural language processing;
2. machine learning for medicine and healthcare; and
3. machine learning for biology and bioinformatics.

In Chapter 3, we delve into the concept of domain adaptation and we discuss the details of existing methods and challenges. The following chapters of this thesis are dedicated to exploring each of these challenges within the particular application in depth. Here, we present new methods for each scenario that outperform the application's current state-of-the-art models.

Chapter 4 discusses our research on domain adaptation with small target data. To examine this challenge, we work with domain-specific social media target datasets to survey public opinion from big social media posts. Strategies for labeling posts relevant to a topic are presented. A two-part framework is proposed in which semi-automatic labeling is applied to a small subset of posts, referred to as the "seed". This seed is used as bases for semi-supervised labeling of the rest of the data.

In chapter 5, we investigate the influence of fine-tuning pretrained models with biased domain-specific news data with the goal of predicting a label for Twitter data within the same domain. We test datasets with both balanced and unbalanced labels. Experimental evaluation demonstrates that tuning with biased news data of different properties changes the classification accuracy up to 9.5%. The experimental studies reveal that the characteristics of the text of the tuning dataset, such as bias, vocabulary diversity, and writing style, are essential for the final classification results, while the size of the data is less consequential.

Chapter 6 presents a study on domain adaptation for datasets with numerous features and limited samples, where a high percentage of data is missing. This is a typical scenario in many medical and mental health studies. We propose using machine learning-based imputation to analyze youth anxiety data with missing rates up to 80%. Our evaluation found that the proposed iterative imputation method using a bag of elastic net regressions outperforms traditional imputation techniques and yields the best prediction results. In the case of adaptation to a dataset of withdrawn patients, we observed a 50% improvement in results when all available studies were used to impute and train the model.

Chapter 7 discusses domain adaptation on datasets with high-dimensional inputs, where pretrained models and simple fine-tuning are not effective. Our study analyzed datasets of phylogenetics research papers with the aim of identifying articles that contain timetrees. Our evaluation showed that fine-tuning BERT for text classification on the whole article resulted in an F1 score of 0.67, which we improved to 0.88 by text-mining the article excerpts that mention figures. This new method is implemented in the TimeTreeFinder (TTF) tool, which automatically processes millions of articles to find those that contain timetrees. The manual evaluation showed that the precision of the tool on newly published out-of-distribution articles is 87%.

Chapter 8, the final chapter, summarizes the results of our research and provides an outlook on what the future may hold.

1.5 Published work

The main sources of information for this dissertation are peer-reviewed articles and a book chapter, which are listed in order of publication below:

Stanojevic, M., Alshehri, J., and Obradovic, Z. (2019), "Surveying public opinion using label prediction on social media data," in Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 188–195.

Stanojevic, M., Alshehri, J., Dragut, E., and Obradovic, Z. (2019), "Biased news data

influence on classifying social media posts," in 3rd Int'l Workshop on Recent Trends in News Information Retrieval (NewsIR 2019).

Stanojevic, M., Alshehri, J., and Obradovic, Z. (2021), "High performance computing for understanding natural language," in Handbook of Research on Methodologies and Applications of Supercomputing, pp. 133–144, IGI Global.

Stanojevic, M., Norris, L. A., Kendall, P. C., and Obradovic, Z. (2022), "Predicting Anxiety Treatment Outcomes with Machine Learning," in The 20th IEEE International Conference on Machine Learning and Applications (ICMLA).

Stanojevic, M., Andjelkovic, J., Kasprowicz, A., Huuki, L. A., Chao, J., Hedges, S. B., Kumar, S., and Obradovic, Z. (2023), "Discovering research articles containing evolutionary timetrees by machine learning," Bioinformatics

The articles listed below, arranged in the order of their publication, are relevant to the research topic of this thesis but will not be extensively discussed:

Alshehri, J., **Stanojevic, M.**, Dragut, E., and Obradovic, Z. (2021), "Stay on Topic, Please: Aligning User Comments to the Content of a News Article," in Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part I 43, pp. 3–17, Springer.

Alshehri, J., **Stanojevic, M.**, Khan, P., Rapp, B., Dragut, E., and Obradovic, Z. (2022a), "MultiLayerET: A Unified Representation of Entites and Topics Using Multilayer Graphs," in The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD).

Alshehri, J., **Stanojevic, M.**, Dragut, E., and Obradovic, Z. (2022b), "On Label Quality in Class Imbalance Setting-A Case Study," in The 20th IEEE International Conference on Machine Learning and Applications (ICMLA)

Diep, B., **Stanojevic, M.**, and Novikova, J. (2022), "Multi-modal deep learning system for depression and anxiety detection," in Empowering Communities: A Participatory Approach to AI for Mental Health

Lastly, the following abstracts, pre-prints, and articles that were completed during the Ph.D. program, although not directly related, are listed in the order of their completion:

Stanojevic, M. and Chanda, A. K. (2017), "Multi-class Image Classification using Deep Neural Networks on Extremely Large Dataset,".

Ball, S., **Stanojevic, M.**, Knighton, C., Campbell, W., Thaung, A., Fisher, A., Bhatti, A., Kang, Y., Srivastava, P., Zhou, F., et al. (2018), "2474. Early Feedback From a Pilot of a Cognitive Computing System to Analyze Immunization Data," in Open Forum Infectious Diseases, vol. 5, p. S741, Oxford University Press.

Brinkley, J., Ball, S., Thaung, A., Campbell, W., Obradovic, Z., **Stanojevic, M.**, Zhou, F., Greby, S., Cindi, K., and Fisher, A. (2018), "Exploring the Metadata of Vaccine-Related Twitter Posts: Just How Much Activity Is There and Where Does It Come from?" in 2018 Annual Research Meeting, AcademyHealth.

Han, C., Cao, X. H., **Stanojevic, M.**, Ghalwash, M., and Obradovic, Z. (2019), "Temporal graph regression via structure-aware intrinsic representation learning," in Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 360–368, SIAM.

Pham, Q., **Stanojevic, M.**, and Obradovic, Z. (2020), "Extracting Entities and Topics from News and Connecting Criminal Records," arXiv preprint arXiv:2005.00950

Ljubic, B., Hai, A. A., **Stanojevic, M.**, Diaz, W., Polimac, D., Pavlovski, M., and Obradovic, Z. (2020), "Predicting complications of diabetes mellitus using advanced machine learning algorithms," Journal of the American Medical Informatics Association, 27, 1343–1351.

Tarca, A. L., Pataki, B. A., Romero, R., Sirota, M., Guan, Y., Kutum, R., Gomez-Lopez, N., Done, B., Bhatti, G., Yu, T., et al. (2021), "Crowdsourcing assessment of maternal blood multi-omics for predicting gestational age and preterm birth," Cell Reports Medicine, 2.

Andjelkovic, J., Ljubic, B., Hai, A. A., **Stanojevic, M.**, Pavlovski, M., Diaz, W., and Obradovic, Z. (2022), "Sequential machine learning in prediction of common cancers,"

Informatics in Medicine Unlocked, 30, 100928

CHAPTER 2

BACKGROUND

In this section, we describe the foundations of machine learning, neural networks, and transfer learning. We also introduce three applications: natural language processing, machine learning for medicine and healthcare, and machine learning for biology. All these areas are broad, and many algorithms and techniques are developed within them. However, in the following sections, we will focus only methods that give appropriate background to understand the following chapters.

2.1 Machine Learning

In machine learning, the input is a vector $\boldsymbol{x} \in \boldsymbol{R^d}$ with d features. The entire dataset is represented as a matrix $\boldsymbol{X} \in \boldsymbol{R^{n \times d}}$, containing n examples. Each row represents a sample, and each column represents a feature.

Various learning algorithms exist, including supervised, unsupervised, semi-supervised, self-supervised, and reinforcement learning. This thesis utilizes all of these learning methods, except reinforcement learning and therefore they are briefly described below.

In **supervised learning**, each input $\boldsymbol{x_i}$ is associated with a separate label y_i, which can be arranged as a vector $\boldsymbol{y}$ for the entire dataset. The main categories of machine learn-

ing supervised tasks are classification and regression, where in classification, the label y_i belongs to one of a pre-determined number of categories, while in regression, y_i is a continuous number. Binary classification refers to a scenario where there are only two categories, while multiclass classification is used when there are more than two categories to be classified.

Unsupervised learning involves a dataset that consists of unlabeled examples $\boldsymbol{x_i}$. It aims to create a model that either transforms original vectors into another vector with different features or produces a value that can be utilized to solve a practical problem. For instance, clustering produces the id of the cluster for each feature vector in the dataset. Outlier detection produces a real number that indicates how different $\boldsymbol{x_i}$ is from a "typical" example in the dataset. On the other side, dimensionality reduction yields a feature vector with fewer features than the input $\boldsymbol{x_i}$,

Semi-supervised learning involves a dataset that consists of both labeled and unlabeled examples, with the latter typically outnumbering the former. The objective of semi-supervised learning is similar to that of supervised learning algorithms. The idea behind using a large number of unlabeled examples is to potentially enhance the learning algorithm's ability to generate a better model, either by finding or computing it.

Self-supervised learning makes predictions about a particular aspect of the input data that is not explicitly labeled. It does this by using the input data itself as a supervisory signal. This means that the model labels the data without requiring human input. For example, in image classification, a self-supervised learning algorithm can predict the rotation angle of an image and then use this prediction to improve its classification performance. By using self-supervised learning, a model can learn from vast amounts of unlabeled data and potentially generalize better to new and unseen data. Self-supervised learning has been effective in many domains, including natural language processing, computer vision, and speech recognition.

2.1.1 Linear Regression

In order to perform supervised, semi-supervised, or self-supervised learning, one need to create functions that will learn from the features. The simple approach is to approximate the continuous output variable y_i as a linear function of the input $\boldsymbol{x_i}$, where the linear function $h_\theta(x)$ is represented by a set of parameters or weights $\boldsymbol{\theta}$. x_{i0} is set to 1, which represents the intercept term so that the linear function can be expressed as the dot product of the weight vector $\boldsymbol{\theta}$ and the input vector $\boldsymbol{x_i}$. This notation can be written as

$$h_\theta(\boldsymbol{x_i}) = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + ... + \theta_n x_{in} = \boldsymbol{\theta^T x_i}. \tag{2.1}$$

x_0 is often called a bias term. It is part of all machine learning models, but it might not always be explicitly given in the equations in this text.

A suitable method is needed to learn or select the parameters $\boldsymbol{\theta}$ given the training set. One approach is to minimize the difference between $h_\theta(\boldsymbol{x_i})$ and y_i for the available training examples. To achieve this, a function is defined that quantifies how much the predicted values $h_\theta(\boldsymbol{x_i})$ deviate from the corresponding actual values y_i for a particular set of $\boldsymbol{\theta}$. This function is known as the **cost function** or **objective function** or **loss function** (eq. 2.2). In order to learn parameters the function can be maximized or minimized. When maximization or minimization objective is given at the begining of equation, we will represent loss function with L, otherwise we will use J.

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\boldsymbol{x_i}) - y_i)^2 \tag{2.2}$$

where m is the number of training examples. The aim is to minimize $J(\boldsymbol{\theta})$ by adjusting the parameters $\boldsymbol{\theta}$ so that the predicted values $h_\theta(\boldsymbol{x_i})$ become closer to the actual values y_i. This equation represents a **mean squared error** (MSE) and can be seen as a **maximum likelihood estimation** (see full derivation in Bishop (2006)).

Linear regression using mean squared error is also known as **linear least squares**. A

common approach is to formulate the problem as a matrix equation over all samples (omitting the bias term): $\boldsymbol{X\theta} = \boldsymbol{y}$. The **normal equation** (eq. 2.3) minimizes the sum of squared differences between the left and right sides, producing the desired parameters $\boldsymbol{\theta}$. The matrix $\boldsymbol{X^T X}$ is known as the normal matrix.

$$\boldsymbol{X^T X\theta} = \boldsymbol{X^T y} \implies \boldsymbol{\theta} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y} \tag{2.3}$$

2.1.2 Logistic Regression

Logistic regression aims to model y_i as a linear function of $\boldsymbol{x_i}$, but this is challenging because y_i is binary. The linear combination of features $\boldsymbol{\theta^T x_i}$ spans from negative to positive infinity, while y_i can only take two values. Defining a negative label as 0 and a positive label as 1 would require only a simple continuous function with a co-domain of (0,1). This would allow us to assign a negative label to example $\boldsymbol{x_i}$ if the model value is closer to 0 and a positive label if it's closer to 1. One such function is the standard **logistic function**, also called the **sigmoid function** (eq. 2.4).

$$f(x) = \frac{1}{1+e^{-x}} \implies h_\theta(\boldsymbol{x_i}) = \hat{y}_i = \frac{1}{1+e^{-(\theta^T x_i)}} \tag{2.4}$$

where e is Euler's number. $f(\boldsymbol{\theta}, \boldsymbol{x_i})$ output can be interpreted as the probability of y_i being positive. A threshold value such as 0.5 can be used to determine whether a given example belongs to the positive or negative class. The optimal threshold value may vary depending on the problem.

The resulting probability is calculated as:

$$\hat{p}(y_i = 1|\boldsymbol{x_i};\theta) = \hat{y}_i = \sigma(\boldsymbol{\theta^T x_i}). \tag{2.5}$$

where $\hat{y}_i$ is predicted label for the example $\boldsymbol{x_i}$. As the output variable conforms to a Bernoulli distribution, the probability of one class is determined by the probability of the other.

For multi-class classification, weight parameters, $\boldsymbol{\theta}$, are learned for each class label, $y_j, j \in (1, K)$, given K classes. The softmax function is then used to squash these values into a categorical distribution.

$$\hat{p}(y_{ij}|\theta, x_i) = \frac{e^{\theta_j \boldsymbol{x_i}}}{\sum_{k=1}^{K} e^{\theta_k \boldsymbol{x_i}}}, \;\; j = 1, \ldots, K \tag{2.6}$$

where the denominator is called the partition function, which normalizes the distribution by summing over the scores of all K classes. The **cross-entropy** between the empirical conditional probability, $p(y_i|\boldsymbol{x_i})$, and the model probability, $\hat{p}(y_i|\boldsymbol{\theta}; \boldsymbol{x_i})$, is then calculated for each example, $\boldsymbol{x_i}$:

$$H(p, \hat{p}; \boldsymbol{x_i}) = -\sum_{j=1}^{K} p(y_{ij}|\boldsymbol{x_i}) \log \hat{p}(y_{ij}|\boldsymbol{\theta}; \boldsymbol{x_i}). \tag{2.7}$$

For binary classification, the cross-entropy can be simplified to:

$$H(p, \hat{p}; \boldsymbol{x_i}) = -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i). \tag{2.8}$$

The objective of the model is to minimize the average cross-entropy over all examples in the dataset:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} H(p, \hat{p}; \boldsymbol{x_i}). \tag{2.9}$$

2.1.3 Common Cost Functions

Several other cost and loss functions can be used, depending on the specific application. Unlike MSE, there is no closed-form solution to find the optimal weights for most loss functions. The iterative **gradient descent** algorithm is used to minimize the error in the model (see more details in subsection 2.1.4).

Similar to MSE, **mean absolute error** (MAE - eq. 2.10) and **mean bias error** (MBE - eq. 2.11) can be used to learn the regression model, but they have different properties.

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} |h_\theta(\boldsymbol{x_i}) - y_i| \tag{2.10}$$

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}(h_\theta(\boldsymbol{x_i}) - y_i) \tag{2.11}$$

The computation of gradients for MAE requires the application of more advanced techniques such as linear programming. Moreover, MAE exhibits greater resilience to outliers compared to MSE, as it does not involve the use of a squared term in its calculation. MBE is similar to MSE, but it is rarely used in machine learning because positive and negative biases cancel out and it is hard to understand the overall error of the model.

The **hinge loss**, also known as max-margin loss, is commonly used in support vector machines (SVMs) and for learning word embeddings. It is defined as:

$$L_\delta = max(\theta, \delta - h_\theta(\boldsymbol{x_p}) + h_\theta(\boldsymbol{x_n})) \tag{2.12}$$

where δ is a positive constant, usually 1, f is the model, and $\boldsymbol{x_p}$ and $\boldsymbol{x_n}$ are positive and negative examples, respectively. The hinge loss encourages the model to have a higher score for positive examples than for negative examples by the margin δ.

The **Huber loss**, also known as smooth MAE, is another loss function less sensitive to outliers than squared error. It is also possible to calculate Huber loss without gradient descent. Huber loss is used to calculate the proxy A distance in domain adaptation. It is defined as:

$$J_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{for } |y - \hat{y}| \leqslant \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases} \tag{2.13}$$

Contrastive loss (Chopra et al. (2005)) is a commonly used loss function in recent works, specifically in deep learning for metric learning tasks, such as image or text similarity learning. The goal is to learn a function that maps inputs of the same class to similar embeddings, while inputs of different classes are mapped to dissimilar embeddings. The loss is defined as:

$$L(\boldsymbol{x_i}, \boldsymbol{x_j}, \theta) = \mathbb{1}[y_i = y_j]||h_\theta(\boldsymbol{x_i}) - h_\theta(\boldsymbol{x_j})||_2^2 + \mathbb{1}[y_i \neq y_j]max(0, \delta - ||h_\theta(\boldsymbol{x_i}) - h_\theta(\boldsymbol{x_j})||_2)^2 \tag{2.14}$$

The hyperparameter δ serves as a threshold for the minimum distance allowed between samples belonging to different classes. Contrastive loss encourages similar samples to be close to each other and dissimilar samples to be far apart, thereby learning discriminative embeddings.

Triplet loss (Schroff et al. (2015)) is a commonly used loss function in deep learning for learning embeddings that encode information about the similarity of input data points. Given a triplet of data points (anchor, positive, and negative), the triplet loss function encourages the distance between the anchor and positive examples to be smaller than the distance between the anchor and negative examples by a margin. This is achieved with the following loss function:

$$L(\boldsymbol{x_a}, \boldsymbol{x_p}, \boldsymbol{x_n}) = max(0, ||h_\theta(\boldsymbol{x_a}) - h_\theta(\boldsymbol{x_p})||_2^2 - ||h_\theta(\boldsymbol{x_a}) - f(\boldsymbol{x_n})||_2^2 + \delta) \tag{2.15}$$

The parameter δ is a margin hyperparameter that controls the degree of separation between the positive and negative examples. The triplet loss function is designed to ensure that the distance between the anchor and positive examples is smaller than the distance between the anchor and negative examples by at least δ. The optimization problem associated with triplet loss can be challenging due to the combinatorial explosion of triplets that need to be considered during training. Various techniques have been proposed to address this challenge, including semi-hard and hard negative mining, which select only the most informative negative examples during training.

2.1.4 Gradient Descent

Gradient descent is used to minimize an objective function, $J(\boldsymbol{\theta})$, by updating the model's parameters, in the direction opposite to the gradient of the function, $\nabla_\theta J(\boldsymbol{\theta})$. The gradient is a vector containing the partial derivatives $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$, where the j-th element of the gradient is the partial derivative of $J(\boldsymbol{\theta})$ with respect to θ_j. To update the parameters, gradient

descent uses the following formula:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta * \nabla_{\theta} J(\boldsymbol{\theta}), \tag{2.16}$$

where η is the learning rate that determines the size of the update. The learning rate is typically decreased during training to ensure convergence, but in recent literature, many learning rate schedules have been designed (see a review of learning rates in Wu et al. (2019)).

If the $\boldsymbol{\theta}$ is updated with all samples at once, the algorithm is called gradient descent. Alternatively, **stochastic gradient descent** (SGD) updates the parameters for each individual training example. This leads to faster convergence since the parameters are updated more frequently, but it can result in noisy updates and slower convergence to the optimal solution.

Modern networks have too many parameters and training samples, so it is impossible to put them all in memory. On the other side reducing the noise in training and speeding up the convergence are also important. Therefore, batch or mini-batch gradient descent is commonly used. It is a compromise between full-dataset gradient descent and stochastic gradient descent. It updates the parameters based on small batches of training examples, typically between 16 and 256, rather than the entire dataset. In the case of very big models, batch size can be as low as 2 or 4 samples due to GPU memory limitations.

2.1.5 Bias-variance Tradeoff

Machine learning aims to create a model that performs well on new and unseen inputs, a process known as generalization. The data is divided into a training set for training the model, a validation (or cross-validation) set for selecting good hyper-parameters, and a test set to evaluate the model's performance. If the dataset is too small to create a separate validation part, cross-validation is used to calculate the error by averaging results from multiple random splits of the data. The most widely used form of cross-validation is k-fold, where the data is divided into k subsets, and training and evaluation are performed k

times with each subset serving as the testing data once. The model's test set performance serves as an indicator of its ability to generalize to new inputs.

Train, validation, and test sets are often assumed to be i.i.d., meaning that examples within each set are independent and both sets are drawn from the same distribution. To achieve this, datasets need to be randomly shuffled before splitting into training, cross-validation and test datasets. However, in some cases, data have an internal dependency, e.g., longitudinal order of patients' visits, and this information is important for better data modeling. In those cases, datasets are usually split so that dependency is kept and not randomized. Some works in this thesis use data with longitudinal dependency, but you can assume we are discussing i.i.d. settings in this thesis except it is specified differently.

The goal of minimizing a model's generalization error results in two objectives:

- Minimizing the training error
- Minimizing the difference between training and test errors.

The bias-variance tradeoff is a fundamental ML concept that deals with balancing model complexity with model performance. The bias refers to the difference between the expected predictions of a model and the true values of the data, while the variance refers to the variability of the model's predictions for different inputs. A model with high bias is too simple and may not capture the underlying patterns in the data, leading to underfitting. On the other hand, a model with high variance is too complex and may overfit the data, resulting in poor generalization to new data. Machine learning aims to find the optimal balance between bias and variance, leading to a model that performs well on training and test data.

The bias of an estimator $\hat{\boldsymbol{\theta}}$ indicates the degree of difference between its estimated value and the actual underlying value of the parameter, $\boldsymbol{\theta}^*$, that is anticipated based on the distribution of the data generating process:

$$bias(\hat{\boldsymbol{\theta}}) = E[\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*]. \tag{2.17}$$

An estimator $\hat{\boldsymbol{\theta}}$ is considered to be unbiased if $bias(\hat{\boldsymbol{\theta}}) = 0$. The variance of an estimator is defined as:

$$var(\hat{\boldsymbol{\theta}}) = E[\hat{\boldsymbol{\theta}}^2] - E[\hat{\boldsymbol{\theta}}]^2. \tag{2.18}$$

The square root of the variance of an estimator is referred to as its standard error, $SE(\hat{\boldsymbol{\theta}})$. To evaluate an estimator's performance, the MSE can be calculated by comparing the estimator's value to the true parameter value. A model trained with MSE has an expected error that is lower bounded by the sum of three terms (see more in Ruder (2019):

1. The square of the method's bias, i.e., the error resulting from the simplifying assumptions inherent in the model.

2. The method's variance, i.e., the amount by which its results vary from the mean.

3. The variance of the true underlying distribution.

$$MSE = bias(\hat{\boldsymbol{\theta}})^2 + var(\hat{\boldsymbol{\theta}}) + \sigma^2 \tag{2.19}$$

A model that has high bias is referred to as **underfitting** and one with high variance is **overfitting**. The capacity of a model, its ability to fit various functions, plays a crucial role in determining if it underfits or overfits. A machine learning model performs optimally when its capacity matches the requirements of the task. Occam's razor is a principle that suggests that when there are multiple explanations for a phenomenon, the simplest explanation is most likely the correct one. According to Occam's razor, a hypothesis that explains the data using the fewest assumptions is preferred over more complex hypotheses, as it is more parsimonious and less prone to errors or overfitting. However, the principle is not a strict rule and must be balanced with other factors such as empirical evidence, explanatory power, and predictive accuracy.

The **No Free Lunch** theorem (Wolpert and Macready (1997)) states that no one algorithm can outperform all others in all situations. On average, all classification algorithms

will have the same performance when classifying new data. In practical terms, the aim is to steer the algorithm towards data distributions or relationships that are likely to be encountered in the real world and to build algorithms that are effective for specific tasks. This thesis will refer to these assumptions as **inductive bias**. It is built into the model to help predict well on new data of a similar kind. The goal is to create models with a useful inductive bias for generalizing to new domains, tasks, and languages.

Conversely, it is important to create a model that did not learn any **spurious correlations** (Simon (1954)). This is crucial for domain adaptation settings. A spurious correlation is a statistical relationship between two variables that appears to be significant but is actually due to chance or a third variable that affects both. In other words, the relationship is not causal and does not have a meaningful explanation. Spurious correlation can occur when there is not enough data or when data is collected from non-random samples. It can also arise when the analysis fails to take into account confounding variables that influence both the dependent and independent variables. Therefore, it is essential to carefully consider the underlying mechanisms behind a correlation and to avoid drawing causal conclusions without proper investigation.

The statistical learning theory limits the generalization error by estimating the gap between training error and generalization error. However, this method has proven unreliable in practice, as determining the capacity of deep neural networks is difficult. Despite these challenges, the generalization capabilities of deep neural networks are still an area of active research.

2.1.6 Regularization

Regularization is used to prevent the overfitting of models by adding a penalty term to the loss function that encourages the model to have smaller weight. This helps prevent the model from overfitting by reducing its complexity and making it more generalizable. Common forms of regularization use vector norms.

L1 regularization, also known as **Lasso regularization**, adds a penalty term to the loss function proportional to the absolute value of the model's weights - 1st norm. This encourages the model to have sparse weights and helps it to focus on the most important features. **L2 regularization**, known as **Ridge regularization**, adds a penalty term proportional to the squared value of the model's weights - 2nd norm. This encourages the model to have smaller weights overall and can help prevent it from being overly sensitive to small variations in the training data. The infinity norm regularization is used to penalize the maximum parameters value. Frobenius norm is used for regularization when two weight matrices need to be orthogonal. **Dropout** regularization involves randomly dropping out some of the neurons in a neural network during training, which helps prevent the network from overfitting by forcing it to learn more robust representations of the data. Regularization strength is controlled with a hyperparameter. Different types of regularization can be combined, including the combination of L1 and L2 regularization, referred to as **elastic net regularization**. This method balances both types of regularization:

$$\alpha||\theta||_1 + (1-\alpha)||\theta||_2^2. \tag{2.20}$$

2.1.7 Evaluation Metrics

Evaluation metrics are an essential aspect of assessing the performance of models and algorithms in machine learning. These metrics measure a model or algorithm's performance on a given dataset or task. Several evaluation metrics are used in machine learning, each with its own strengths and weaknesses. The most common evaluation metrics include **accuracy**, **precision**, **recall**, **F1-score**, the **area under the ROC curve** (AUC), **mean squared error** (MSE), and **root mean squared error** (RMSE).

Classification metrics: In binary classification, each instance is classified into one of two classes, typically labeled as positive or negative. The **true positives** (TP) are the instances correctly classified as positive by the model. The **true negatives** (TN) are the instances correctly classified as negative by the model. **False positives** (FP) are the instances

that are incorrectly classified as positive, while **false negatives** (FN) are the instances that are incorrectly classified as negative. These metrics can be used to calculate various evaluation measures for binary classification models.

Accuracy measures the percentage of correctly classified instances in a dataset. It is useful when the dataset is balanced, meaning the number of instances in each class is roughly equal. However, accuracy can be misleading when the dataset is imbalanced, as it may give a false sense of good performance on the majority class while ignoring the minority class. Accuracy can be calculated as:

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.21}$$

Precision and recall are particularly useful metrics when dealing with imbalanced datasets. Precision measures the proportion of true positives among all positive predictions made by the model, while recall measures the proportion of true positives among all actual positive instances in the dataset. These two metrics are often used to give a more complete picture of a model's performance on an imbalanced dataset.

Precision is calculated as:

$$Precision = \frac{TP}{TP + FP}. \tag{2.22}$$

Recall (also known as sensitivity) is calculated as:

$$Recall = \frac{TP}{TP + FN}. \tag{2.23}$$

The F1-score is the harmonic mean of precision and recall and is another useful metric for evaluating the performance of models on imbalanced datasets. It balances precision and recall and is particularly useful when the dataset is highly imbalanced. It is calculated as:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{2.24}$$

The AUC represents the area under the receiver operating characteristic (ROC) curve, which plots the true positive rate against the false positive rate. The AUC is useful when comparing models that have different decision thresholds.

These metrics can help assess the performance of a binary classification model and provide insights into its strengths and weaknesses. However, it is important to consider the specific problem domain and the costs associated with false positives and false negatives when interpreting these metrics.

Multiclass classification metrics generalize binary classification metrics (accuracy, precision, recall, and F1 score) using macro-averaging and micro-averaging. The macro-averaged metrics compute the average metric value across all classes, while the micro-averaged metrics compute the metric value globally across all classes. In addition, **confusion matrix** is commonly used to evaluate multiclass classification. The confusion matrix provides a tabular summary of the predicted class labels versus the actual class labels. In the confusion matrix, the rows represent the true classes, and the columns represent the predicted classes. The diagonal elements of the confusion matrix represent the number of correct predictions, while the off-diagonal elements represent the number of incorrect predictions.

Regression metrics: Mean Squared Error (MSE): MSE is the most common metric used for regression problems. It measures the average of the squared differences between the predicted and actual values. Mathematically, it is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.25}$$

where n is the number of data points, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

Root Mean Squared Error (RMSE): RMSE is similar to MSE but takes the square root of the average squared differences between the predicted and actual values. Mathe-

matically, it is calculated as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.26}$$

Mean Absolute Error (MAE): MAE measures the average of the absolute differences between the predicted and actual values. It is less sensitive to outliers than MSE. Mathematically, it is calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.27}$$

Coefficient of Determination (R-squared): R-squared measures the proportion of variance in the dependent variable that is explained by the independent variables in the model. It is a value between 0 and 1, where 1 indicates that the model perfectly predicts the dependent variable. Mathematically, it is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.28}$$

where $\bar{y}$ is the mean of the actual values.

Mean Absolute Percentage Error (MAPE): MAPE measures the percentage of average absolute difference between the predicted and actual values. It is often used to compare the performance of different models. Mathematically, it is calculated as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{2.29}$$

These metrics can be used individually or in combination to evaluate the performance of a regression model. It is important to choose the appropriate metric based on the problem requirements and characteristics of the data.

2.2 Effective Strategies for Data Handling in Machine Learning

Data handling is a critical aspect of machine learning, as the quality and quantity of data available directly affect the performance of machine learning algorithms. Machine learning models depend on large amounts of data to make accurate predictions, but the sheer volume of data can present challenges for handling and processing. In addition, data can be noisy, unbalanced, and contain missing values, which further complicates the learning.

Effective strategies for data handling in machine learning have been developed to address these challenges. These strategies aim to optimize data quality and quantity while also ensuring that the data is properly preprocessed to ensure compatibility with machine learning algorithms. The following are some of the key strategies for data handling in machine learning:

- **Data Cleaning**: The first step in data handling is to clean the data, which involves removing any irrelevant data, dealing with missing values, and correcting any errors in the data. Data cleaning can be time-consuming, but it is essential to ensure that the data is of high quality and error-free.

- **Data Preprocessing**: After cleaning the data, the next step is to preprocess the data, which involves transforming the data into a format that can be easily used by machine learning algorithms. This can include tasks such as scaling, normalization, and feature extraction.

- **Data Augmentation**: Data augmentation involves creating new data samples by applying transformations to existing data. This can be useful when there is limited data available, as it can help increase the dataset's size and improve the accuracy of machine learning models.

- **Data Balancing**: In some cases, the data may be unbalanced, meaning there are significantly more samples of one class than another. This can lead to biased predic-

tions, as the machine learning model will be more likely to predict the majority class. Data balancing techniques can be used to address this issue, such as oversampling the minority class or undersampling the majority class.

- **Data Splitting**: Finally, splitting the data into training and testing sets is essential to evaluate the performance of machine learning models. This ensures that the model is trained on one set of data and evaluated on a separate set, which helps avoid overfitting.

Effective strategies for data handling in machine learning are crucial for developing accurate and reliable machine learning models. By cleaning, preprocessing, augmenting, balancing, and splitting the data appropriately, machine learning models can be optimized to deliver superior performance on a range of tasks. In the upcoming chapters, we will use those strategies to develop good datasets and learn the most information from them.

In the rest of this section, we describe three specific data-handling topics in more detail. The next chapters describe proposed techniques to evaluate and remedy those challenges in domain adaptation settings as they are common in real-world datasets.

- influence and imputation of missing data
- effects of imbalanced and skewed datasets
- bias in data

2.2.1 Missing Data and Imputation Techniques

Missing data is a common issue with data coming from real world experiments including social sciences, healthcare, physics, and engineering datasets. Many machine learning models are not able to work with missing data and require data to be fully known. Even when the models are able to process data with missing information, the missingness can negatively affect the accuracy and reliability of statistical analyses and machine learning

models. In such cases, researchers often use imputation techniques to fill in the gaps in the dataset.

Imputation is the process of estimating missing values based on observed data. The choice of imputation method largely depends on the characteristics of the dataset and the research question being investigated. One commonly used method for imputing missing data is **mean imputation**, where the missing values are replaced with the mean value of the observed data for that variable. However, this method can lead to biased estimates and an underestimation of the standard errors.

Another commonly used method is **regression imputation**, which involves fitting a regression model to the observed data and using this model to predict the missing values. This method can provide more accurate estimates than mean imputation, but it assumes that the relationship between the missing variable and the other variables in the dataset is linear.

Other simple imputation methods for imputing missing data include **k-nearest neighbor imputation**, **hot deck imputation**, and **probabilistic principal component analysis**.

Multiple imputations is a newer and popular technique for handling missing data, where multiple imputed datasets are created. The analysis is run on each dataset separately, with the results combined to generate an overall estimate. This method is more robust than other imputation techniques, but it can be computationally intensive and may require a large sample size.

The most advanced imputation is named **iterative imputation**. The missing values are imputed by estimating the values based on the observed data, then using these estimates to impute the missing values again. This process is repeated multiple times until the imputed values converge. There are several iterative imputation methods, such as the **Expectation-Maximization (EM) imputation** algorithm, the **Multiple Imputation by Chained Equations** (MICE) algorithm, and the **Sequential Regression Multivariate Imputation** (SRMI) algorithm.

The EM algorithm iteratively estimates the missing values and parameters of the data distribution. It maximizes the likelihood function to obtain the imputed values under the normal distribution assumption. The MICE algorithm, on the other hand, imputes the missing values in a stepwise manner, by regressing each missing variable on the observed variables. It then iteratively updates the imputed values for each variable based on the updated values of the other variables. The SRMI algorithm fits regression models for each variable with missing data, using the other variables as predictors. It then uses the estimated coefficients to impute the missing values.

Chapter 6 discusses proposed ways for imputations of data from anxiety clinical studies and evaluates the performance of those imputation methods as well as their influence on the prediction of the target variable.

2.2.2 Strategies for Achieving Dataset Balance in Machine Learning

Dataset balance refers to the distribution of samples across different classes or categories in a dataset. In many machine learning applications, having a balanced dataset is important for obtaining accurate and reliable results. A balanced dataset is one where the number of samples in each class is approximately the same. However, in practice, datasets are often imbalanced, with some classes having significantly fewer samples than others.

Imbalanced datasets can lead to biased models that perform poorly on the underrepresented classes. This is because machine learning algorithms are typically optimized to maximize overall accuracy, which can result in the model being biased towards the majority class. In such cases, the model may fail to capture the patterns and characteristics of the minority class, leading to poor performance on these samples.

A **skewed dataset** often contains a long tail, meaning that most of the observations belong to a few classes, while a small number of observations belong to all other classes. The class distribution often follows Zipf's law. This type of dataset is common in real-world applications such as fraud detection, medical diagnosis, customer churn prediction,

or any dataset which comes from the internet. Skewed and imbalanced names are often used interchangeably and both have the same challenges.

Several techniques have been developed for handling imbalanced and skewed datasets. One common approach is to **oversample** the smaller classes to balance the dataset. This can be achieved by replicating samples from the minority class, either randomly or by using more advanced techniques such as **Synthetic Minority Over-sampling Technique** (SMOTE). Oversampling can lead to overfitting and errors on the samples near the decision line. Another approach is to **undersample** the bigger classes to balance the dataset. This involves removing samples from the bigger classes to match the size of the smaller classes. Undersampling reduces dataset size and might remove important datapoints. There are many algorithms developed for oversampling and undersampling to overcome those challenges for specific types of datasets or machine learning methods.

The other group of commonly used techniques, **cost-sensitive learning**, assigns higher misclassification costs to the smaller classes, thereby encouraging the model to focus more on those classes. **Ensemble methods**, which combine multiple models trained on different oversampled or undersampled versions of the data to improve overall performance, can also be used to improve performance on imbalanced datasets. In recent years, **contrastive learning** has shown to be useful in improving performance on skewed or imbalanced datasets.

Precision, recall, **precision-recall (PR) curve**, F1, AUC-ROC, **Cohen's kappa coefficient**, and **Matthews correlation coefficient** metrics should be used to evaluate the performance of machine learning algorithms on imbalanced and skewed datasets. AUC-ROC evaluates the classifier's ability to distinguish between the positive and negative classes across different probability thresholds. A high AUC-ROC score indicates that the classifier is able to correctly identify the positive class even when the threshold is varied. The PR curve plots the precision (true positive rate) against the recall (sensitivity) for different probability thresholds. It is a measure of the classifier's ability to identify the positive class

correctly while minimizing the false positives.

We evaluate the behavior of deep learning models in learning properties of classes when the classes are balanced and unbalanced in Chapter 5 and in many other works published during this PhD studies.

2.2.3 Uncovering and Addressing Bias in Data

Bias can arise in many ways, including sampling bias, measurement bias, and selection bias, among others. These biases can lead to inaccurate and unreliable results and, in some cases, can even have serious consequences. It is especially important to understand and handle bias in data if a dataset contains information collected from different sources, which is becoming common due to big models which require a lot of data to perform well. It is also very important to understand bias in source and target domains. The chapter 5 explores how bias in different datasets used for domain adaptation can influence the performance.

Types of Bias:

- **Sampling Bias**: Sampling bias occurs when the sample of data collected is not representative of the population of interest. This type of bias can occur when the sample is too small or when certain groups are overrepresented or underrepresented in the sample.

- **Measurement Bias**: Measurement bias occurs when the data collection method is flawed, leading to inaccurate or incomplete data. This type of bias can occur due to measurement errors or due to differences in the way data is collected from different groups.

- **Selection Bias**: Selection bias occurs when the selection process for data collection favors certain groups over others. This type of bias can occur when the data collection process is non-random, leading to the overrepresentation or underrepresentation of certain groups.

- **Reporting Bias**: Reporting bias occurs when data is selectively reported or withheld. This type of bias can occur when researchers selectively report only the results that support their hypothesis or when certain results are not reported at all.

Data unbiasing in the context of machine learning refers to the practice of mitigating bias in the training data used to build a model. Bias can occur when the training data is not representative of the broader population, resulting in incorrect or unfair predictions when the model is applied to new data. To address this issue, data unbiasing techniques strive to balance the training data to accurately represent the population. This may involve oversampling underrepresented groups, undersampling overrepresented groups or re-sampling the data to achieve balance. Furthermore, fairness algorithms and metrics can be used to evaluate and reduce the impact of bias in the model.

Techniques for Addressing Bias:

- **Oversampling**: Oversampling involves intentionally increasing the representation of underrepresented groups in the data. This technique can be useful in correcting sampling bias.

- **Weighting**: Weighting involves assigning weights to the data points in order to correct for imbalances in the sample. This technique can be used to correct for both sampling bias and selection bias.

- **Imputation**: Imputation involves filling in missing data with estimates based on the available data. This technique can be used to correct for measurement bias.

- **Sensitivity Analysis**: Sensitivity analysis involves examining the results of a study under different scenarios and assumptions to determine the robustness of the findings. This technique can be used to assess the impact of potential biases on the results.

By carefully considering and addressing potential biases in data, the accuracy and reliability of the findings are ensured, leading to better decision-making and outcomes. Data

unbiasing is crucial in applications that hold high stakes, such as medical diagnoses, criminal justice predictions, and hiring decisions. In these scenarios, flawed or biased predictions can have significant impacts on people and communities. Eliminating bias in machine learning models can result in fairer and more reliable models, leading to improved decision-making processes, heightened trust in technology, and more accurate outcomes.

2.2.4 Curse of dimensionality

The curse of dimensionality is a problem that arises when working with high-dimensional data. As the number of dimensions increases, the data becomes increasingly sparse and the distance between points becomes less meaningful. This can lead to issues such as overfitting, poor generalization, and reduced model performance. This is especially significant in medical and genetic data where number of patients is usually small, but a number of features is very high (Stanojevic et al. (2022); Ljubic et al. (2020); Andjelkovic et al. (2022)). It is also a prominent issue with very long texts where different extraction methods are applied to select relevant features (Alshehri et al. (2022); Stanojevic et al. (2023)). Chapter 7 shows the proposed solution for the high-dimensionality problem of phylogenetic research articles.

Consequences of high dimensionality:

- As the dimensionality increases, the number of possible models also increases exponentially, making it more likely that the model will overfit.
- The model is unable to generalize to new data.
- As the dimensionality increases, the number of possible configurations of the data also increases, making it more difficult for the model to learn the underlying patterns and generalize to new data.
- The model becomes more complex and harder to train.

There are several solutions to mitigate the impact of the curse of dimensionality:

- **Feature Selection**: Feature selection involves selecting a subset of the most informative features from the dataset. This can help reduce the dimensionality of the data and improve model performance. Feature selection can be done manually or using automated methods such as LASSO or Ridge regression.

- **Dimensionality Reduction**: Dimensionality reduction involves transforming the data into a lower-dimensional space while preserving the most important information. This can help reduce the curse of dimensionality and improve model performance. Some common methods for dimensionality reduction include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-SNE.

- **Regularization**: Regularization involves adding a penalty term to the loss function of the model to prevent overfitting. This can help reduce the complexity of the model and improve generalization. Some common regularization techniques include L1 regularization, L2 regularization, and Dropout.

- **Ensemble Learning**: Ensemble learning involves combining multiple models to improve performance. This can help reduce the impact of the curse of dimensionality by leveraging the strengths of different models.

2.3 Neural Networks

A neural network is made up of a series of functions that are interconnected with nonlinear activation functions interspersed throughout. For classification, the output layer frequently incorporates the softmax or sigmoid functions to generate categorical or Bernoulli distributions, respectively. Any layer that isn't an input or output layer is referred to as a hidden layer. Linear regression is a basic form of a neural network with no hidden layers and uses

a linear activation function. Logistic regression, on the other hand, employs a nonlinear activation function.

A one-layer **feed-forward neural network** (FFNN), also known as a **multi-layer perceptron** (MLP), contains one hidden layer. It can be defined mathematically with these equations:

$$h = \sigma_1(\boldsymbol{W_1 x} + \boldsymbol{b_1}), y = softmax(\boldsymbol{W_2 h} + \boldsymbol{b_2}) \tag{2.30}$$

where σ_1 is the activation function for the first hidden layer and W_1 and W_2 are weights connecting the input and hidden layer and hidden and output layer, respectively. Each layer has its own weight matrix ($\boldsymbol{W}$) and bias vector ($\boldsymbol{b}$), although different layers can share parameters, a technique referred to as tying or sharing, which often improves generalization by inducing an inductive bias. Parameter sharing is common in multi-task learning.

The expressiveness of deep neural networks lies mainly in their non-linear activation functions, which add non-linearity to the otherwise linear compositions of functions. While the sigmoid and softmax functions are commonly used at the output layers activation functions, the **rectified linear unit** (ReLU) (Fukushima (1975)) is a widely used activation function for hidden layers and is defined as $\sigma(\boldsymbol{x}) = max(0, \boldsymbol{x})$. The hyperbolic tangent function (tanh), which outputs values in the range (-1, 1), is another activation function that is less frequently used because of the complexity of calculating it.

Similar to many machine learning models with no closed-form solutions, neural networks are usually trained with mini-batch gradient descent. As the models have multiple layers, computing the loss function gradient with respect to the parameters ($\nabla_\theta J(\boldsymbol{\theta})$) is complex. To calculate the gradient, a dynamic programming algorithm named **backpropagation** (Rumelhart et al. (1986)) is used.

Forward propagation refers to computing the output of each layer in a neural network, starting from the input layer and proceeding through intermediate hidden layers to the final output layer.

Back-propagation employs the chain rule of calculus to propagate the gradient from the output layer backward to the input layer by calculating gradients of each layer. The loss function gradient with respect to each lower-level layer is then obtained by using the updated parameters and repeating the calculations. This requires knowing the gradients on all computations that result from each parameter. The computed gradients update the corresponding parameters with gradient descent for each layer. Given functions $\boldsymbol{y} = g(\boldsymbol{x})$ and $\boldsymbol{z} = f(g(\boldsymbol{x})) = f(\boldsymbol{y})$, the chain rule states that the derivative of $\boldsymbol{z}$ with respect to $\boldsymbol{x}$ ($\frac{d\boldsymbol{z}}{d\boldsymbol{x}}$) is equal to the derivative of $\boldsymbol{z}$ with respect to y ($\frac{d\boldsymbol{z}}{d\boldsymbol{y}}$) multiplied by the derivative of $\boldsymbol{y}$ with respect to $\boldsymbol{x}$ ($\frac{d\boldsymbol{y}}{d\boldsymbol{x}}$). Full derivation and discussion are available in the Deep Learning book (Goodfellow et al. (2016)).

2.3.1 Important Models

FFNNs are the oldest and standard neural network type. However, many specific layers or architectures have been made to better learn certain properties, such as local or global space and time correlations within data. The most commonly used networks today are **recurrent neural networks** (RNN) (Elman (1990)), **convolutional neural networks** (CNN) (LeCun et al. (1998); Fukushima (1980)), and **transformers networks** (Vaswani et al. (2017)). There are many models made by combining them with into more complex and task-specific architectures. We will describe RNN and transformers networks and models derived from them that are relevant to this thesis.

RNN networks specialize in processing sequential data, including text, patients' visits, time-series, DNA, and proteins. RNN can be viewed as a feed-forward neural network with dynamic hidden layers with the same parameters. Instead of having many layers, the model has one layer that is unfolded over time. In contrast to FFN, an RNN processes a new input at each time step or "layer" (as illustrated in Figure 2.1). The RNN retains a hidden state h_t, which serves as its memory of the sequence's information at each time step t. At each

time step, the RNN executes the following operation:

$$\begin{aligned} \boldsymbol{h_t} &= \sigma_h(\boldsymbol{W_h x_t} + \boldsymbol{U_h h_{t1}} + \boldsymbol{b_h}), \\ \boldsymbol{y_t} &= \sigma_y(\boldsymbol{W_y h_t} + \boldsymbol{b_y}) \end{aligned} \tag{2.31}$$

The activation functions σ_h and σ_y are used in the transformation process of the previous hidden state $\boldsymbol{h_{t-1}}$ through $\boldsymbol{U_h}$ and the current input $\boldsymbol{x_t}$ through $\boldsymbol{W_h}$, resulting in the updated hidden state $\boldsymbol{h_t}$ at each time step. Moreover, the RNN generates an output y_t at each time step.

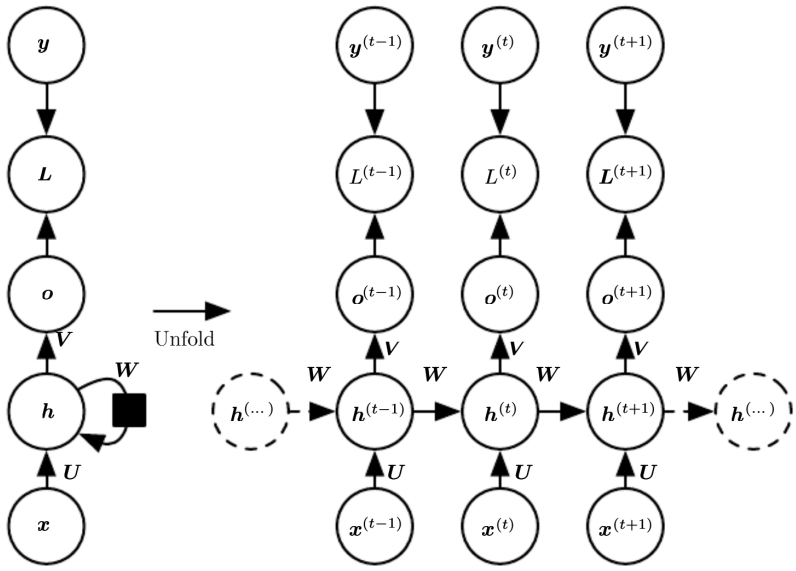


FIGURE 2.1: Recurrent neural network architecture: signal flow (left) and unfolded view (right) (Goodfellow et al. (2016))

When training RNNs, the gradients must be propagated through time instead of the model's depth, called **backpropagation through time** (BPTT) (Werbos (1990)). The gradient is typically propagated until a predetermined fixed length, enabling the model to learn longer dependencies.

While RNNs are very useful for modeling sequences, they also have many challenges listed below. Other architectures were proposed to solve those issues.

- The **vanishing and exploding gradient problem** is a common issue in RNNs. This problem arises when the gradients become either too small (vanishing) or too large

(exploding) during the backpropagation process due to chain multiplications. This can make it difficult for the network to learn long-term dependencies in sequential data, which is a key requirement for many applications of RNNs.

- RNNs are **computationally complex and expensive**, particularly when dealing with long sequences of data. This is because the computations performed by the RNN at each time step must be repeated for each time step in the sequence. This can make training and inference time-consuming and can limit the size of the input sequence that the network can process.

- RNNs are **difficult to parallelize**, because the computations at each time step must be performed sequentially. This limits their ability to scale to large datasets and take advantage of parallel processing architectures such as GPUs.

- Although RNNs are designed to capture long-term dependencies in sequential data, they can struggle to do so in practice - **catastrophic forgetting**. This is because the network may have difficulty retaining information from earlier time steps as the sequence becomes longer, leading to errors in the prediction.

Long-short term memory (LSTM) (Hochreiter and Schmidhuber (1997)) are favored over RNNs for processing sequential data as they can retain information for longer periods. This is crucial for modeling long-term dependencies. The LSTM cell (Figure 2.2) can be viewed as a more advanced RNN cell that implements mechanisms to determine what to ”remember” and ”forget”. The LSTM enhances the RNN with forget gate $\boldsymbol{f_t}$, input gate $\boldsymbol{i_t}$, and output gate $\boldsymbol{o_t}$, which are functions of the current input $\boldsymbol{x_t}$ and the previous hidden state $\boldsymbol{h_t}$. These gates interact with the previous cell state $\boldsymbol{c_{t-1}}$, the current input, and the current cell state $\boldsymbol{c_t}$, enabling the model to selectively overwrite or retain information. The

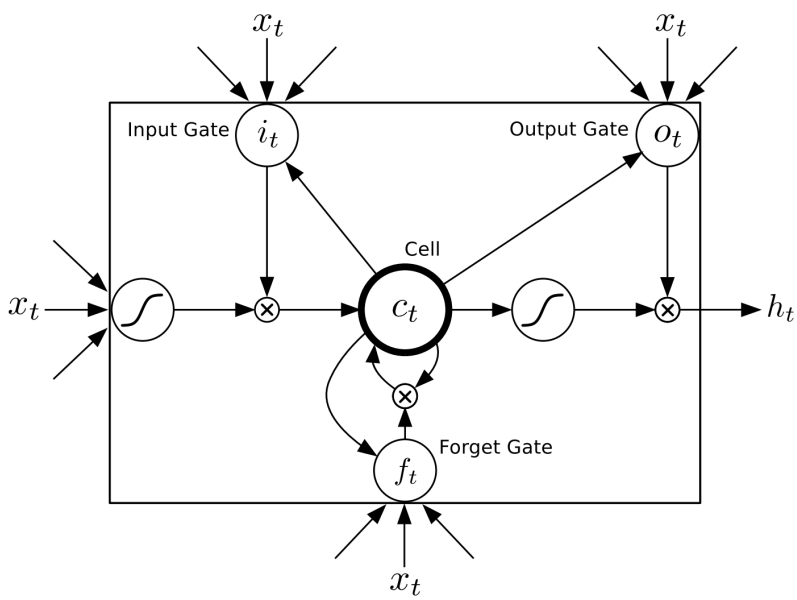


FIGURE 2.2: LSTM cell (Graves et al. (2013))

entire model is expressed as follows:

$$\begin{aligned}
\boldsymbol{f_t} &= \sigma_g(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t1}} + \boldsymbol{b_f}), \\
\boldsymbol{i_t} &= \sigma_g(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t1}} + \boldsymbol{b_i}), \\
\boldsymbol{o_t} &= \sigma_g(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t1}} + \boldsymbol{b_o}), \\
\boldsymbol{c_t} &= \boldsymbol{f_t} \circ \boldsymbol{c_{t1}} + \boldsymbol{i_t} \circ \sigma_c(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t1}} + \boldsymbol{b_c}), \\
\boldsymbol{h_t} &= \boldsymbol{o_t} \circ \sigma_h(\boldsymbol{c_t}),
\end{aligned} \tag{2.32}$$

where σ_g is the sigmoid activation function, σ_c and σ_h are the tanh activation function, and $\circ$ is element-wise multiplication, also known as the Hadamard product.

A **bidirectional LSTM** (BiLSTM) (Graves et al. (2013)) is frequently used, consisting of separate LSTMs that process the input sequence forwards and backwards. At each time step, the hidden state concatenates the hidden states from the forward and backward LSTMs.

Multiple LSTM cells can be stacked to form a deeper network.

Transformers (Vaswani et al. (2017))architecture (Figure 2.3) is designed specifically to deal with sequential data. The transformer replaces the RNN with a self-attention mecha-

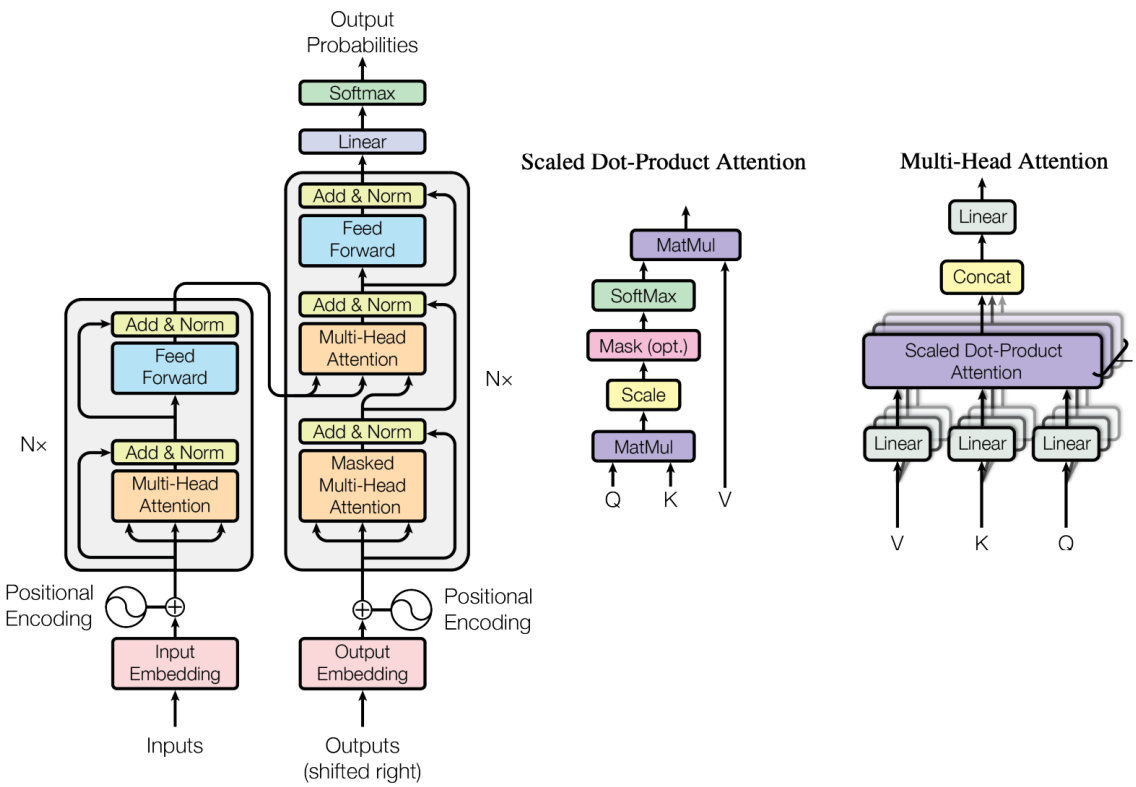


FIGURE 2.3: Transformers architecture (Vaswani et al. (2017))

nism. It allows the model to learn to focus on the most important parts of the input sequence for a given task. This mechanism is based on the dot-product between the queries ($\boldsymbol{Q}$), keys ($\boldsymbol{K}$), and values ($\boldsymbol{V}$). Given a sequence of input vectors $\boldsymbol{x}$, the self-attention mechanism computes a set of attention scores between each element in the sequence, which can be expressed mathematically as:

$$attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d_k}})\boldsymbol{V} \tag{2.33}$$

where d_k is the number of dimensions in keys.

The transformer architecture also uses residual connections and layer normalization, which help to stabilize the training process and improve the generalization performance of the network.

The transformer consists of self-attention blocks, followed by feedforward neural net-

works and layer normalization. The self-attention blocks calculate the attention scores for each element in the input sequence with respect to all other elements. These scores are then used to weigh the input elements and produce a new representation to be passed to the feedforward neural networks. The layer normalization is used to normalize the inputs to the feedforward neural networks, which helps to prevent the model from overfitting.

The strengths of the transformer architecture lie in its ability to handle sequential data effectively and efficiently. Unlike RNNs, the transformer can operate on sequences of arbitrary length. Additionally, the self-attention mechanism allows the model to learn to focus on the most important parts of the input sequence, which is particularly useful for tasks such as sentiment analysis and question answering.

However, the transformer architecture is not without its weaknesses. One of the main challenges with the transformer is its computational complexity (quadratic in terms of input length), which can make it difficult to train on large datasets and extremely large sequences, such as documents. Additionally, the attention mechanism is difficult to interpret, even though some mechanisms have been developed to do so.

The transformer architecture has been applied to a wide range of tasks, with some of the most significant applications including language modeling, machine translation, and question answering. For example, in machine translation, transformer models have achieved remarkable results, outperforming previous state-of-the-art methods in many cases. In question-answering tasks, it demonstrated remarkable performance in answering questions based on large text corpora.

2.4 Transfer learning

Supervised models are typically trained to perform well on new data of the same task and domain ($\mathbb{A}$). However, when faced with a new domain ($\mathbb{B}$) lacking labeled data, training a new model from scratch becomes difficult. Transfer learning addresses this challenge by utilizing related data from a source task and domain to transfer knowledge to the target task

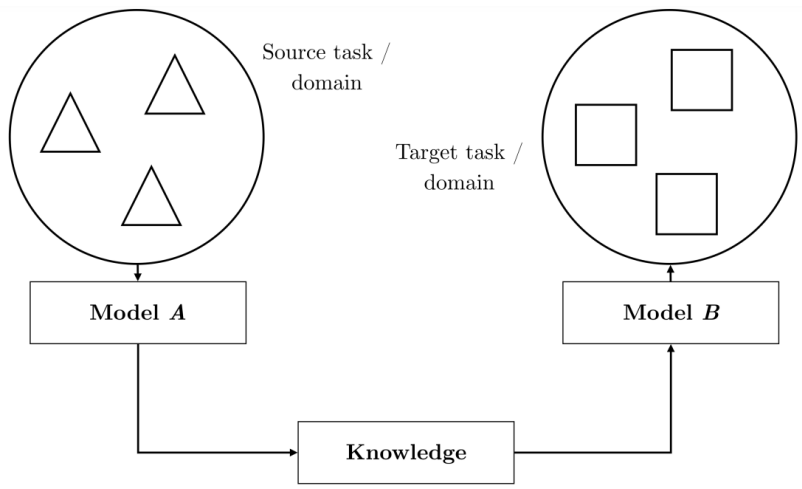


FIGURE 2.4: Transfer learning approach (Ruder (2019))

and domain (Figure 2.4).

In transfer learning, the objective is to transfer as much relevant information as possible from the source environment to the target task or domain. The type of knowledge that can be transferred varies depending on the task and data. A domain $\mathbb{D}$ is comprised of a feature space $\mathbb{X}$ and a marginal probability distribution $P(X)$ over the feature space, where $\boldsymbol{X} = \boldsymbol{x_1}, \ldots, \boldsymbol{x_n} \in \mathbb{X}$. Given a domain $\mathbb{D} = (\mathbb{X}, P(\boldsymbol{X}))$, a task $\mathbb{T}$ includes a label space $\mathbb{Z}$, a prior distribution $P(\boldsymbol{Z})$, and a conditional probability distribution $P(\boldsymbol{Z}|\boldsymbol{X})$, $\mathbb{T} = (\mathbb{Z}, P(\boldsymbol{Z}), P(\boldsymbol{Z}|\boldsymbol{X}))$. The $P(\boldsymbol{Z}|\boldsymbol{X}))$ distribution is typically learned from training data consisting of pairs $\boldsymbol{x_i} \in \boldsymbol{X}$ and $\boldsymbol{z_i} \in \boldsymbol{Z}$ Pan and Yang (2010).

Transfer learning aims to utilize information gained from a source domain $\mathbb{D}_\mathbb{S}$ and a source task $\mathbb{T}_\mathbb{S}$ to learn the target conditional probability distribution $P_T(\boldsymbol{Z_T}|\boldsymbol{X_T})$ in a target domain $\mathbb{D}_\mathbb{T}$ and a target task $\mathbb{T}_\mathbb{T}$. This becomes required either when there are limited labeled target examples or when there is a large number of unlabeled target examples available. Five transfer learning scenarios can arise (Figure 2.5), given the different properties of the source and target domains and tasks (Ruder (2019)). The initial three cases indicate an inconsistency between the source and target tasks, signifying that $\mathbb{T}_\mathbb{S} \neq \mathbb{T}_\mathbb{T}$. On the other hand, the final two cases arise due to a difference between the source and target domains,

implying that $\mathbb{D}_{\mathbb{S}} \neq \mathbb{D}_{\mathbb{T}}$.

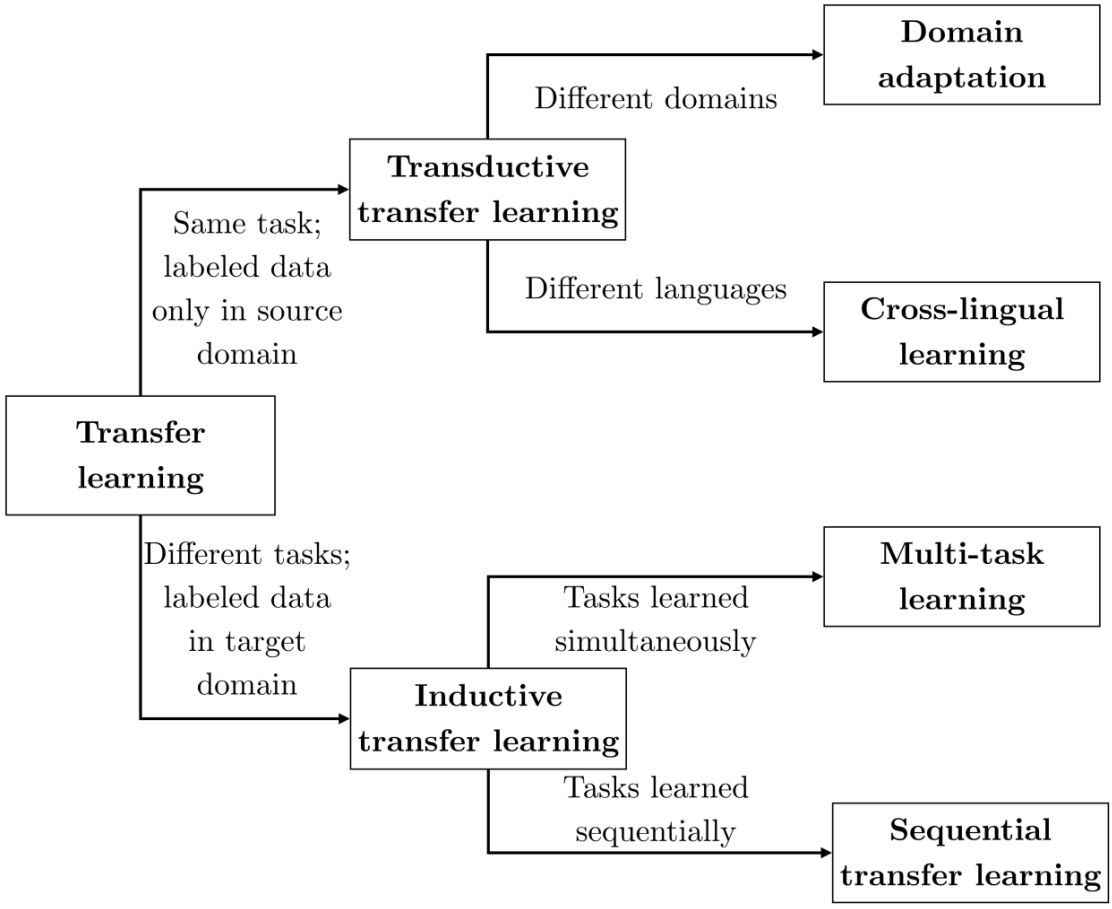


FIGURE 2.5: Transfer learning categorization (Ruder (2019))

1. **Different prior distributions between the source and target tasks**: $P_S(\boldsymbol{Z_S}) \neq P_T(\boldsymbol{Y_T})$. This occurs when the datasets have different label distributions.

2. **Different conditional probability distributions between the source and target tasks**: $P_S(\boldsymbol{Y_S}|\boldsymbol{X_S}) \neq P_T(\boldsymbol{Y_T}|\boldsymbol{X_T})$. This occurs when the source and target datasets are unbalanced with regards to their classes.

3. **The label spaces of the source and target tasks are different**: $\mathbb{Y}_{\mathbb{S}} \neq \mathbb{Y}_{\mathbb{T}}$. This occurs when source and target data have a different set of labels.

4. **The marginal probability distributions of the source and target domains are different**: $P_S(\boldsymbol{X_S}) \neq P_T(\boldsymbol{X_T})$. This occurs when datasets come from different domains, named domain adaptation.

5. **The feature spaces of the source and target domains are different**: $\mathbb{X}_\mathbb{S} \neq \mathbb{X}_\mathbb{T}$. This occurs when datasets come from different modalities (cross-modal learning) or languages (cross-lingual learning).

In inductive transfer learning, the source task is different from the target task, whereas, in transductive transfer learning, the source and target tasks are identical (Figure 2.5).

2.4.1 Pretraining

When choosing a pretraining task, it should capture properties useful for multiple target tasks. Access to a large amount of data is crucial in pretraining the model effectively. The main factor that sets apart different pretraining tasks is the source of supervision. They can be categorized into three types, in terms of ease of obtaining the training data:

- **No supervision**: This is the easiest to obtain as it only requires a raw form of data.
- **Distant supervision**: Distant supervision uses heuristics and domain knowledge to automatically gather large amounts of noisy supervised data.
- **Traditional supervision**: This requires manual labeling of each training example and is the most labor-intensive method.

Self-supervised learning is similar to distant supervision in that both require expertise in the domain to define the way the labels are generated. However, the main difference is that self-supervised learning automatically generates labels for all the training examples, whereas, in distant supervision, only specific instances are labeled based on predefined patterns and the labels can be noisy.

Unsupervised pretraining better resembles human learning and it does not require large amounts of labeled examples. It allows the model to capture broader aspects of language structure and meaning that can be more easily transferred (Erhan et al. (2010); Le (2013)). A model is trained on a large amount of unlabelled data, with the aim of learning general features that can be applied to various downstream tasks. The idea is that by training a model on a large and diverse set of unlabelled data, it can learn underlying patterns and structures in the data that can be useful for a variety of tasks.

Distantly supervised pretraining The concept of distantly supervised pretraining was initially proposed in relation to relation extraction models, where sentences containing entity pairs involved in a known Freebase relation were utilized as examples of that relation (Mintz et al. (2009)). Another approach to distantly supervised pretraining was presented by Ratner et al. (2016), which involved developing labeling functions capable of generating noisy labels for a vast amount of unlabeled data. The noisy labels are then used to train a generative model that can learn to aggregate the outputs of the labeling functions and generate high-quality labels for the unlabelled data.

Distantly supervised pretraining resembles the concept of auxiliary tasks in multi-task learning, acting as hints for the model. This leads the model to focus on smaller parts of the input that unsupervised models could overlook. Distant supervision can also capture discourse markers that reveal inter-sentence relations, making the model more attuned to the relationships between sentences and improving its understanding of sentence meaning.

Supervised pretraining Compared to distantly supervised methods, which use specific heuristics for a certain category of tasks, supervised pretraining involves utilizing existing tasks and datasets. Sometimes, for a specific downstream task, it is possible to choose existing tasks that are appropriate. For example, in Zoph et al. (2016), a machine translation model was trained on a high-resource language pair and then applied to a low-resource language pair. More recent methods use various supervised tasks with large datasets to learn general-purpose representations. These tasks are selected based on their demand for

general knowledge of the data structure.

2.4.2 Lifelong Learning

Lifelong learning in machine learning is a method in which models are continually trained to learn and adapt to new information even after deployment. This is different from traditional machine learning where models are trained on a fixed dataset and can't be updated after deployment. The ability to continuously learn and adapt to new information improves the model's performance and helps it to handle changing environments (Van de Ven and Tolias (2019); Parisi et al. (2019); Silver et al. (2013)).

In lifelong learning, the model is trained using a stream of incoming data, which allows it to refine its understanding and update its parameters. This enables the model to respond to changing patterns in the data and evolve its behavior based on new information. For instance, in speech recognition, a lifelong learning model would be able to learn new accents or dialects as they appear, without the need to retrain the whole model from scratch.

The significance of lifelong learning becomes apparent in scenarios where the distribution of data changes or new goals arise. In such situations, models trained using conventional methods will quickly become obsolete and ineffective, whereas models that possess the capability of lifelong learning can adjust and maintain a high level of performance.

Lifelong learning is a rapidly developing field in machine learning and has various potential applications in various domains such as healthcare, robotics and many more.

2.4.3 Meta-learning

Meta-learning, also referred to as "learning to learn," is a branch of machine learning that aims to create algorithms that can continuously adapt and improve their performance on various tasks through experience and exposure to different problems and environments. The objective of meta-learning is to design models that can quickly learn new tasks without having to be completely retrained every time. By using prior experiences with related tasks,

meta-learners can learn new tasks more efficiently by taking advantage of information and knowledge gained from previous experiences (Finn and Levine (2017); Munkhdalai and Yu (2017)).

The objective of meta-learning is to create machine learning algorithms that can evolve and enhance their performance on various tasks over time, through exposure to various problems and environments. The purpose is to build models that are capable of quickly adapting to new tasks without having to be entirely retrained, by leveraging their previous experiences and knowledge gained from similar problems. Meta-learning has applications in different areas of machine learning such as deep learning, reinforcement learning, and few-shot learning.

2.4.4 Zero-shot and Few-shot Learning

Zero-shot learning and few-shot learning are machine learning techniques that allow models to generalize to unseen classes or examples.

In zero-shot learning, the model is trained on a set of classes or examples and then tested on new, unseen classes or examples. The model must perform this task without ever having seen the new classes or examples during training. This is achieved through the use of auxiliary information or side information, such as textual descriptions or attributes, that can be used to predict the unseen classes.

Few-shot learning focuses on learning to recognize new classes or categories with very limited data. Traditional machine learning algorithms typically require large amounts of labeled data for training, whereas few-shot learning algorithms can generalize to new categories with just a few examples. This is achieved by learning to extract relevant information from the small amount of labeled data available, and then using this information to recognize new examples (Snell et al. (2017); Baik et al. (2021)).

Mathematically, both zero-shot and few-shot learning are usually framed as classification problems. In zero-shot learning, the goal is to predict the class label for a new example

based on the auxiliary information. This can be represented as a function $f(\boldsymbol{x}, \boldsymbol{a})$, where $\boldsymbol{x}$ is the input data and $\boldsymbol{a}$ is the auxiliary information. In few-shot learning, the goal is to predict the class label for a new example based on a small number of training examples. This can be represented as a function $f(\boldsymbol{x}, \boldsymbol{S})$, where $\boldsymbol{S}$ is the set of training examples.

The model optimizes a loss function that measures how well the model's predictions match the true labels of the training data. The loss function can be a cross-entropy loss for classification problems or a mean squared error for regression problems.

2.4.5 Multi-task Learning

Multi-task learning is a type of machine learning where multiple tasks are learned concurrently with the goal of improving performance on each of the individual tasks compared to learning them independently. The idea behind multi-task learning is that learning-related tasks simultaneously can lead to more efficient and effective learning, as knowledge gained from one task can help improve the learning of another task (Caruana (1998); Crawshaw (2020)).

Multi-task ML learning is formulated as a joint optimization problem with the goal of finding a set of parameters that simultaneously minimize the loss functions associated with each task. For example, consider a scenario where there are two tasks, task A and task B, and two corresponding loss functions, L_A and L_B. The objective of multi-task learning can be written as:

$$minimize(w)(L_A(w) + L_B(w)) \tag{2.34}$$

where w represents the parameters of the model. In this formulation, the goal is to find a set of parameters that minimize the sum of the losses associated with both tasks. The model parameters are shared across tasks, meaning that they are updated in response to the gradients of both loss functions.

In practice, multi-task learning can be implemented using neural networks, where different tasks are assigned different output units, and the model parameters are shared across

all tasks.

2.4.6 Sequential Transfer Learning

Sequential transfer learning focuses on the development of algorithms that can learn from previously acquired knowledge and experiences, and apply this knowledge to new and related tasks. Traditional machine learning approaches train a model from scratch on a fixed dataset. This requires large amounts of training data and can be time-consuming and computationally expensive. Transfer learning provides a way to overcome these limitations by leveraging knowledge acquired from previous tasks to learn new tasks more efficiently. In sequential transfer learning, the learning process is sequential, with the model gradually adapting to new tasks and learning from its experiences (Zamir et al. (2018); Mao (2020)).

There are several methods for sequential transfer learning, including fine-tuning, incremental learning, and meta-learning. Fine-tuning involves using a pretrained model as a starting point and fine-tuning the model on new data for a related task. Incremental learning involves training the model on new data for a related task, while retaining the knowledge acquired from previous tasks. Meta-learning involves training a model to learn how to learn new tasks more efficiently by exploiting knowledge gained from previous experiences.

Evaluating the performance of sequential transfer learning models can be challenging, as it often involves comparing the performance of the model on a new task with its performance on a similar task in the past. A common evaluation metric is transfer efficiency, which measures the improvement in performance achieved by the model through transfer learning.

2.5 Natural Language Processing

Natural language processing (NLP) teaches computers to comprehend natural language. It can be defined by its tasks, which usually map a text to a vector space and uses that representation to predict different properties and structures within the text. Machine learning

techniques are applied to develop a mathematical representation of this mapping. Specifically, the aim is to train a model that maps from an input $\boldsymbol{x}$, a sequence of words, to an output y by maximizing our model's conditional probability $P_\theta(y|\boldsymbol{x})$ through maximum likelihood estimation. The parameters θ are optimized in this process. Our focus is on discriminative models, which directly estimate the conditional probability from raw data, as opposed to generative models that learn the joint probability distribution $P(\boldsymbol{x}, y)$.

Bag-of-Words (BOW): In some cases, a text is represented as a bag-of-words $\boldsymbol{x} \in \mathbb{R}^{|}V|$, where V is the vocabulary. The value of $\boldsymbol{x_i}$ corresponds to the number of times the i-th word in the vocabulary appears in the text. This frequency may be adjusted with tf-idf, which considers the importance of a term in a collection of texts known as a corpus. An n-gram is a sequence of n words in a text. In the standard BOW model, only single words, unigrams, are considered. Bigrams (trigrams), sequences of two (three) words, may also be included in newer versions of BOW. The BOW approach does not take grammar or word order into account. To capture compositional relationships and dependencies in the input, we will primarily represent sequences of words, $w_1, ..., w_T$, as sequences of corresponding word embeddings $\boldsymbol{x_1}, ..., \boldsymbol{x_T}$ (Jurafsky and Martin (2023)). We list below the tasks used in this thesis's later chapters.

Topic classification (text classification) categorizes contiguous sequences of words, such as sentences or documents. Topic classification focuses on assigning topics relevant to a specific dataset, often news articles, and tends to rely heavily on keywords. This task is a fundamental problem in natural language processing (NLP) and has a wide range of applications, such as email filtering, sentiment analysis, and content recommendation.

To train a text classification model, one needs to have a labeled dataset of documents, where each document is assigned a predefined category. Then, a supervised learning algorithm is used to learn a model that can predict the category of new documents based on their features. Traditional models based on bag-of-words (BOW) with term frequency-inverse document frequency (tf-idf) weighted unigram, and bigram features were commonly used

as effective starting points. Recently, word2vec (Mikolov et al. (2013b)) or GloVe (Pennington et al. (2014)) embeddings have been commonly used as input embedding.

To classify texts in a simple way, one can compute the posterior probability of each text category using the naive Bayes formula and then predict the category with the highest probability:

$$\hat{c} = argmax_c P(c|d) = argmax_c \frac{P(d|c)P(c)}{P(d)} \tag{2.35}$$

where c is a category, d is the document, $argmax_c$ denotes the category that maximizes the probability, given $P(d|c) = P(\boldsymbol{x_1}|c)P(\boldsymbol{x_2}|c)...P(\boldsymbol{x_{|V|}}|c)$ - likelihood of a document given the category, and $P(c)$ - the prior probability of the class estimated from the training data. The probability of the document, $P(d)$ can be skipped, because the formula is calculated for each category c, but $P(d)$ does not depend on the class change (Jurafsky and Martin (2023); Manning and Schutze (1999); Manning (2008)).

Sentiment analysis, also known as opinion mining, is automatically determining the emotion expressed in a given text, typically classified as positive, negative, or neutral. This technique is widely used in social media monitoring, brand management, and market research. Sentiment analysis can be performed on different levels, such as at the document, sentence, or aspect level. The task is commonly applied to tweets or movie and product reviews, with star ratings often serving as a proxy for sentiment in review domains. Sentiment analysis can be performed using various techniques, including rule-based methods, machine learning algorithms, and deep learning models (Zhao et al. (2016); Zhang et al. (2018a)).

The simple way of computing the probability of a document belonging to a particular class, given its features, is the Naive Bayes algorithm which uses Bayes' theorem, stating:

$$P(class|features) = \frac{P(features|class) * P(class)}{P(features)} \tag{2.36}$$

where $P(class|features)$ is the probability of the document belonging to the class,

given its features, $P(features|class)$ is the probability of the features given the class, $P(class)$ is the prior probability of the class, and $P(features)$ is the probability of the features. In the context of sentiment analysis, one can calculate the probability of a document belonging to the positive or negative class, given its features (i.e., words), using the following formula:

$$P(class|features) = P(w_1|class) * P(w_2|class) * ... * P(w_n|class) * P(class) \quad (2.37)$$

where w_1, w_2, ..., w_n are the words in the document, and $P(w_i|class)$ is the probability of the i-th word given the class. To calculate $P(w_i|class)$, one can use the bag-of-words model, which represents the document as a vector of word frequencies. Specifically, one can calculate the probability of a word given the class as follows:

$$P(w_i|class) = \frac{count(w_i, class) + 1}{count(class) + V} \quad (2.38)$$

where $count(w_i, class)$ is the number of times the word appears in documents of the given class in the training set and 1 is a smoothing factor to avoid zero probabilities.

To classify a new document as positive or negative, one can compare the probabilities of the document belonging to each class and select the class with the highest probability.

$$\begin{aligned} P(positive|features) &= \frac{P(features|positive) * P(positive)}{P(features)} \\ P(negative|features) &= \frac{P(features|negative) * P(negative)}{P(features)} \end{aligned} \quad (2.39)$$

Language modeling is a crucial task in natural language processing (NLP), which involves predicting the likelihood of a sequence of words occurring in a given context. It is considered an unsupervised learning problem, as it only requires access to the raw text. Despite its simplicity, language modeling is crucial for many NLP advancements and it is a fundamental component of various NLP applications, including speech recognition, machine translation, and text summarization.

Language modeling aims to estimate the probability distribution of a sequence of words, given the previous words in the sequence. This probability distribution, known as the language model, assigns a probability to each possible next word in the sequence. Formally, given a sequence of words $w = (w_1, w_2, ..., w_t)$, the language model assigns a probability $P(w_{t+1}|w_1, w_2, ..., w_t)$ to the next word w_{t+1} in the sequence.

One of the most common approaches to language modeling is n-gram modeling, where the probability of the next word is estimated based on the previous $n - 1$ words in the sequence. Specifically, the probability is calculated using the following formula:

$$P(w_{t+1}|w_1, w_2, ..., w_t) = P(w_{t+1}|w_{t-n+1}, w_{t-n+2}, ..., w_t) \tag{2.40}$$

where n is the order of the n-gram model. For example, a bigram model ($n = 2$) estimates the probability of the next word based on the previous two words in the sequence.

However, n-gram models suffer from the sparsity problem, where many n-grams in the test set do not appear in the training set, resulting in zero probabilities. To overcome this problem, smoothing techniques are used to assign non-zero probabilities to unseen n-grams. One popular smoothing technique is the add-k smoothing, where a constant k is added to the counts of each n-gram to obtain smoothed probabilities. The formula for add-k smoothing is as follows:

$$P(w_t|w_{t-n+1}, w_{t-n+2}, ..., w_{t-1}) = \frac{count(w_{t-n+1}, w_{t-n+2}, ..., w_t) + k}{count(w_{t-n+1}, w_{t-n+2}, ..., w_{t-1}) + k * V} \tag{2.41}$$

where k is a smoothing parameter.

In recent years, neural network-based language models, such as the recurrent neural network (RNN) and the transformer models achieved state-of-the-art performance on various NLP tasks. These models learn to predict the next word in the sequence based on the context of the previous words, and can be adapted to solve many tasks by adding a new layer on top of the language model (Goodfellow et al. (2016); Devlin et al. (2019); Vaswani et al. (2017)).

2.5.1 *Zipf Law*

Zipf's law is a principle in linguistics and information science that states that the frequency of a given word in a text is inversely proportional to its rank in a list of the words that appear in that text, sorted by their frequency of appearance. Zipf's law has been observed to hold for many different languages and text corpora, and is used for modeling a wide range of phenomena in fields such as linguistics, information science, and communication engineering. Despite its empirical success, Zipf's law's exact origin and explanation is still a matter of active research.

2.5.2 *Traditional Word Representations*

Traditional word representations refer to the ways in which words are represented in natural language processing and computational linguistics. These representations are typically based on simple models such as one-hot encoding, bag-of-words, or term frequency-inverse document frequency (TF-IDF). They are widely used as input features for text-based machine learning models, but have limitations, such as a lack of semantic meaning and the inability to capture relationships between words. Newer representations, such as word embeddings and transformers, have been developed to address these limitations and have shown improved performance on a wide range of NLP tasks.

Latent Semantic Analysis (LSA) is a widely used method for learning dense word representations (Deerwester et al. (1990)). The method starts with a sparse word-word co-occurrence matrix (C) obtained from a corpus, which is then transformed into a pointwise mutual information (PMI) matrix (P). The PMI matrix is then factorized using singular value decomposition (SVD), yielding three matrices: U, Σ, and V. The word embedding matrix (X) is created by reducing the word representations to k dimensions by taking the top k singular values from Σ and the corresponding columns from U. This form of dimensionality reduction enables the use of co-occurrence counts as features in various applications. LSA is closely related to canonical correlation analysis (CCA) which has also

been used for learning word representations.

Brown clustering (Brown et al. (1992)) is a well-known unsupervised pretraining technique that tackles the issue of sparse word features by assigning each word to one of C classes. Using a class-based bigram model, the conditional probability of a word following another word is broken down into class transition probabilities, $P(c_{i+1}|c_i)$, and word emission probability, $P(w_{i+1}|c_i)$. The authors use a greedy hierarchical clustering algorithm to find a hard class assignment for each word, resulting in a binary tree with words as its leafs. The path to each word, made up of concatenated binary codes, is then used as a feature in downstream tasks. Brown clusters have shown to group words with similar morphological stems and related words with different stems.

Latent Dirichlet Allocation (LDA) is a generative probabilistic model introduced by Blei et al. (2003), represents each topic in a document as a mixture of latent topics, each represented as a distribution of words. The parameters of LDA are typically estimated through Gibbs sampling. The primary purpose of LDA is to uncover the topics in a corpus, but it can also serve as a representation of a document for text classification or as word representations via the distribution of topics for each word.

LDA is a unique unsupervised pretraining technique that combines elements of both language models and matrix factorization methods. While it can be viewed as a form of dimensionality reduction like matrix factorization, its use of latent variables makes it similar to HMMs, often used as language models.

Hidden Markov Models (HMM) are statistical models that are commonly used in various fields such as speech recognition, bioinformatics, and natural language processing. HMMs are based on the assumption that there is a hidden process that generates a sequence of observations, and the goal is to infer the underlying state sequence that produces these observations. The basic concept of an HMM is that there is a set of N states, each of which generates an observation with a certain probability. These probabilities are represented by the emission probabilities $\boldsymbol{B_{ij}}$, which describe the likelihood of observing a particular

observation given a state. HMMs also have a set of transition probabilities $\boldsymbol{A_{ij}}$, which describe the likelihood of moving from one state to another.

Formally, an HMM can be defined as a tuple $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$, where $\boldsymbol{A}$ is the $N \times N$ transition probability matrix, $\boldsymbol{B}$ is the $N \times M$ emission probability matrix, and $\boldsymbol{\pi}$ is the initial state distribution. The initial state distribution $\boldsymbol{\pi}$ is a vector of length N that describes the probability of starting in each state.

In HMMs, the states are not directly observable (hidden states), but the observations are. Therefore, the goal is to infer the most likely sequence of states that generated the observed sequence of observations. This is done using the forward-backward algorithm, which computes the probabilities of being in each state at each time step given the observed sequence of observations using dynamic programming algorithm. The probability of the observed sequence of observations is given by:

$$P(O|\lambda) = \sum_{s_1} \sum_{s_2} \ldots \sum_{s_T} P(s_1, s_2, \ldots, s_T, O|\lambda) \tag{2.42}$$

where λ represents the model parameters, s_t represents the hidden state at time t, and O represents the observed sequence.

The forward-backward algorithm computes the forward probabilities $\alpha_t(i)$ and backward probabilities $\beta_t(i)$, which represent the probability of being in state i at time t given the observed sequence of observations up to time t. The forward probabilities are computed as follows:

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = i|\lambda) \tag{2.43}$$

where $o_1, o_2, ..., o_t$ are the observed sequence of observations up to time t, and $q_t = i$ is the event that the hidden state at time t is i. The backward probabilities are computed as follows:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T|q_t = i, \lambda) \tag{2.44}$$

where $o_{t+1}, o_{t+2}, ..., o_T$ are the observed sequence of observations after time t. The probabilities can be computed recursively as follows:

$$\begin{aligned} \alpha_1(i) &= \pi_i B_{i,o_1} \\ \alpha_t(i) &= \sum_{j=1}^{N} \alpha_{t-1}(j) A_{j,i} B_{i,o_t} \\ \beta_T(i) &= 1 \\ \beta_t(i) &= \sum_{j=1}^{N} A_{i,j} B_{j,o_{t+1}} \beta_{t+1}(j) \end{aligned} \tag{2.45}$$

The base case is $\beta_T(i) = 1$ for all states i.

The forward-backward probability $\gamma_t(i)$ is the probability of being in state i at time t given the observations up to time T. It is computed as follows:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O|\lambda)}. \tag{2.46}$$

Pretrained Hidden Markov Models (HMM): Although not as popular as current pretraining language models, HMM language models have been used for pretraining in sequence labeling. In Huang and Yates (2009), a latent variable HMM was trained on unlabeled data to learn the chance of a given word appearing before and after the other word. The learned probabilities were then utilized as features for the left and right context in a sequence labeling model, and a dense representation was obtained through LSA.

2.5.3 Word Embeddings

The previous approaches learned word representations which were then used in downstream models. With the rise of neural networks, the prevalent method of obtaining dense word representations is through word embeddings. These embeddings are preferred due to their compactness, dense structure, and expressive power as compared to the traditional approaches, as mentioned in Baroni et al. (2014). However, classic approaches such as LSA

can also induce dense, competitive representations with more recent neural approaches when used under similar settings (Levy et al. (2015)).

Collobert and Weston (2008) introduced a method for learning word embeddings, CW model. They trained a neural network on a corpus C to score a correct word sequence higher than an incorrect one using a max-margin loss function:

$$L_{MMHL} = |C| - \sum_{i=C} \sum_{w_0 \in V} max\left(0, 1 - f\left([w_{i-C}, ..., w_i, ..., w_{i+C}]\right) + f\left([w_{i-C}, ..., w_0, ..., w_{i+C}]\right)\right) \tag{2.47}$$

The outer summation in the equation goes through all the words in the corpus C, while the inner summation goes through all the words in the vocabulary V. A word sequence consists of a central word w_i and C words on its left and right side. The function $f(\cdot)$ outputs a score for a given word sequence and is trained to produce a higher score for a word sequence present in the corpus (left term) than for a sequence where the central word is substituted with an arbitrary word w_0 from the vocabulary (right term).

Skip-gram with negative sampling (SGNS) Mikolov et al. (2013b,a) is a widely-used method for learning word embeddings. It is preferred for its training efficiency and robustness compared to other methods. Instead of accurately predicting word probabilities like a language model, SGNS focuses on learning efficient word representations. It trains representations that can predict surrounding context words given a target word. The goal is to minimize the negative log-likelihood of the training data using the skip-gram objective. The skip-gram is a neural network lacking a hidden layer. The word embedding x_i of the input word w_i serves as the model's hidden state. This embedding is then utilized in a softmax layer, where each word has its own representation $\tilde{x}_i$ that depicts its role in the context of other input words.

Negative sampling trains a model to distinguish a target word w_t from negative samples chosen from a noise distribution P_n. It functions similarly to the CW model, which ranks real sentences above false ones. Negative sampling does not actually minimize the negative

log-likelihood, but instead factors a shifted PMI matrix, similar to LSA.

Continuous Bag-of-Words (CBOW) architecture is the reverse of the skip-gram model, aiming to predict the target word w_t from a window of context words by minimizing the CBOW objective. Like skip-gram, CBOW is often trained using negative sampling. Various modifications to both SGNS and CBOW models have been proposed. Le and Mikolov (Le and Mikolov, 2014) enhanced both models by incorporating a vector for representing paragraphs and documents, which is used in either center word or surrounding context prediction, depending on the specific model.

Global Vectors (GloVe) is a word embedding method that uses matrix factorization to represent words as vectors in a high-dimensional space. It was introduced by Pennington et al. (2014) and has been shown to achieve state-of-the-art performance on various natural language processing tasks, including word similarity and analogy tasks, sentiment analysis, and machine translation.

GloVe aims to capture the distributional semantics of words, which means that words that occur in similar contexts should have similar vector representations. It achieves this by minimizing the difference between the dot product of the embeddings of a word w_i and its context word w_j, and the logarithm of their co-occurrence frequency within a specific window size.

Let $\boldsymbol{C}$ be a matrix of word co-occurrence counts, where each element C_ij represents the number of times word i co-occurs with word j in a given corpus. The goal of GloVe is to find a matrix of word embeddings $\boldsymbol{X}$ and a matrix of context word embeddings $\hat{\boldsymbol{X}} such that the dot product of \mathrm{x}_i$ and $\hat{x_j}$ captures the co-occurrence information between words i and j in C. Specifically, the objective of GloVe can be expressed as follows:

$$L_{\text{GloVe}} = \sum_{i,j \in V} f(C_{ij})(x_i^T \tilde{x}_j + b_i + \tilde{b}j - \log Cij)^2 \tag{2.48}$$

where b_i and $\hat{b_j}$ are the biases for the word i and its context word j, and $f()$ is a weighting function scales down the co-occurrence count of frequent words while amplifying the

counts of less frequent words.

The weighting function "f(·)" is defined as:

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{2.49}$$

where α is a hyperparameter that controls the weight assigned to different co-occurrence frequencies, and x_{max} is a threshold that prevents the weighting function from assigning too much weight to very frequent co-occurrences.

To minimize the objective function L_{GloVe}, GloVe uses stochastic gradient descent (SGD) to update the parameters of the word embeddings and context word embeddings. By iteratively updating the parameters using SGD, GloVe learns word embeddings and context word embeddings that capture the distributional semantics of words in a given corpus.

One advantage of GloVe over other methods is that it incorporates both global and local word co-occurrence statistics. Global statistics capture the overall co-occurrence patterns of words across the entire corpus, while local statistics focus on the co-occurrence patterns within a specific window of words. This allows GloVe to capture both the semantic and syntactic information of words, making it particularly useful for downstream NLP tasks.

GloVe also addresses some of the limitations of earlier methods, such as the skip-gram and CBOW models. For example, the skip-gram model may struggle with rare words or words that have multiple senses, as it treats each occurrence of a word as independent. In contrast, GloVe uses the logarithm of the co-occurrence frequency, which effectively down-weights the influence of common words and up-weights the influence of rare words.

Furthermore, GloVe allows for efficient training on large corpora, as it factorizes the co-occurrence matrix using singular value decomposition (SVD). This reduces the dimensionality of the matrix and removes noise while preserving the most important information. The resulting word embeddings are more compact and efficient to store, making them ideal for use in large-scale NLP applications.

2.5.4 Deep Pretrained Representations

The neural network-based methods presented so far are simple models that prioritize efficiency over expressiveness. To make them scalable, they use a neural network without a hidden layer, which limits the relationships they can capture. Word embeddings can be useful for many tasks, but they lack the ability to capture higher-level information that could be even more valuable. A model initialized with word embeddings must learn from scratch to disambiguate words and derive meaning from sequences of words, which is the core of language understanding and requires modeling complex language phenomena. This is why NLP models initialized with shallow word embeddings still require a large number of examples to perform well.

The foundation of recent breakthroughs in NLP is a major shift in strategy: from just initializing the first layer of models to pretraining the entire model with hierarchical representations. This shift is like moving from learning only the edges of an image to learning the full hierarchy of features, from edges to shapes to high-level semantic concepts. As a result, these models employ deep neural networks to learn representations and the hidden state of the model can be used to obtain document representations easily.

Kiros et al. (2015) introduced skip-thoughts, a technique that trains an RNN to predict the words in the surrounding sentences. This approach extends SGNS to the sentence level and requires training on longer documents that are comprised of ordered sentences.

The skip-thoughts method requires a long time to train. Therefore, Hill et al. (2016) suggested a faster method that sums the input the word embeddings, while Jernite et al. (2017) proposed cost-effective objectives based on predicting whether two sentences have been swapped. The task of next-sentence prediction is treated as a classification problem and it is used in BERT Devlin et al. (2019) as one of the objectives. In Jernite et al. (2017), the model is presented with the first three sentences of a paragraph and is required to select the next sentence from a set of five possible options later in the paragraph. BERT Devlin

et al. (2019) on the other hand only predicts if one sentence follows another.

Advancements in language modeling have led to the development of deep neural language models, which were previously restricted by computational resources, forcing the use of approximations to the more complex models. Although the first proposal for pretraining a language model was put forward by Dai and Le (2015), this method was limited to in-domain documents. Radford et al. (2018) further improved the model by training it on larger data sets. Devlin et al. (2019) introduced a masked language model objective that conceals certain words in the input and then predicts only the masked words. Compared to a denoising autoencoder, this method reconstructs only the masked words and not the entire input. Language modeling is more efficient regarding sample usage and particularly effective in syntactic tasks.

The widespread use of pretrained language models in NLP has far-reaching implications for transfer learning. The models have delivered improvement rates of 10-30% on various tasks, some of which have been researched for years.

Pretrained language models improve NLP performance and capture hierarchical features similar to those captured by ImageNet models in computer vision. They also can aid domain adaptation. The word embedding layer in pretrained language models is responsible for capturing morphological information, while the lower layers capture local syntax and the upper layers capture longer-range semantics such as coreference. Additionally, these models have demonstrated the ability to implicitly learn logic rules.

Simple sentence and document representation approaches, such as averaging pretrained word embeddings, can be surprisingly effective. Shallow models are quick to train, but recent advancements indicate that deeper models are more effective. LSTMs outperform word vector averages on paraphrase pretraining tasks. Conneau et al. (2017) found that a BiLSTM with max-pooling condensed hidden representations performed best in their experimentation with different models. The Transformer, an architecture based on self-attention, has been proven to be successful with recent advancements in the field, as evi-

denced by studies conducted by Radford et al. (2018); Devlin et al. (2019).

There are many recent pretrained methods in NLP that can be used for transfer learning and domain adaptation. Those are often called **generalized language models**:

- **Contextual Word Vectors (CoVe)**, is a form of word embeddings that are learned by an encoder in an attentional seq-to-seq machine translation model, as introduced by McCann et al. (2017). In contrast to the traditional word embeddings discussed earlier, CoVe word representations are influenced by the entire input sentence, making them contextual.

- **Embeddings from Language Model** (ELMo) (Peters et al. (2018)) is a deep contextualized word embedding technique that learns vector representations of words by taking into account their context within a sentence. It is based on a deep bidirectional language model pretrained on a large corpus of text data. The resulting word representations capture not only the inherent meaning of the word itself but also its meaning in context, which makes ELMo embeddings highly effective for a wide range of NLP tasks, including sentiment analysis, question answering, and machine translation.

- **Universal Language Model Fine-tuning** (ULMFiT) is a transfer learning-based approach for text classification tasks introduced by Howard and Ruder (2018). It is built on the concept of pretraining a language model on a large corpus of text and then fine-tuning it on a smaller task-specific dataset. The pretraining is performed using an unsupervised language model like ULMFiT, which learns the next word in a sequence given the information from the previous words. The model is then fine-tuned using supervised learning on a smaller labeled dataset. One of the key contributions of ULMFiT is its novel approach to fine-tuning, where the model is first fine-tuned on the task-specific dataset at the end layers and then fine-tuned on the earlier layers with a lower learning rate. This approach allows the model to retain

its pretrained knowledge while still adapting to the new task. ULMFiT has achieved state-of-the-art results on a variety of text classification tasks, including sentiment analysis, question classification, and language identification.

- **Generative Pretraining Transformer** (GPT)(Radford et al. (2018)) is based on a similar idea to ELMo, in that it is an unsupervised language model. However, GPT takes this concept to a much larger scale by training on a vast collection of free text corpora. Despite the similarities between the two models, there are two significant differences between them.

- **Bidirectional Encoder Representations from Transformers** (BERT) (Devlin et al. (2019)), builds upon the success of GPT. It first pretrains a large language model on unannotated text, and then fine-tunes it for specific tasks without requiring customized network architectures. The key innovation of BERT is its bi-directional training approach, which allows the model to capture context from both left and right directions. According to the authors' ablation study, this bi-directional nature is the most significant contribution of the model. Compared to GPT, BERT substantially improves language understanding and has been widely used in various natural language processing applications.

- **A Lite BERT** (ALBERT) is a variant of the BERT model designed to be more lightweight and efficient (Lan et al. (2019)). ALBERT achieves a 1.7x speedup in training time and 18x reduction in parameters compared to a similarly configured BERT model. The improvements in efficiency are achieved through three changes. The first two modifications reduce the number of parameters and memory usage, leading to faster training. The third change involves a more challenging training task that replaces the next sentence prediction objective used in BERT.

- **GPT-2** language model is the next iteration of GPT and features a massive parameter

count of 1.5 billion, which is ten times more than its predecessor. GPT-2 achieved state-of-the-art results on seven of the eight tested language modeling datasets without fine-tuning for any specific tasks. The pretraining dataset for GPT-2 includes eight million web pages collected by crawling Reddit's qualified outbound links. This model's large improvements are especially noticeable in small datasets and datasets used for measuring long-term dependencies.

- **Robustly Optimized BERT Approach** (RoBERTa) (Liu et al. (2019)) is a new method for training the BERT model that aims to improve its performance. The authors of the method observed that the original BERT model was significantly undertrained and thus proposed the following improvements:

  - training for longer with bigger batch size,
  - removing the next sentence prediction task,
  - using longer sequences in training data format, and
  - changing the masking pattern dynamically.

  Additionally, RoBERTa incorporated a new dataset, CommonCrawl News[1], and confirmed that pretraining with more data can enhance performance on downstream tasks. The model was trained using **byte-pair encoding** (BPE) on byte sequences, similar to GPT-2, and the authors found that hyperparameter choices had a significant impact on the model's performance.

- **Text-to-Text Transfer Transformer** T5 (Raffel et al. (2020)), is a language model that follows the Transformer architecture with an encoder-decoder implementation of tokens, embeddings, encoder, decoder, and output. The model adopts the "Natural Language Decathlon" framework (McCann et al. (2018)) where multiple NLP tasks are translated into question-answering format. However, unlike the explicit QA

[1] https://commoncrawl.org/2016/10/news-dataset-available/

format, T5 utilizes short task prefixes to differentiate between tasks and fine-tunes the model separately for each individual task. The text-to-text framework allows for easier transfer learning evaluation of the same model on a diverse set of tasks.

- **GPT-3** (Brown et al. (2020)) shares the same architecture with GPT-2 but boasts a remarkable increase in scale, with a whopping 175B parameters, 10 times more than GPT-2 (1.5B). To make it work across multiple GPUs, GPT-3 uses a combination of dense and locally banded sparse attention patterns, similar to the sparse transformer. The training data is a filtered version of Common Crawl and other curated high-quality datasets. The authors made an attempt to avoid downstream tasks' contamination in the training data by removing any overlaps with the benchmark dataset. However, there were still some imperfections due to a bug.

- **XLNet** is a language model that combines the advantages of both autoencoder and autoregressive approaches by using a permutation language modeling objective Yang et al. (2019). Autoregressive (AR) models, such as GPT, and autoencoder (AE) models, such as BERT, are two common approaches to language modeling. However, AR models do not consider bidirectional context, which is necessary for downstream tasks like reading comprehension, while AE models assume that masked positions are independent given all other unmasked tokens, oversimplifying long context dependencies. XLNet overcomes these limitations by generalizing the AE method to incorporate the advantages of AR. XLNet introduces the permutation language modeling objective, which samples a factorization order for a text sequence and decomposes the likelihood using a factorization. The model can consider all possible permutations of the input sequence, effectively capturing bidirectional context and long-range dependencies.

- **BART** (Lewis et al. (2019)), is a denoising autoencoder that aims to reconstruct the original text from a randomly corrupted version by training a bidirectional encoder

similar to BERT and an autoregressive decoder similar to GPT, and minimizing the negative log-likelihood loss, with text infilling and sentence shuffling being the most effective noising approaches after experimenting with various transformations.

- **ELECTRA** (Clark et al. (2020)) seeks to enhance pretraining efficiency by proposing a novel pretraining task called "Replaced Token Detection" (RTD), which replaces a randomly sampled set of tokens in the original text with a plausible alternative predicted by a small language model (known as the generator) and trains a discriminator to predict whether each token is original or replaced, framing the language modeling as a discrimination task instead of a generation task to save computation resources.

A summary of all these models with more details about each of them is given in Lilian Weng's blog [2]. In the next few subsections we will give more details about models used in the later chapters of this thesis.

2.5.5 Universal Language Model Fine-tuning for Text Classification

Universal Language Model Fine-tuning (ULMFiT) is a versatile transfer learning technique that can be applied to any NLP task (Howard and Ruder (2018)). The key idea behind ULMFiT is to leverage pretrained language models on large corpora to improve performance on downstream NLP tasks. This method includes essential techniques for fine-tuning a language model and has shown remarkable results. It can produce results equivalent to training from scratch using 100 times more data with only 100 labeled examples.

ULMFiT is composed of three steps:

1. general-domain language model (LM) pretraining;
2. target task LM fine-tuning;

[2] https://lilianweng.github.io/posts/2019-01-31-lm/

3. target task classifier fine-tuning.

The first step involves training the LM on a large general-domain corpus to capture general language features at different layers. The pretraining step is similar to training a neural language model where the goal is to predict the next word given the information from previous words. In step two, the entire LM is fine-tuned on the smaller corpus of text related to a specific target task data using techniques such as discriminative fine-tuning and slanted triangular learning rates (STLR) to gain task-specific features. Finally, in step three, the classifier is fine-tuned on the target task using a combination of gradual unfreezing, discriminative fine-tuning, and STLR, to preserve low-level representations while adapting high-level ones.

The pretraining step is performed using a variant of the AWD-LSTM architecture, a recurrent neural network type. The AWD-LSTM is trained using semi-supervised learning, where the model is trained on labeled and unlabeled data. This approach helps capture more general language patterns and improves the model's performance on downstream tasks. ULMFiT is pretrained on the Wikitext-103 corpus, a large dataset containing 103 million words from 28,595 cleaned Wikipedia articles, as described in Merity et al. (2017). Pretraining is particularly useful for tasks with limited data and can facilitate generalization even when there are only a few labeled examples available.

The fine-tuning stage involves adapting the pretrained language model to the specific task. The target task data is likely to come from a different distribution than the general-domain data used for pretraining. Fine-tuning the LM on the target task data can converge faster and improve performance even with small datasets. ULMFiT uses discriminative fine-tuning and slanted triangular learning rates to adapt the LM to the target task data. Discriminative fine-tuning is a novel method that adjusts the fine-tuning intensity of different layers based on the information they capture.

Discriminative fine-tuning allows different layers of the model to be fine-tuned with

different learning rates, rather than using the same learning rate for all layers. The regular stochastic gradient descent (SGD) updates a model's parameters θ at each time step t as $\theta_t = \theta_{t-1} - \eta * \nabla_\theta J(\theta)$, where η is the learning rate and $\nabla_\theta J(\theta)$ is the gradient of the model's objective function. In discriminative fine-tuning, the parameters θ are divided into $\theta_1, ..., \theta_L$, with θ_l containing the parameters of the l-th layer of the model. Each layer is then updated with its own learning rate, $\eta_{(1)}, ..., \eta_{(L)}$. The SGD update with discriminative fine-tuning becomes $\theta(l)_t = \theta(l)_{t-1} - \eta(l) * \nabla\theta(l)J(\theta)$.

Slanted Triangular Learning Rates (STLR): To optimize parameters to task-specific features, it is ideal for the model to converge quickly to a suitable parameter space at the beginning of training and then refine its parameters. Using a constant or annealed learning rate throughout training may not produce optimal results. Thus, the ULMFiT paper proposes STLR which first increases linearly and then decreases linearly. The STLR approach enhances the triangular learning rate strategy Smith (2017) by including a steep rise followed by a prolonged fall.

For fine-tuning the classifier in ULMFiT system, the pretrained language model is enhanced with two additional linear blocks. Each block utilizes batch normalization and dropout, with ReLU activation for the intermediate layer and a softmax activation for the last layer, which outputs the probability distribution for target classes. Only the parameters in these task-specific classifier layers are learned from scratch. The input for the first linear layer is the pooled last hidden layer state.

In text classification tasks, significant information is often contained in a few keywords, which may appear anywhere in the document. As documents can consist of hundreds of words, relevant information can be lost if only the last hidden state of the model is considered. To address this issue, ULMFiT concatenates the hidden state at the final time step of the document, h_T, with both the max-pooled and mean-pooled hidden states representation over the time steps that fit in the memory of GPU. The final concatenated representation, h_c, is defined as $[h_T, maxpool(H), meanpool(H)]$, where [] represents concatenation.

Fine-tuning the classifier is a crucial step in transfer learning. Overzealous fine-tuning can result in catastrophic forgetting, negating the advantages gained from language modeling. On the other hand, cautious fine-tuning may result in slow convergence and overfitting. ULMFiT proposes gradual unfreezing besides discriminative fine-tuning and triangular learning rates.

Gradual unfreezing prevents catastrophic forgetting by unfreezing the layers of the model gradually, starting with the last layer, which contains the least general knowledge. First, the last layer is unfrozen, and all unfrozen layers are fine-tuned for one epoch. Then, the next lower frozen layer is unfrozen, and the process is repeated until all layers are fine-tuned to convergence.

2.5.6 BERT and RoBERTa

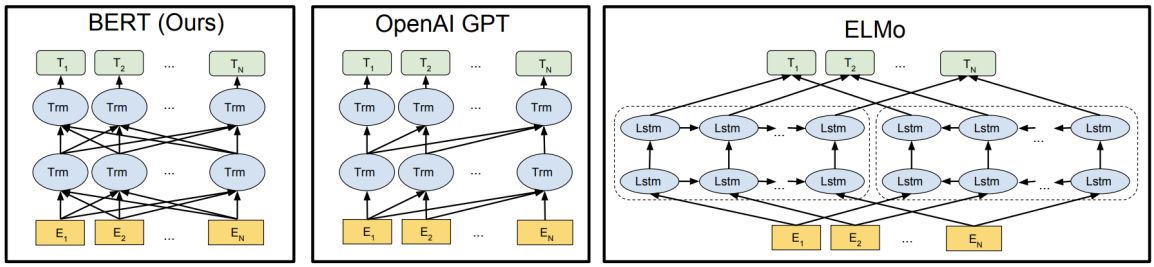


FIGURE 2.6: Architectuures of BERT, GPT, and ELMO (Devlin et al. (2019))

Bidirectional Encoder Representations from Transformers (BERT) is a pretrained deep learning model for Natural Language Processing (NLP) tasks. Figure 2.6 compares architectures of BERT, GPT and ELMO. The main principles of BERT are as follows:

1. **Bidirectionality**: BERT is a bidirectional model, which means it takes into account the context of the words before and after the target word when encoding the input text. This allows BERT to better understand the meaning of words in context.

2. **Transformer architecture**: BERT is based on the transformer architecture Vaswani et al. (2017), a neural network architecture that allows for parallel processing of

sequences, which is important for NLP tasks.

3. **Pretraining**: BERT is pretrained on large amounts of text data, allowing it to learn general language representations that can be fine-tuned for specific NLP tasks. The pretraining process involves training the model to predict masked words in a sentence and to predict the next sentence given a prompt.

4. **Fine-tuning**: BERT can be fine-tuned for specific NLP tasks by adding task-specific layers to the model and training on task-specific data. This fine-tuning process allows BERT to adapt its pretrained representations for the specific task at hand.

5. **Attention mechanism**: BERT uses an attention mechanism to focus on the most relevant parts of the input when encoding the text. This allows the model to better understand the relationships between words in a sentence and to better capture the meaning of the text.

The pretraining process of BERT involves learning general language representations by training the model on a large corpus of text data. During this process, the model learns to predict missing words, **masked language modeling** (MLM), and predict the next sentence given a sequence of sentences **next sentence prediction** (NSP). This pretraining step allows BERT to learn a rich set of representations for words, phrases, and sentences, which can be fine-tuned for specific NLP tasks like sentiment analysis, named entity recognition, or question answering.

In MLM, BERT randomly masks 15% of the tokens in each sequence to encourage the model to learn the context around the words. BERT uses several heuristics to avoid the special placeholder token [MASK] never being encountered during fine-tuning. Specifically, with 80% probability, the chosen words are replaced with [MASK], with 10% probability replaced with a random word, and with 10% probability, they are kept the same. The model only predicts the missing words without any information on which words have been

replaced or which words should be predicted. The MLM task can be expressed mathematically as: Let $\boldsymbol{X}$ be the input sequence, and $\boldsymbol{Y}$ be the output sequence with the masked tokens predicted. Then the probability of the output sequence given the input sequence is:

$$P(Y|X) = \prod_{i=1}^{n} P(\boldsymbol{y_i}|\boldsymbol{x_i}) \tag{2.50}$$

where n is the number of masked tokens and $\boldsymbol{y_i}$ is the predicted token for the i-th masked token.

In NSP, BERT trains a binary classifier to predict whether a given sentence is the next sentence of another sentence. BERT samples sentence pairs $(\boldsymbol{A}, \boldsymbol{B})$ such that 50% of the time $\boldsymbol{B}$ follows $\boldsymbol{A}$, and 50% of the time $\boldsymbol{B}$ does not follow $\boldsymbol{A}$. The model processes both sentences and outputs a binary label indicating whether $\boldsymbol{B}$ is the next sentence of $\boldsymbol{A}$. The NSP task can be expressed mathematically as: Let $\boldsymbol{S_A}$ and $\boldsymbol{S_B}$ be two input sentences. Then the probability of $\boldsymbol{B}$ being the next sentence of $\boldsymbol{A}$ is:

$$P(NSP = 1|\boldsymbol{S_A}, \boldsymbol{S_B}) = \sigma(\boldsymbol{W} \cdot [\boldsymbol{H_{CLS}}; \boldsymbol{H_B}] + \boldsymbol{b}) \tag{2.51}$$

where $\boldsymbol{H_{CLS}}$ is the output of the special [CLS] token for the input sequence, $\boldsymbol{H_B}$ is the output of the [CLS] token for the second sentence, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias vector of the binary classifier, and σ is the sigmoid activation function.

The training loss for BERT is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood. The training data for both auxiliary tasks can be generated from any monolingual corpus, which unbounded the scale of training.

Input Embedding: The input embedding for BERT (Figure 2.7) can be represented as the sum of three components:

- **WordPiece tokenization embeddings**: BERT uses the WordPiece model to further divide English words into smaller sub-word units, which makes it more effective

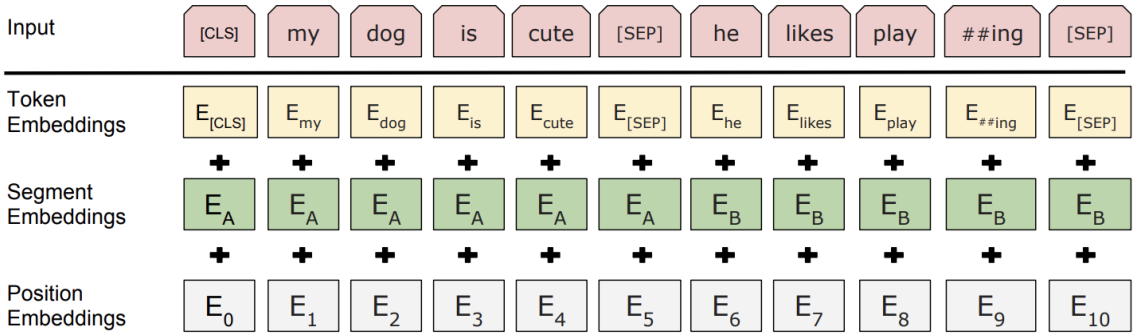


FIGURE 2.7: BERT input representation (Devlin et al. (2019))

to handle rare or unknown words. The tokenization process assigns an embedding vector to each token in the input sequence.

- **Segment embeddings**: If the input sequence contains two sentences, BERT assigns segment embeddings to distinguish the two sentences. The sentence $\boldsymbol{A}$ and sentence $\boldsymbol{B}$ embeddings are separated by a special token [SEP]. If the input sequence contains only one sentence, BERT assigns segment embeddings to sentence $\boldsymbol{A}$ only.

- **Position embeddings**: BERT uses positional embeddings to capture the relative position of each token in the input sequence. These embeddings are learned rather than hard-coded, allowing BERT to handle input sequences of different lengths.

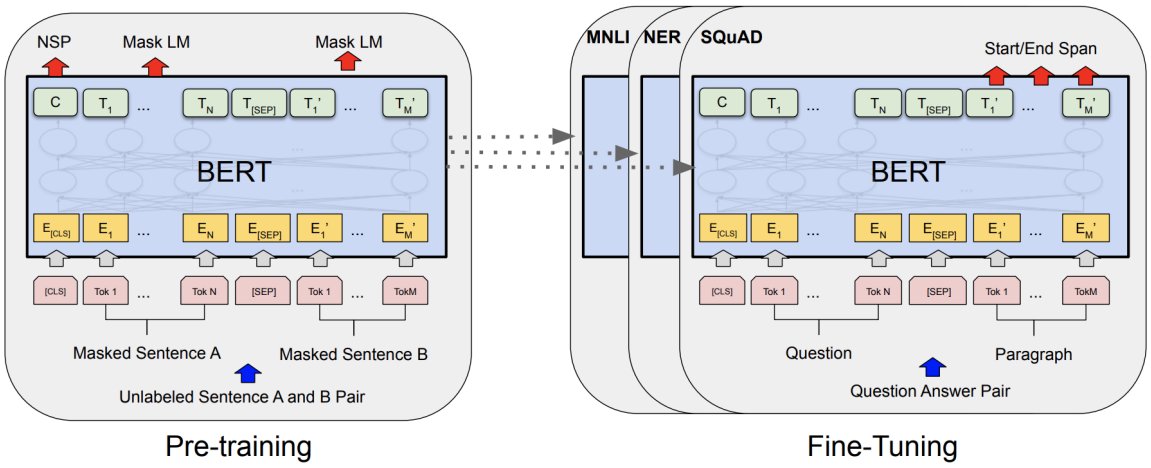


FIGURE 2.8: BERT fine-tuning (Devlin et al. (2019))

To fine-tune BERT for downstream tasks (Figure 2.8), only a few new parameters need to be added, similar to OpenAI GPT. For classification tasks, BERT predicts the label by taking the final hidden state of the special [CLS] token and multiplying it with a small weight matrix, $\boldsymbol{W_c}$. For question answering tasks, BERT predicts the start and end positions of the answer span by outputting two probability distributions over the tokens in the input sequence. Two small weight matrices, $\boldsymbol{W_s}$ and $\boldsymbol{W_e}$, are learned during fine-tuning to define these probability distributions.

Overall, the modifications required for fine-tuning BERT for downstream tasks are minimal and involve only one or two weight matrices to convert the transformer hidden states to an interpretable format.

RoBustly Optimized BERT Approach (RoBERTa) is a modified version of BERT model. The main differences between BERT and RoBERTa are:

- **Pretraining corpus**: BERT was trained on the BooksCorpus and English Wikipedia, while RoBERTa was trained on a much larger corpus, called the WebText, which contains a diverse range of web pages and texts.

- **Pretraining objective**: BERT used the masked language modeling objective, where some tokens in the input sequence are masked and the model is trained to predict the original token based on the context. RoBERTa uses a variation of the masked language modeling objective that emphasizes the importance of the loss function and encourages the model to attend to more diverse context.

- **Training parameters**: RoBERTa uses a larger batch size, longer sequence length, and more training steps than BERT. These changes help RoBERTa achieve better performance in various NLP tasks.

- **Dropout rate**: RoBERTa reduces the dropout rate from BERT, which helps the model to capture more information from the input sequences.

RoBERTa was designed to improve the performance of BERT by using a larger training corpus and optimizing the training process, which has led to improved results on various NLP tasks.

2.5.7 GPT-2, GPT-3, and GPT-chat

Generative Pretrained Transformer 2 (GPT-2) is an advanced language model that surpasses its predecessor GPT in both size and performance. With 1.5 billion parameters, 10 times more than GPT, GPT-2 achieved state-of-the-art results on 7 out of 8 language modeling datasets without any task-specific fine-tuning. The model is trained on a dataset of 8 million web pages sourced from outbound links on Reddit. The main principles of GPT-2 are:

- **Pretraining**: GPT-2 is pretrained on a large corpus of text, allowing it to learn general patterns in language. This pretraining step is crucial to the success of GPT-2, as it allows the model to gain a deep understanding of the language before fine-tuning for specific tasks.
- **Transformer architecture**: GPT-2 uses a transformer architecture, which is a deep neural network that is well-suited for processing sequences of data. This architecture allows the model to effectively capture long-range dependencies in language.
- **Generative model**: GPT-2 is a generative model, meaning that it can generate new text based on the input it receives. The goal of GPT-2 is to predict the next word in a sequence, given the context of the previous words.
- **Fine-tuning**: GPT-2 can be fine-tuned for specific tasks, such as text classification or question-answering, by fine-tuning the pretrained weights with task-specific data.
- **Unsupervised learning**: GPT-2 is trained in an unsupervised manner, meaning that it is trained on text data without any explicit supervision. This unsupervised learning

allows the model to learn from a wide variety of sources, making it more flexible and robust.

GPT-2 uses **zero-shot transfer learning**, where downstream tasks are framed as predicting conditional probabilities without any task-specific fine-tuning. This approach enables GPT-2 to excel at a variety of language tasks, including text generation, machine translation, QA, and summarization. For example, machine translation is achieved by conditioning the language model on pairs of English and Chinese sentences, while QA tasks are formatted with pairs of questions and answers. Summarization is induced by adding a ”TL;DR” summary to the end of the article.

GPT-2 uses **Byte Pair Encoding (BPE)** on UTF-8 byte sequences to reduce the vocabulary size needed for efficient language modeling. This allows the model to assign a probability to any Unicode string, without the need for preprocessing or tokenization. GPT-2 improves upon BPE by preventing the generation of multiple versions of common words and increasing the quality of byte segmentation. Despite the benefits, current byte-level language models still lag behind word-level models in performance.

GPT-2, compared to GPT, has several architecture modifications along with a much larger number of transformer layers and parameters:

- Layer normalization was moved to the input of each sub-block, similar to a residual unit of type ”building block” instead of type ”bottleneck”. Batch normalization is also applied before weight layers.
- An additional normalization layer was added after the final self-attention block.
- A modified initialization was created as a function of the model depth.
- The weights of residual layers were initially scaled by a factor of $\sqrt{\frac{1}{N}}$ where N is the number of residual layers.

- GPT-2 uses a larger vocabulary size and context size.

Generative Pretrained Transformer 3 (GPT-3) is the next version of GPT-2. Some of the key differences between GPT-3 and GPT-2 include:

- **Model size**: GPT-3 is much larger than GPT-2, with 175 billion parameters as opposed to GPT-2's 1.5 billion parameters. This increased size allows GPT-3 to capture much more nuanced patterns in the data it has been trained on.

- **Training data**: GPT-3 has been trained on a larger and more diverse corpus of text data, including web pages, books, and scientific articles, compared to GPT-2 which was trained on a smaller corpus of web pages.

- **Improved performance**: GPT-3 has shown remarkable improvements in its ability to generate text, answer questions, summarize documents, and perform other language-related tasks compared to GPT-2.

- **Flexibility**: GPT-3 has the ability to perform a achieve higher accuracy in various range of tasks and activities, compared to GPT-2, which was primarily designed for text generation and language-related tasks.

GPT-3 represents an advance over GPT-2 in terms of size, training data, performance, and flexibility, making it one of the most powerful language models to date. **GPT-Chat** is a modification of GPT-3 model designed for conversational AI.GPT-3 LM is fine-tuned on a large dataset of conversations and is optimized for generating human-like text in a conversational context. This fine-tuning allows GPT-Chat to perform better in tasks that involve generating responses in a conversational setting, such as customer service, chatbots, and more.

2.6 Machine learning for medicine and healthcare

Machine learning has emerged as a powerful tool for solving problems in various domains, including healthcare. The healthcare industry has a vast amount of data generated every day through patient records, medical images, and wearable devices. Machine learning algorithms have the potential to analyze this large amount of data and identify patterns, predict outcomes, and make informed decisions.

One of the most promising applications of machine learning in healthcare is diagnosis. Medical diagnosis is a complex task that requires expertise and experience, and machine learning can assist in this process by providing accurate and efficient diagnosis predictions. For instance, algorithms can analyze medical images, such as X-rays and MRI scans, and detect anomalies and diseases. This can help healthcare providers make more informed decisions, reduce the risk of misdiagnosis, and improve patient outcomes.

Machine learning algorithms can also be used for pattern recognition in other types of medical data, such as electronic health records. For example, a deep learning algorithm may be trained on a large database of patient records to identify risk factors for a particular disease, such as age, gender, or family history. Algorithms can also look into patterns that lead to disease worsening and based on that predict how likely the patient is to get complications and comorbidities. This information can then be used to make personalized recommendations for preventative care or early intervention.

Deep learning algorithms can be trained to understand and interpret written medical reports and patient histories using NLP methods, which can then be used to make predictions about a patient's health. For example, a deep learning model trained on written medical reports may be able to identify patterns that indicate a patient is at risk of a particular disease, such as diabetes or heart disease.

Machine learning algorithms can analyze a patient's medical history, genetic data, and other relevant information to develop a customized treatment plan. This approach takes into

account a patient's individual needs and characteristics and can improve the effectiveness of treatment and reduce the risk of adverse side effects.

The process of developing new drugs can take many years and cost billions of dollars. Machine learning algorithms can analyze vast amounts of data from various sources, such as chemical databases and clinical trials, to identify potential drug targets and predict the likelihood of success. This can speed up the drug discovery process and reduce the cost of developing new drugs. Deep learning algorithms can be trained on large amounts of genomic and chemical data to predict the likelihood of a drug being effective for a particular patient based on their unique genetic profile. This information can be used to develop targeted therapies and personalized treatment plans, leading to more effective and efficient healthcare outcomes.

Machine learning can also be used to improve patient outcomes by predicting and preventing adverse events. Algorithms can analyze patient data, including vital signs, medical history, and demographics, to predict the risk of adverse events, such as hospital readmissions. This information can be used to intervene and prevent adverse events, improving patient outcomes and reducing healthcare costs.

In the upcoming chapters and other work published during the PhD studies, we propose new techniques to advance performance of machine learning algorithms for medical applications. We extensively write about machine learning for cognitive and mental health (CMH) applications in Chapter 6, which is why next subsection describes ML for CMH in more details.

2.6.1 Machine Learning for Cognitive and Mental Health

Machine learning is used for the analysis of neuroimaging data, such as magnetic resonance imaging (MRI) scans or functional magnetic resonance imaging (fMRI) scans. Deep learning algorithms can be trained to identify patterns in brain activity and anatomy associated with various mental health conditions, such as depression, anxiety, or schizophrenia. This

allows for the development of more accurate and objective diagnostic tools, which can help improve the accuracy and speed of diagnosis and treatment.

Personalized treatment recommendations for patients with mental health conditions can be recommended by machine learning algorithms. By analyzing large amounts of data about an individual's symptoms, medical history, and response to treatment, deep learning algorithms can identify the most effective treatments for that individual. This can help improve treatment speed and effectiveness and reduce the risk of side effects or negative outcomes Stanojevic et al. (2022).

Machine learning algorithms can be trained to identify patterns in speech Diep et al. (2022) and other forms of communication, such as text messages, that are indicative of depression or other mental health conditions. This can be useful for the early detection and monitoring of mental health conditions and for tracking treatment outcomes.

Deep learning models are created to improve our understanding of the underlying mechanisms of mental health conditions. By analyzing large amounts of data about the brain and behavior, deep learning algorithms can identify patterns and relationships that are not immediately obvious to human researchers. This can provide new insights into the causes and underlying mechanisms of mental health conditions, which can be used to develop new treatments and therapies.

2.7 Machine Learning for Biology

Machine learning has been used to analyze large amounts of biological data and extract meaningful insights. It would be impossible for human to identify analyse that amount of data and to understand all kinds of patterns that are easy for machine to find. One of the most prominent applications of machine learning in biology is in the analysis of genomic data to identify patterns and correlations between genetic variations and specific diseases. For example, machine learning algorithms have been used to identify genetic mutations associated with an increased risk of developing cancer, helping to identify individuals at

risk of developing the disease and allowing for early diagnosis and treatment. It is also used to better understand high-throughput sequencing technologies' complexity and vast data. It is helping to tackle various challenges in genomics such as gene prediction, variant calling, classification of genetic disorders, and gene regulatory network inference. ML algorithms are used to predict the function of genes based on their DNA sequences, gene expression data, and protein-protein interaction networks. This can help identify novel genes, improve gene annotation, and prioritize genes for functional validation.

Variant calling involves identifying differences in DNA sequences between individuals or populations. ML algorithms are used to improve the accuracy of variant calling, especially for rare and complex variants, and to predict the functional impact of variants on gene function. ML algorithms are trained on large genomic data and clinical information datasets to classify individuals into different disease categories.

Gene regulatory network inference involves identifying the interactions between genes and their regulation by other genes and environmental factors. Machine learning algorithms are used to analyze large amounts of genomic data, such as gene expression data, to infer the underlying gene regulatory network.

Proteins are essential for various biological processes such as metabolism, growth, and repair. Machine learning algorithms have been used to analyze the structures of proteins and predict their function, stability, and interactions, helping researchers to understand the molecular mechanisms underlying various diseases and design new treatments. The models can analyze large amounts of data and find patterns to help predict protein behavior. Predicting the structure and three-dimensional folding of a protein from its amino acid sequence is a challenging problem that is crucial for understanding protein function. The protein-protein interactions are key to understanding biological processes and are important for drug discovery and development. ML algorithms are trained on large datasets of known protein-protein interactions and can then be used to predict interactions between proteins that have not been studied before.

In addition to the analysis of genomic and protein data, machine learning has also been applied in the field of bioinformatics. Bioinformatics is the use of computational techniques to analyze and interpret biological data. Machine learning algorithms have been used in bioinformatics to develop predictive models of biological processes. In this thesis, we will focus on ML for phylogenetics.

The discipline of phylogenetics involves examining the evolutionary connections between different species. ML models are used to infer evolutionary relationships from molecular data such as DNA or protein sequences. There are several ways machine learning models are used in phylogenetics:

- **Sequence alignment**: Machine learning models are used to align sequences to maximize the similarities between homologous regions. This is a critical step in constructing a phylogenetic tree.

- **Phylogenetic tree construction**: Once sequences are aligned, machine learning models can be used to infer the evolutionary relationships among species based on the similarities and differences in their sequences. These models can be used to construct both maximum likelihood and Bayesian phylogenetic trees.

- **Model selection**: In phylogenetics, various models of molecular evolution are used to infer evolutionary relationships. Machine learning models can be used to compare different models and select the best one based on the data.

- **Phylogenetic inference under uncertainty**: Machine learning models can also quantify the uncertainty in phylogenetic estimates using bootstrapping or Bayesian methods.

- **Predicting functions of proteins**: Machine learning models can be used to predict the functions of proteins based on their evolutionary relationships with other proteins. This is particularly useful in cases where experimental data is limited or unavailable.

- **Finding relevant information**: Processing large amount of scientific articles to extract phylogenetics knowledge and timetrees. We discuss this application further in chapter 7

Machine learning has proven to be a valuable tool for biologists and bioinformaticians, providing new ways to analyze and interpret large amounts of biological data. By identifying patterns and correlations in this data, machine learning has the potential to help with understanding the implicit mechanisms of diseases. It can also help develop new treatments. However, as with any technology, machine learning in biology also has its limitations and challenges, including the need for high-quality data, the difficulty of interpretability, and the risk of overfitting.

2.8 Conclusion

This chapter serves as an introduction to the key concepts and background knowledge required to comprehend the forthcoming chapters. We have covered various topics, including probability, information theory, and machine learning, which are crucial for this understanding. Furthermore, we have introduced neural network methods, natural language processing tasks, and models, which will be extensively used throughout the thesis. In addition, we have provided an overview of how machine learning is applied in various fields, including NLP, medicine, healthcare, biology, and bioinformatics.

CHAPTER 3

INTRODUCTION TO DOMAIN ADAPTATION

This chapter focuses on domain adaptation, where the source and target domains are distinct ($\mathbb{D}_{\mathbb{S}} \neq \mathbb{D}_{\mathbb{T}}$), but the source and target tasks are the same ($\mathbb{T}_{\mathbb{S}} = \mathbb{T}_{\mathbb{T}}$). This is crucial in real-world scenarios. This study focuses on the four major categories of domain adaptation techniques in NLP:

- Representation-based methods
- Data weighting and selection techniques
- Self-labelling techniques
- Combining multiple source domains

In machine learning, training and test data are usually assumed to be independently and identically distributed (i.i.d.). However, this assumption becomes unreliable when the models are used in real-world scenarios, where the data distribution can be different from the one encountered during training. This leads to the problem of domain adaptation, which involves adapting the models from the training distribution to the test distribution.

Existing datasets often do not fully evaluate a model's robustness as they only assess the model's ability to interpolate, meaning it can perform well on examples similar to those seen in training with a large number of examples. However, to determine a model's true generalization ability, it must also be able to extrapolate to examples outside of its training distribution.

Domain adaptation is a field in machine learning that focuses on adapting a model trained on one dataset to perform well on a similar, but different dataset. This becomes necessary when the distribution of data or labes in the real world differs from the distribution seen during training. In domain adaptation, the goal is to learn representations that are beneficial for the target domain, unlike sequential transfer learning which seeks to learn representations that are useful in general.

There are two main scenarios in domain adaptation:

- unsupervised, where only unlabeled examples are available in the target domain and a sufficient number of labeled examples in the source domain, and
- supervised, where a small number of labeled examples are available in the target domain.

Representation approaches: Representation methods aim to alter the fundamental representation of data. These methods either search for common features in both domains based on their similarity, or aim to represent the data in a shared low-dimensional space. Recent advancements have combined both techniques.

Distribution similarity approaches: The central idea of distribution similarity methods is to make the source and target domain feature distributions as similar as possible. Many distribution similarity methods rely on a measure of the distance between the source and target domains. Minimizing the distance between the representations of two domains while maximizing performance on the source domain is a common strategy in distributional similarity approaches.

Latent feature learning decreases the dimensionality of the data, thereby increasing the similarity between the source and target domains, by utilizing the features of both domains. Singular Value Decomposition (SVD), Latent Dirichlet Allocation (LDA), and Canonical Correlation Analysis (CCA) are commonly used for this purpose. Blitzer et al. (2006) introduced Structural Correspondence Learning (SCL), a block-sparse multi-task learning method that utilizes multiple auxiliary classifiers to adapt to different domains. The classifiers predict pivot features and the weights of these classifiers are stored in a matrix. The selection of pivot features for SCL can be done either manually or automatically by considering their mutual information with the label. Once the pivot features are chosen, singular value decomposition (SVD) is applied to the matrix, and the top singular vectors are utilized as the representation for SCL.

Recently, the most impactful trend in the field has been the integration of distributional similarity approaches with latent feature learning techniques, aimed at minimizing the distance between source and target domains. Tzeng et al. (2014) was among the first to employ this approach, adding a bottleneck layer to their architecture and using regularization to minimize the Maximum Mean Discrepancy (MMD) between source and target domain representations.

Domain-adversarial loss minimizes the model's loss while simultaneously confounding the domain classifier, which is similar to the minimax game in generative adversarial networks Goodfellow et al. (2020). The goal is to minimize the Jensen–Shannon (JS) divergence between the source distribution (P_S) and the target distribution (P_T). Minimizing the Wasserstein distance between the source and target domain distributions is more stable and it achieves a similar outcome Shah et al. (2018).

Multi-task learning is often utilized in domain adaptation, such as by using a secondary domain classifier. SCL in particular uses multiple auxiliary classifiers.

Weighting and selecting data: Instead of assigning significance to specific features or constructing a new representation that emphasizes specific aspects of the domains, other

methods carry out weighting and selection on a per-instance basis.

Instance weighting: In domain adaptation, the goal is to minimize the expected error for examples from the target domain, expressed as:

$$J(\theta) = \sum_{i=1}^{n} \mathcal{P}_T(x, y) L(x, y, \theta) \tag{3.1}$$

However, labeled examples from the target domain are often not available. Usually, we have only examples from the source domain. To work around this, Equation 3.22 can be expanded as:

$$J(\theta) = \sum_{i=1}^{n} \frac{P_T(x, y)}{P_S(x, y)} P_S(x, y) L(x, y, \theta). \tag{3.2}$$

This can be re-written as an expectation over the empirical source domain distribution:

$$J(\theta) = \mathbb{E}_{x,y \sim P_S} \left[\frac{P_T(x, y)}{P_S(x, y)} L(x, y, \theta) \right]. \tag{3.3}$$

For a proper instance weighting approach in domain adaptation, each example from the source domain should be weighed by $\frac{P_T(x,y)}{P_S(x,y)}$. However, as the target domain examples $P_T(x, y)$ are not available to compute, computing this term accurately is unfeasible. To address this, existing methods make simplifying assumptions, such as assuming equal conditional probability distributions between source and target domains. Two categories of methods exist one that handles class imbalance and assumes $P_T(X|Y) = P_S(X|Y)$ and another that deals with covariate shift, where $P_T(Y|X) = P_S(Y|X)$. However, these methods only handle a weaker form of domain adaptation. When conditional probability distribution between source and target domains differ, instance weighting minimizes the difference between the weight-corrected source domain examples and the target domain examples.

Instance selection: Instance weighting techniques can be considered a form of ”soft” data selection, where examples are assigned a weight of 0, 1, or a value greater or less than

1 to indicate the degree of attention given to the example. In contrast, a ”hard” selection process is more efficient and easier to implement, as it simply disregards examples that may be harmful. Most data selection methods use instance weighting, first calculating a weight or relevance score for each example, and then setting a threshold to discard examples with a score below the threshold.

Data selection is a common practice in machine translation, with several studies dedicated to it. Instance weighting and instance selection are related to semi-supervised learning and specifically self-labeling approaches, which will be discussed in the following paragraph. Unlike instance weighting and selection, which are pre-processing steps, self-labeling approaches aim to select useful examples and estimate their labels jointly during training.

Self-labelling approaches: Semi-supervised learning, a method that involves learning from both labeled and unlabeled data, has a long history in machine learning. Refer to Van Engelen and Hoos (2020); Zhu (2005) for a comprehensive understanding. One specific type of semi-supervised learning is self-labeling, which trains a model using labeled examples and then uses that model to generate pseudo or proxy labels for unlabeled examples. In subsequent iterations, these labels are used to refine the model. While semi-supervised learning is generally applied in cases where the data is i.i.d., many methods have been adapted for cases where there is a domain shift.

Self-training is a basic approach to semi-supervised learning, first introduced by Yarowsky (1995). The technique relies on using the model's own predictions on unlabeled data to generate additional information for training. Confident predictions with a probability higher than a threshold are often selected and used as labeled examples for the next iteration. Despite its popularity, a major drawback of self-training is that the model is unable to correct its own mistakes, leading to an amplification of errors in the presence of a domain shift.

Expectation maximization (EM) algorithm has a close connection to self-training. EM can be applied in both ”soft” and ”hard” forms in domain adaptation. The distinction

between the two forms lies in that hard EM calculates a hard assignment of hidden variables during the E step, while soft EM calculates a distribution.

Multi-view training trains multiple models by using various perspectives of the data. These perspectives can vary in terms of the features utilized, the model architecture, or the data used for training. The goal is to have the models complement each other, leading to improved performance.

Co-training Blum and Mitchell (1998), is a well-known method of multi-view training. However, it imposes strong assumptions, including data representation using two conditionally independent feature sets, each sufficient to train a model. The process starts by training each model on its own feature set. Then, at each iteration, inputs confidently predicted by only one of the models are moved to the training set of the other model. One model can label inputs that the other model is uncertain about, effectively extending the co-training idea and reducing constraints. We extend co-training idea and reduce constraints needed in Chapter 4.

A multi-view training approach called **Democratic co-learning** is used to train models with different inductive biases, using either different network architectures for neural networks or entirely different learning algorithms.

Tri-training, a well-known multi-view training method derived from democratic co-learning, leverages the agreement of three independently trained models to improve predictions on unlabeled data. To ensure diversity among initial models, they can be trained on different variations of the original data obtained through bootstrap sampling. If two out of three models agree on the sample label, it is added to the training set of a model, and the process continues until the classifiers no longer change.

Multi-source Domain Adaptation is a type of unsupervised domain adaptation where data from multiple sources, not just one, is used for training. The simplest method is to merge all source data to train a single model (Stanojevic et al. (2022)). Alternatively, each source can be trained with its own model, which can then be combined using ensembling

techniques. We explore multi-source domain adaptation in Chapter 5.

Adversarial Training is a type of domain adaptation technique where a model is trained on a source domain and a discriminator is trained to differentiate between the source and target domains. The model is trained to fool the discriminator by generating domain-invariant features. Adversarial training can be effective when the source and target domains are very different.

Domain-Specific Preprocessing involves applying domain-specific transformations to the input data to make it more suitable for the target domain. For example, if the target domain has a different distribution than the source domain, domain-specific preprocessing techniques can be used to transform the data to match the target distribution.

Unsupervised Domain Adaptation (UDA) is a type of domain adaptation that does not require labeled data in the target domain. In UDA, the goal is to learn a model that can generalize well to the target domain by leveraging the labeled data in the source domain and the unlabeled data in the target domain. Recent work has shown that UDA can be used to improve the performance of deep learning models in several computer vision and natural language processing tasks.

Domain Generalization is another type of domain adaptation that aims to learn a model that can generalize well to unseen domains. In domain generalization, the goal is to learn a model that can capture the common patterns across multiple domains and can generalize to new domains. Recent work in domain generalization has focused on developing methods that can improve the robustness of deep learning models to variations in the input data.

Adapters (Houlsby et al. (2019) are small neural networks that are inserted between pre-trained and task-specific layers of a deep neural network. It allows fine-tuning of pre-trained models to new tasks while keeping the original model intact. They are designed to improve the transferability of pre-trained models by adding a small adapter layer to the model architecture, which can be trained with a small amount of labeled data for a specific

task. Adapters are trained only on the target task data and can adapt the pre-trained model to a new domain or task without requiring fine-tuning on the entire model. Adapters reduce the number of parameters that needs to be learned and speeds up training. Recent work has shown that adapters can achieve state-of-the-art results in several NLP tasks.

The mathematical background of deep learning adapter models involves linear transformations. The adapter layer in the model consists of a series of learnable parameters, such as weight matrices, that can be optimized through backpropagation to adjust the output of the pre-trained model for a specific task. The layer typically includes only a few parameters, making it computationally efficient to train and allowing for rapid adaptation to new tasks.

The mathematical formula for an adapter layer can be represented as follows: $\boldsymbol{h_t} = f(\boldsymbol{W_{adapter}} * \boldsymbol{h_p} + \boldsymbol{b_{adapter}})$ where $\boldsymbol{h_t}$ where is the output of the adapter layer, $\boldsymbol{h_p}$ is the output of the pre-trained model, $\boldsymbol{W_adapter}$ and $\boldsymbol{b_adapter}$ are the weight matrix and bias terms for the adapter layer, and f is an activation function. The weight matrix and bias terms are updated during training, allowing the adapter layer to learn to adjust the output of the pre-trained model for a specific task.

CHAPTER 4

ENHANCING PERFORMANCE ON TARGET DATA WITH LIMITED SAMPLES THROUGH DOMAIN ADAPTATION

Domain adaptation improves models' performance on target data, which distribution differs from the source distribution. This technique is particularly useful when the target data is scarce or has limited samples. It exploits the similarities between the source and target domains while adapting to the differences, even with a small amount of labeled target data (few-shot learning). In general, domain adaptation involves two steps: (1) finding a good representation of the data that captures the shared features between the domains and (2) adapting the model to the target domain using the shared features.

In this chapter, we propose a framework that is able to adapt to new unseen examples by using knowledge and context from samples whose label is easier to predict. It is designed to work with different NLP models to classify public opinion or other short texts. It can also be used in active learning or to transfer knowledge to samples from different sources. We measure our framework's performance when combined with different text representations and classification models. We also examine the number of labeled examples needed to achieve good performance.

4.1 Introduction

Obtaining a comprehensive understanding of public opinion is vital for effective decision-making, policymaking, and product development. With the proliferation of social media platforms, gathering vast amounts of text containing people's opinions on various subjects has become increasingly convenient. Social networks, blogs, forums, product reviews, and news comments serve as primary platforms for public debates, enabling individuals to share their viewpoints freely Otte and Rousseau (2002); Java et al. (2007); Watts and Dodds (2007); Pak and Paroubek (2010); Glynn (2018).

However, conducting traditional surveys to assess public opinion can be expensive and time-consuming, requiring the involvement of a large number of participants. Online data, on the other hand, presents an attractive alternative, but the inherent noise and bias in such text can be challenging to address. To mitigate these issues, researchers must carefully collect data to ensure relevant posts are included, and non-relevant ones are filtered out Buntain et al. (2018).

One significant challenge when using social media, news comments, forums, and product reviews to model public opinion is the brevity of the text. Moreover, such texts may include pictures, links, and emoticons that can significantly alter the tone or meaning. To overcome these challenges and achieve higher accuracy, recent studies have proposed text classification models that require hundreds of thousands or even millions of labeled documents Conneau et al. (2016). Building a model to represent public opinion in this way demands collecting massive amounts of data along with their corresponding labels, reflecting diverse perspectives on the same topic.

Labeling a large amount of data is a challenging and expensive process that academic and industry organizations often lack the resources to support. Even if resources are available, it may take years to label enough data for certain tasks, particularly when the labeling process is complex, or the content is very brief or has multiple interpretations that require

expert input. Automating the labeling process enables researchers to focus on modeling human behavior and opinions.

One approach to automating the labeling process is by clustering documents Steinbach et al. (2000); Rangrej et al. (2011); Yin and Wang (2014); Xu et al. (2015). However, document clustering may not necessarily cluster documents based on their views on a particular topic of interest, which could lead to clusters containing similar opinions.

In this study, we propose a semi-supervised approach, Semi-supervised Label Prediction (SLP), which predicts the labels of vast amounts of unlabeled documents based on a small seed of labeled documents. We also describe a method to quickly label the seed. Our approach addresses the problem of costly and time-consuming labeling and provides evidence that our model can predict labels more accurately than existing classification methods while maintaining the same time complexity.

Contributions of this study include the following:

1. A semi-supervised label prediction (SLP) algorithm is proposed to label big textual data using a small labeled seed.

2. A semi-automatic labeling framework is proposed to label the seed in order to decrease labeling effort for an order of magnitude.

3. SLP is evaluated on multiple datasets using a range of seed size and label prediction significance thresholds. It outperformed all existing solutions on all of the datasets considered.

4. The SLP framework is successfully applied to characterize public opinion on gun advocacy using possibly biased twitter posts of 200 gun control/rights advocates.

4.2 Related Work

The proposed approach's related work can be divided into three primary components: language modeling to convert documents to vectors, training with labeled document vectors for text classification, and semi-supervised label prediction.

In a character-level convolution model Zhang et al. (2015), it has been demonstrated that language can be modeled as a collection of character n-grams, and calculating tf-idf scores of those n-grams and performing logistic regression achieves the best results on classification tasks for smaller datasets (less than 600K examples). Therefore, we utilize this model as a part of our framework and as a baseline, referring to it as **tf-idf + n-grams**.

Word2vec Mikolov et al. (2013b) embeds words in a self-supervised manner by predicting the context around the word (skip-gram) or the word from the context (CBOW). It can better capture the syntax and semantic meaning of the words than tf-idf scores of words. Global vectors for word representation (**GloVe**) Pennington et al. (2014) captures the meaningful linear substructures through matrix decomposition of a word-word concurrence matrix and performs even better in understanding syntax and semantic meaning. However, both of these models require additional steps to create a document vector, such as word vector averaging, which can lower their power and heavily depends on the text content, length, and cleaning process. The **Doc2vec** algorithm is capable of learning representations of larger blocks of text (documents) along with learning representations of words in it Le and Mikolov (2014). Nonetheless, our experiments indicate that Doc2vec performs poorly on short and domain-specific texts.

There are several algorithms that aim to improve task accuracy by integrally modeling and classifying texts in an end-to-end approach. Some solutions utilize a long short-term memory (LSTM) approach, either unidirectional or bidirectional, on characters or words Zhou et al. (2016). Others use multi-task learning to improve accuracy Liu et al. (2016). Bidirectional LSTM (bi-LSTM) approaches perform well on sequential data since both

the previous and future context are important. However, with longer input sequences, the number of weights to learn becomes large, requiring more data and training time, and gradients issues start to affect model performance.

To address these issues, researchers turned to convolutional neural networks (CNNs). The character-level convolutional network achieves very good accuracy for datasets with over 600K examples Zhang et al. (2015). Even better results were obtained using a combination of convolutional layers and bi-LSTM to learn from context Xiao and Cho (2016). In this approach, the convolutional layer learns higher level features, which produces a short sequence that is then used as input to the bi-LSTM. Similar to how very deep CNN (VDCNN) was beneficial in computer vision, a similar method is applied to text Conneau et al. (2016). VDCNN consists of convolutional blocks, where each block contains two convolutional layers followed by batch normalization and ReLU. The output of each block can be connected with its input, similar to ResNet He et al. (2016). This architecture achieved better results than all previous methods for datasets with over 500K examples. While all of these methods can be a part of our framework, we use VDCNN as it has the best performance and as our baseline.

Recently, attention layers have been incorporated into deep learning models. For instance, the hierarchical attention network (**HAN**) Yang et al. (2016) combines bidirectional GRU units with five components. The sequence encoder maps words/sentences to vectors and generates annotations of words/sentences by aggregating information from the forward and backward hidden states. The word/sentence level attention layer extracts important words/sentences and aggregates them into sentence/document vectors, which are then classified in the final step. Another model, **fastText**, utilizes weighted word embedding to classify documents and achieves similar performance as previous models, but is significantly faster due to hashing. We use fastText as a component and baseline in our framework. Additionally, the **StarSpace** model is an extension of fastText that learns entity embeddings with discrete feature representation using additional entities besides text,

which has shown slight improvements in classification results. If additional information beyond the text is available, StarSpace can be used as part of our framework Wu et al. (2017).

Semi-supervised label prediction. There have been several traditional semi-supervised label prediction models, including co-training Blum and Mitchell (1998), naive bias classifier, and expectation-maximization Nigam et al. (2000). However, these models are designed to work with longer texts and require additional information or appropriate mathematical representation to optimize. In recent years, newer methods based on matrix factorization have been proposed Dharmadhikari et al. (2012), but they cannot be combined with deep learning models. Some studies have attempted to label words as positive or negative Liu et al. (2004); Sindhwani and Melville (2008). Still, this approach is limited when it comes to social media posts, where words are often used with different meanings, typographical errors or in a cynical tone, making it difficult to label the text accurately.

4.3 Methodology

4.3.1 Seed Labeling

Manually labeling a small number of posts (seeds) for initial model training can be a time-consuming and expensive task, especially considering that data sharing is often restricted on social media platforms. However, research has shown that about 90% of data posted by users actively discussing a particular topic is relevant to that topic Llewellyn et al. (2016). To simplify the labeling process, one approach is to identify profiles of users and organizations that support each label position. Lists of these organizations can often be found online for significant political and social discussions, and social platforms may provide recommendations and search systems to identify similar profiles or relevant organizations. It's important to only extract profiles with a clear opinion on the topic of interest to ensure consistent labeling of all posts from that user. Profiles of objective organizations such as media outlets should be avoided, and once a user's opinion is known, their posts in the

dataset should be labeled accordingly.

4.3.2 Semi-supervised Label Prediction (SLP)

The present study proposes a general semi-supervised model that can incorporate different language modeling and classification algorithms, as opposed to existing semi-supervised methods which require large documents or additional information and work with a single model. The main objective of this proposed model is to assist in the analysis of information from social networks, hence, it is designed to work with small documents and algorithms that provide the best short-text classification results, including deep learning models. Given that opinion modeling usually involves only a few categories, the focus of this framework is on optimal performance with a limited number of classes.

The proposed framework's architecture, as shown in **Figure 4.1**, assumes a small labeled training data (a few thousand samples) and a large unlabeled data (millions of samples). Labels are assigned values $(1, 2, ...L)$ for an L-class problem, which can represent different meanings such as reviews scores or opinions on a particular topic. The language model and the classification system, built on top of it, are trained on the small labeled training data to obtain the initial model. This model is used to predict label probabilities for each post from the unlabeled data. Posts assigned the highest probabilities (at least X times larger than the next most probable label) are moved from the unlabeled to the training dataset.

The iterative process continues until the number of posts moved to the training data in an iteration is less than PL or the amount of unlabeled data is less than UL samples. If this occurs, any remaining unlabeled data is assigned the label with the highest probability from the last iteration, though this step is optional and its necessity depends on the purpose of the modeling.

The proposed method in this paper increases the size of the training dataset by including model-labeled data, which enables the addition of new contexts to the training dataset along

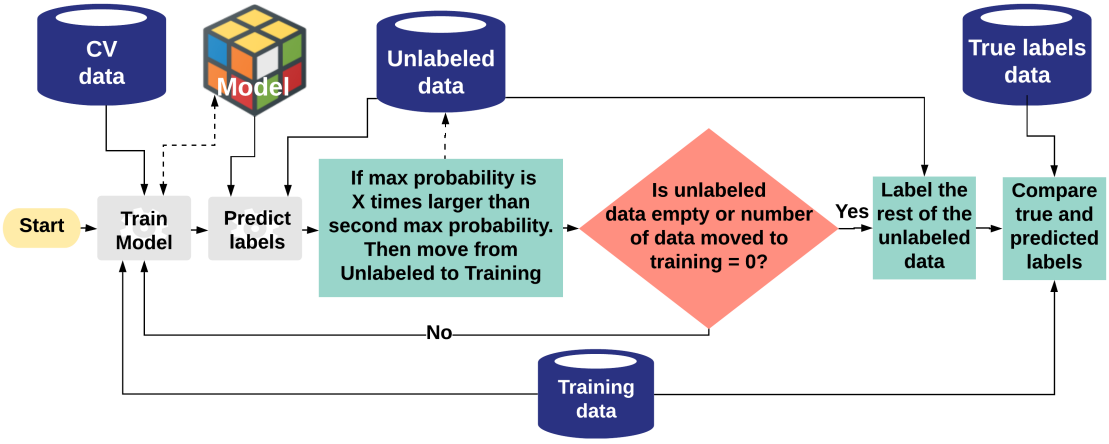


FIGURE 4.1: Architecture of the proposed semi-supervised label prediction (SLP) framework.

with their probable labels. While this approach is beneficial for models that learn from contexts, it also introduces noise into the training dataset as the labels are generated with some probability. The value of the ratio X determines the level of strictness the model should have when accepting new labels. Smaller values of X lead to more elements being added to the training dataset, thus speeding up the model's learning. However, larger values of X allow for more noise to be introduced.

Our study investigates the hypothesis that the SLP architecture is particularly advantageous for language models that rely on the order of words rather than frequency-based alternatives like bag-of-words. When working with short texts, deep neural networks have demonstrated much better results, as they learn from contexts and embed higher-level features and meanings of the language. The proposed model increases the training dataset iteratively with well-labeled contexts, providing more valuable information that can be used for training. It is anticipated that for models that learn from context, such as fastText and deep learning models, the addition of new examples will provide greater benefits than the noise created by their most probable labels.

4.3.3 Handling very small seeds

Different language models have varying performance on small and large datasets. Models like tfidf+ngrams tend to perform better on small datasets while models like vdcnn and fasttext are better suited for larger datasets. One advantage of using the proposed iterative process is that a different model can be trained in each iteration, allowing for the combination of distinct classification methods to achieve improved performance. For instance, starting with a small seed, one can use the tfidf+ngrams model in the first iteration to add more examples, followed by a deep learning model in subsequent iterations.

Alternatively, to work with a very small seed (i.e., a few hundred examples), one can use pre-trained word vectors from word2vec or Glove available online as the input of the first step. However, since these vectors are created from general text, such as Wikipedia or news articles, it is essential to add new examples of topic-specific text through the iterative process to achieve better results. The hypothesis is that by adding new topic-specific contexts, the model can adapt and learn new meanings of words and phrases in the particular domain, thereby improving classification accuracy.

4.3.4 Exit criteria

The criteria for stopping language modeling algorithms can be customized based on various factors, such as the nature of the text, the algorithms being used, and the desired speed of classification. For example, one can choose to terminate the algorithm after a certain number of iterations or when the proportion of labeled posts falls below a certain threshold relative to the size of the original dataset.

One possible rationale for stopping the algorithm at this point is that the unlabeled data may contain noisy or ambiguous posts that do not contribute much to the final model. For instance, such posts may lack a clear opinion or express multiple conflicting viewpoints that are difficult to annotate even for humans.

However, in some cases, the difficulty of labeling certain posts may warrant further

investigation or consultation with a domain expert. For instance, social media posts may contain multimedia content that provides additional clues, while comments or reviews may benefit from analysis of the surrounding context.

4.3.5 Streaming

The model we propose is particularly well-suited for handling streaming data, as it allows for the straightforward incorporation of new unlabeled and training examples. Moreover, it can identify which examples are challenging to classify (i.e., those that remain unclassified after several iterations). By focusing human labeling efforts on these difficult cases, significant improvements to the model can be achieved with only a modest investment of additional resources.

4.3.6 Model testing

There are two methods for evaluating the effectiveness of our model. The first involves comparing its performance to true labels, which are available for some of the unlabeled data. The second method entails testing on a small portion of the labeled data that was set aside prior to the training process. This type of evaluation is particularly relevant for real-world applications. However, if labeled datasets with short texts are available, the first option can also be employed to test new combinations of language and classification models.

For this study, we employed both evaluation methods. We tested our model on large labeled datasets from Zhang et al. (2015) as well as a newly created Twitter dataset.

4.4 Experimental Evaluation

Evaluating the performance of the proposed algorithm involves several variables, such as the language and classification algorithms used, dataset characteristics, and parameter values. To streamline the experimentation process while covering a range of test cases, we

chose components carefully. Since the proposed model is primarily intended for short text, its performance was tested exclusively in this domain, despite the existence of multiple semi-supervised approaches for longer texts.

4.4.1 Experimental Environment

To model and classify text using the deep learning model (VDCNN), we conducted experiments on four high-performance nodes with 512 GB of DDR4 2400MHz RAM. Each node featured two sockets with Intel Xeon E5-2667 v4 3.2GHz processors, as well as two NVIDIA Tesla P100 PCIe 12GB GPUs and SSDs as local hard drives. For other experiments, we used a 64-bit processor (Intel Core i7-6700 CPU @ 2.60 GHz) with four cores and 16.0 GB of RAM.

Table 4.1: Large-scale text classification datasets

Dataset	Unlabeled	Data Type	Classes
AG news	128k	Topics	4
Yelp Review Polarity	598k	Reviews	2
Amazon Review Full	3,650k	Sentiment	5
Amazon Review Polarity	4,000k	Sentiment	2
Twitter guns	11,750k	Opinions	2

4.4.2 Datasets

To assess various aspects of the proposed model, we selected four datasets from Zhang et al. (2015) with different characteristics. Since our model focuses on opinions, we chose datasets with a small number of classes. We excluded the Sogou News dataset since we lack Chinese language expertise.

Furthermore, we applied our method to a new and relevant problem in the domain of gun advocacy, which we extracted from Twitter. These benchmark datasets represent a range of classification tasks, including sentiment analysis, opinion modeling, product

reviews, and news categorization. All the published datasets are balanced, so we couldn't test the influence of class balance on the results. In reality, many problems are imbalanced, which is why we extracted the gun advocacy dataset from Twitter to test such a scenario.

The number of unlabeled examples varies significantly across the selected datasets. Additionally, to test the robustness of our model, we considered different numbers of classes. Some datasets had only two classes, which made them suitable for independent testing, due to differences in the size of unlabeled data and text purpose. The number of unlabeled examples used in our experiments is reported in Table 4.1. We extracted these unlabeled examples by combining the training and testing data from Zhang et al. (2015), and removing labels for all examples not used for training and cross-validation in our experiments.

4.4.3 Twitter guns advocacy dataset

This dataset comprises of two classes: gun rights and gun control advocacy. It was sourced from Twitter by starting the search with a list of gun rights/control organizations obtained from Wikipedia and using hashtags suggested by experts for data selection. Relevant tweets were selected while reducing noise, following a protocol previously developed by Buntain et al. (2018). The dataset contains 11.75 million unlabeled posts with information about the username of the post author.

Twitter's feature that suggests profiles similar to the current profile was also used to recursively explore profiles that align with the particular class, and if a profile was closely related to the particular class, all of its posts were labeled accordingly. Only profiles whose descriptions clearly stated that their goals align with one of the classes were considered. This resulted in the selection of 171 profiles, with 88 labeled as gun rights advocacy and 83 labeled as gun control advocacy groups.

The dataset had a total of 8,448 posts, with 7,782 posts belonging to the gun control class, indicating that the downloaded data is imbalanced.

The purpose of using this dataset was to replicate a scenario that a researcher or an

opinion research center may encounter. The goal was to evaluate the performance of the SLP model in handling imbalanced datasets, which is a common challenge in tasks such as modeling public opinion or filtering out irrelevant posts. This study aimed to address this issue and provide insights into its potential solutions.

4.4.4 Parameters settings

In this model, the two key parameters are the **ratio** and **seed size**. The proposed algorithm trains and predicts repeatedly until all posts have been labeled, or until there is no more unlabeled data left. However, the iterative labeling process can be stopped earlier by introducing different stopping criteria. Several options are possible, including $PL \in \mathbb{N}_0$ (the number of examples added to the training dataset in the current iteration), $UL \in \mathbb{N}_0$ (the number of examples left in the unlabeled dataset), $NI \in \mathbb{N}_0$ (the number of iterations that have passed), and $UP \in \mathbb{N}_0$ (the percentage of unlabeled data that remains). If the training component works with batches, the newly labeled data should contain at least a few batches, so PL and UL should be multipliers of the batch size. These variables mainly affect the runtime of the algorithm and can also be used to extract a certain percentage or amount of unlabeled data that are challenging to label. Although not required, these stopping criteria provide flexibility and enable researchers to adjust the performance of the model according to their needs.

The parameter **Ratio X** determines the threshold probability required for a predicted label to be considered reliable enough for the corresponding example to be moved from the unlabeled dataset to the training dataset. In each prediction step, a probability is assigned to each label, and the two highest values are compared for each post. If the ratio between these values exceeds X, the post is added to the training data for the next iteration. Ratios of 2, 3, and 4 were tested in the experiments, and additional tests were conducted with ratios of 1.2 and 1.5 after finding that the model performed best with a ratio of 2, and accuracy decreased with larger ratios.

The **seed size** is the number of labeled examples used for training and cross-validation (if required) in the model. The study examined two seed sizes in detail (6.7K and 67K), with 5K/50K examples used for training and 1.7K/17K for cross-validation. Other seed size options considered were 500 and 500K, but labeling 500K examples requires significant effort, and existing models already perform well with such a large dataset. To address the problem of having too little text in 500 samples of short-text, pretrained GloVe vectors from Mikolov et al. (2018) [1] were added to the first iteration classification. Thompson et al. Thompson et al. (2018) show that continuous training can provide additional benefits even with large datasets.

4.4.5 Models used for training step

SLP is a versatile framework that can accommodate various language models and classification algorithms. The language modeling options that can be used with SLP fall into three main categories: 1) traditional models based on word-document matrix; 2) deep learning models; and 3) models based on word2vec and GloVe. For the training phase of the proposed model, one algorithm was selected from each group: 1) tfidf of character n-grams with $n \in 1, 2$; 2) a very deep convolutional neural network (VDCNN) with 9 layers and k-max-pooling, as described in Conneau et al. (2016); and 3) fastText classification, as presented in Grave et al. (2017). Logistic regression was applied on top of tfidf+ngram to classify the documents, since this model only models the language. Furthermore, tfidf+ngram and fastText were combined due to their complementary properties. In the first iteration, tfidf+ngram was utilized because it performs well on small data. In all subsequent iterations, fastText was used since the training size became large enough to take advantage of this algorithm's benefits.

[1] Pretrained 300-dimensional vectors of 16B tokens of wikipedia text: https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip.

4.4.6 Baseline models

The proposed model underwent a comparative analysis with three other classification models: tfidf+ngram, VDCNN, and fastText.

4.4.7 Time performance

The time complexity of the proposed model is equivalent to the training model, with label prediction time being significantly shorter than the training time. The number of iterations is relatively constant, ranging from 2 to 30, depending on various factors such as dataset characteristics, training algorithms, and stopping criteria. When using deep learning models, the model need not be trained on the same examples multiple times, and newly labeled data can be used for additional training as if it were a new batch added. Consequently, in this case, the training time of SLP with a deep learning classification component is nearly identical to that of the classification algorithm alone.

Table 4.2: The label prediction accuracy (LP) was evaluated for a seed size of 5,000, and the best ratio for achieving the highest LP for each proposed model is indicated in parentheses.

Model	**AG**	**Amz. F.**	**Amz P.**	**Yelp P.**	**Twiter**
VDCNN	0.375	0.286	0.683	0.790	0.921
ngrams+tfidf	0.729	0.270	0.744	0.776	0.954
fastText	0.336	0.201	0.500	0.500	0.920
SLP(VDCNN)	0.645 (4)	0.285 (2)	0.758 (3)	0.521 (4)	**0.965 (4)**
SLP(ngrams+tfidf)	0.721 (1.2)	0.311 (1.2)	0.731 (1.2)	0.807 (1.2)	0.927 (1.2)
SLP(fastText)	**0.855 (2)**	**0.409 (2)**	**0.817 (2)**	**0.810 (2)**	0.923 (4)
SLP(fT+ngrams+tfidf)	0.797 (3)	0.354 (1.2)	0.771 (3)	0.744 (1.2)	0.951 (1.5)

4.5 Results and Discussion

Table 4.2 and 4.3 present the results for seed sizes of 5K and 50K, respectively, comparing three text classification models from the literature (VDCNN, ngrams+tfidf with

Table 4.3: The accuracy of label prediction (LP) is reported for a seed size of 50,000, with the best ratio for each value of the proposed model indicated in parentheses.

Model	AG	Amz. F.	Amz P.	Yelp P.
VDCNN	0.869	0.458	0.862	**0.908**
ngrams+tfidf	0.757	0.388	0.749	0.805
fastText	0.853	0.387	0.809	0.501
SLP(VDCNN)	**0.884(3)**	**0.482(2)**	**0.876 (4)**	**0.908(3)**
SLP(ngrams+tfidf)	0.750(4)	0.372(4)	0.731(1.2)	0.809(4)
SLP(fastText)	0.871(2)	0.431(2)	0.840 (2)	0.863(2)
SLP(fT+ngrams+tfidf)	0.861(4)	0.398(1.2)	0.816(4)	0.808(1.2)

logistic regression, and fastText) to the proposed model (SLP) trained in four different ways: VDCNN, ngrams+tfidf with logistic regression, fastText, and a combination of ngrams+tfidf+fastText. Accuracy was calculated by comparing the true and predicted labels of the large unlabeled datasets obtained from the literature. Variability in results between repeated experiments was insignificant due to the large number of examples. To evaluate the models, they were also tested using 1.7K/17K labeled samples from each dataset. However, the small number of test samples (especially in the 1.7K test dataset) resulted in more variable results, with accuracies ranging from 0.1% to 5% when the model was trained with a seed of 5K and tested with 1.7K documents, and from 0.1% to 1.5% when trained with 5K and tested with 17K examples. Therefore, the reported accuracy in Tables 4.2 and 4.3 is based on the percent of accurately labeled examples from the unlabeled dataset, as this accuracy is stable. For the twitter data, where correct labels of unlabeled data are not available, the accuracy was calculated as an average of testing results achieved by multiple experiments.

The proposed model's performance was evaluated for seed size 5K, and the best results were obtained using SLP trained with fastText for four datasets, while SLP trained with VDCNN was the best for the last two datasets. Notably, for the four datasets where SLP trained with fastText performed the best, the optimal ratio was found to be $X = 2$.

A comparison was also made between the original classification algorithm and SLP using the same algorithm for the training step. As expected, SLP with ngrams+tfidf was rarely better than the original ngrams+tfidf since it is a bag-of-words algorithm that can already capture word frequencies well with a small training dataset, and adding contexts through additional automatically labeled samples did not provide significant new information but instead introduced noise, causing SLP trained with ngrams+tfidf to perform worse in four datasets for both seed sizes. In contrast, SLP trained with VDCNN or fastText performed better than the original VDCNN and fastText for most datasets for both seed sizes, as these algorithms rely on contexts, and adding new training samples improved their performance despite the noise in the added labels.

For the smaller seed size (5K), the SLP model trained with ngrams+tfidf achieved the best results with ratio X = 1.2, while SLP trained with VDCNN performed better with higher ratios, and SLP trained with fastText most often obtained the best results with a ratio of 2.

Experiments were conducted on datasets with 10 (Yahoo Answers) and 14 (DBpedia) different classes. The performance of the framework on these datasets was only slightly better than random selection, likely due to the small number of labeled examples per class and the noisy nature of the short text. In particular, the DBpedia dataset contains entity categories, and the text is a formal description of the entity, with most words used only once to describe relevant information and others used frequently, which makes distinguishing classes challenging with limited labeled data.

Overall, the accuracy of classification was strongly dependent on the number of classes. For the Amazon Full dataset, the best accuracy achieved was 0.409 and 0.482 for 5K and 50K seed sizes, respectively, which was significantly lower than the accuracy achieved on the Amazon Polarity dataset using the same texts. Thus, an increasing number of classes from two to five had a profound effect on the accuracy of the algorithms.

Regarding the impact of the size of unlabeled data on accuracy, better performance was

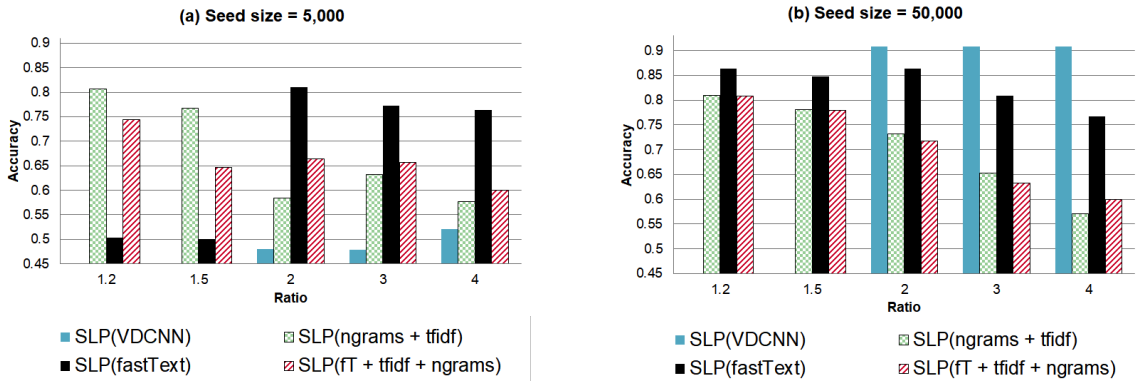


FIGURE 4.2: The label prediction accuracy of the SLP model was evaluated for the Yelp Review Polarity dataset using multiple language modeling algorithms and various ratios for two different seed sizes: (a) 5,000 and (b) 50,000.

observed when the amount of unlabeled data was smaller, meaning that a higher percentage of the dataset was labeled. Figure 4.2 depicts the influence of the ratio X on SLP accuracy for the Yelp Review Polarity dataset. A higher ratio value had a negative impact on SLP trained with tfidf+ngrams and tfidf+ngrams+fastText, whereas fastText achieved its maximum accuracy at ratio 2 for all examples before gradually declining. In most cases, the seed size had a positive effect on accuracy when the other parameters were kept constant.

We also tested the hypothesis that our model could be applied to very small datasets (with just a few hundred examples) by adding pretrained vector embeddings as input, using the fastText algorithm, which is known to struggle with such small datasets. Results showed that adding pretrained vector embeddings to fastText classification improved the accuracy by 8-27%. The accuracy was further enhanced when SLP was trained with fastText initialized with pretrained vector embeddings. The influence of initialization was most significant on the AG news dataset, possibly because of the high similarity between news texts and Wikipedia data, compared to other datasets. The effectiveness of SLP was more dependent on the text type than the seed size.

To validate the hypothesis that the unlabeled examples in SLP iterations were ambiguous or contained weak signals, a closer examination of these examples was conducted. For

instance, in the Amazon Full dataset, most of the reviews that remained unlabeled exhibited conflicting opinions (such as those who believed the product was cheap and functional but not of good quality, or those who initially loved the product but had a change of heart due to a specific event). Examples of such reviews are presented below:

"The tacos shells were mainly broken. shells are too thin for transportation. the shells are very good. mainly use for dipping."

"grass grows quickly and its cool at first, but with no sun, eventually molds, roots invade the bowl,and my cats didn't catch on to it.... pfft"

Eliminating the last approximately 5,000 examples that were the most difficult to label resulted in a 3-5% improvement in accuracy for the SLP model.

Although the Twitter dataset had high accuracy for all models, it was not the best performance indicator because of its imbalanced nature, so the F1 score was also measured. SLP trained with a combination of ngrams+tfidf+fastText achieved the best F1 score of 0.675, while the worst score was obtained by the fastText classification. While the original VDCNN and tfidf algorithms can adjust to imbalanced classes through their settings, fastText does not perform well in such cases. Therefore, the SLP model that uses tfidf and VDCNN achieved a good F1 score, while the SLP model with fastText had a low F1 value.

4.6 Summary

In this chapter, a semi-automatic labeling approach is proposed for surveying public opinion from large domain-specific textual data in social media while minimizing the challenges associated with modeling public behavior. The approach is based on a two-part framework that uses a small subset of posts, referred to as the "seed," for semi-supervised labeling of the rest of the data. The proposed approach was evaluated on four datasets and compared to three popular text modeling algorithms, and the results showed that the proposed semi-supervised method outperformed alternative algorithms by capturing additional contexts from the unlabeled data. The proposed model was also applied to label Twitter posts re-

lated to gun control/rights advocacy and achieved better performance than existing models with 96.5% accuracy and 0.68 F1 score.

The obtained results provide evidence that the proposed model outperforms state-of-the-art classification models in labeling vast amounts of unlabeled data based on the small seed, except when the number of categories is substantial. The study highlights the importance of the choice of training component based on the dataset characteristics. Overall, the proposed semi-automatic labeling approach is an efficient and effective method for surveying public opinion from large textual data in social media, and it can be applied to various domains and topics with minor modifications.

CHAPTER 5

EXPLORING THE IMPACT OF UNFORESEEN BIAS ON DOMAIN ADAPTATION

In recent years, domain adaptation has emerged as a popular technique in machine learning to tackle the problem of limited labeled data. However, a significant challenge in domain adaptation is the presence of unforeseen bias, which can lead to a significant drop in performance on the target domain. Unforeseen bias refers to the situations where the source and target domains have similar statistical properties, but the target domain contains specific variations or biases that are not present in the source domain.

One of the main challenges in dealing with unforeseen bias is that it is difficult to detect and measure. Bias can arise from a variety of factors such as demographic factors, socioeconomic factors, cultural factors, and even technical factors such as the sensor used to capture data. A biased dataset can lead to a model that generalizes poorly to the target domain, leading to unfair and discriminatory outcomes.

In this chapter, we test domain adaptation outcomes when the model pre-trained on English language Wikipedia data is fine-tuned into the domain of politics using news from different biased political sources (news outlets). We measure the results drop given a fine-tuning dataset of different bias strengths. We also examine the performance behavior on

balanced and imbalanced labeled datasets.

5.1 Introduction

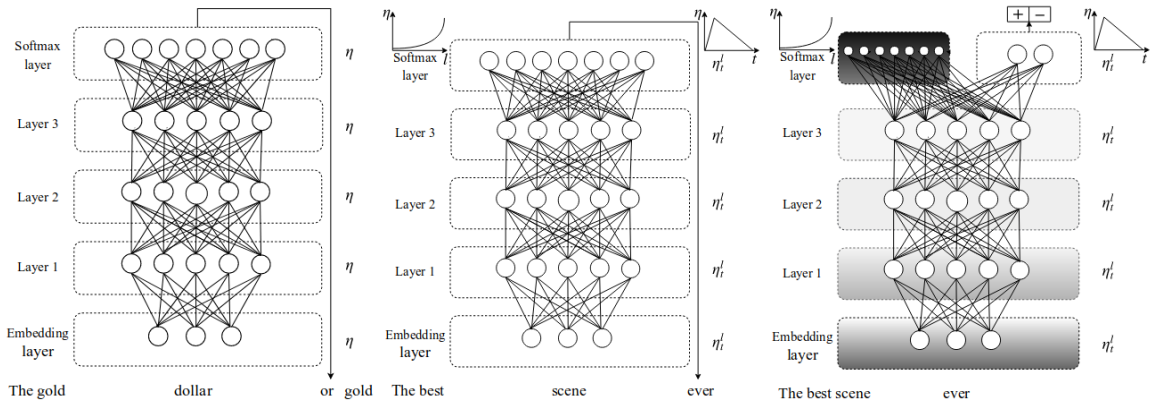
In the past few years, social media platforms have emerged as major channels for knowledge exchange, debates, and product or opinion advertising Pak and Paroubek (2010); Watts and Dodds (2007); Glynn (2018); Sobkowicz et al. (2012). Social scientists use data from social media platforms to examine public opinion on specific topics Mostafa (2013); Celli et al. (2016); Howard et al. (2017); Bastos and Mercea (2018). In contrast, computer scientists employ the data to enhance the performance of natural language processing (NLP) algorithms Chen et al. (2017); Appel et al. (2016); García-Pablos et al. (2018); Zhang et al. (2018b).

Despite the abundance of social media data, using it poses several challenges. Firstly, user demographics are rarely available. Secondly, posts are often short and difficult to understand without context. Lastly, manually labeling millions of posts is time-consuming and challenging. To address the latter two challenges, we require systems that can classify data into different opinion classes with limited human involvement. Although we may address the first challenge by selecting information only from users where demographic information is available across multiple social platforms, doing so may bias the data.

FIGURE 5.1: ULMFiT model training-flow overview

There have been numerous algorithms proposed to deal with large amounts of short text, such as those discussed in Zhang et al. (2015); Zhou et al. (2016); Liu et al. (2016); Xiao and Cho (2016); Conneau et al. (2016); Yang et al. (2016); Mikolov et al. (2018). However, these algorithms are supervised and require a significant number of labels to perform well.

In recent years, several methods have been introduced to reduce the need for labeled data and improve performance by using self-supervised learning to fine-tune word vectors pre-trained on general documents for use on domain-specific documents. These include CoVe McCann et al. (2017), ELMo Peters et al. (2018), ULMFiT Howard and Ruder (2018), and OpenAI GPT Radford et al. (2018). In ULMFiT, for example, the self-supervised process predicts the next word based on the previous words in the context. The model is then fine-tuned on a small number of manually labeled test instances (as shown in Figure 5.1).



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

FIGURE 5.2: ULMFiT model training details

Most of the datasets used for testing text classification algorithms, such as AGNews, DBPedia, and Yahoo Answers, are typically balanced and contain longer texts than social media posts Zhang et al. (2015). However, social media data collected for modeling opinions is often unbalanced. This study aims to investigate the performance of the ULMFiT model in classifying social media posts using different fine-tuning settings and labeled datasets, including balanced and imbalanced social media datasets, and news texts with various characteristics such as size, bias, and writing style.

For our experiments, we used Twitter data related to the 2018 USA midterm elections and news data from the 2016 USA elections, collected from six major outlets known to have a bias towards the left or right political spectrum[1]. We tested how fine-tuning with articles

[1] Outlet bias information sourced from https://mediabiasfactcheck.com

from different news outlets affects the accuracy of social media post classification. We hypothesize that fine-tuning with relevant news articles can improve classification accuracy, but the bias in the news articles can also negatively affect performance. We tested our hypothesis using the ULMFiT algorithm, described in the next section[2].

5.2 Related Work

Text classification is a widely studied topic in natural language processing. Numerous algorithms have been proposed to cope with large amounts of short text, such as character-level models, recurrent neural networks, and hierarchical models Zhang et al. (2015); Zhou et al. (2016); Liu et al. (2016); Xiao and Cho (2016); Conneau et al. (2016); Yang et al. (2016); Mikolov et al. (2018). However, most datasets used for testing text classification algorithms are balanced and, on average, contain much longer texts than social media posts, which are usually unbalanced and shorter in length.

To address this challenge, recent studies have proposed self-supervised learning techniques such as CoVe, ELMo, ULMFiT, and OpenAI GPT to minimize the need for labeled data and increase performance by leveraging text characteristics McCann et al. (2017); Peters et al. (2018); Howard and Ruder (2018); Radford et al. (2018). These methods start from word-vectors pre-trained on general documents and fine-tune them on domain-specific documents by employing self-supervised learning. For example, in the Universal Language Model Fine-tuning for Text Classification (ULMFiT) Howard and Ruder (2018), the self-supervised process predicts the next word based on the previous words in the context.

Several studies have investigated the impact of biased news data on text classification. For example, Atanasova et al. (2019) proposed a method to combine news and social media data to improve the fact-checking of statements.

In this paper, we aim to investigate how fine-tuning news data from different outlets

[2] For the latest progress in text classification, see http://nlpprogress.com/english/text_classification.html

with varying political biases affects the accuracy of classifying social media posts related to the US midterm elections from 2018. We hypothesize that fine-tuning with appropriate topic-related text from news can improve classification performance, but bias in news articles can also hurt the performance. We test our hypothesis on the ULMFiT algorithm.

5.3 Methodology

The ULMFiT model Howard and Ruder (2018) comprises three training components,(Figure 5.2) each of which is based on the AWD-LSTM language model Merity et al. (2017). Each component consists of a word-embedding layer for input, multiple LSTM-layers, and a softmax layer for output prediction. Prior experimental results in the literature have shown that models with multiple LSTM-layers can learn more complex contexts McCann et al. (2017); Peters et al. (2018); Howard and Ruder (2018) than models with only a single LSTM-layer.

In the first part of ULMFiT, the model learns word and context embeddings from general texts, such as Wikipedia. In the second part, these embeddings are fine-tuned with domain-specific data to learn words and phrases related to a specific topic. In the third part, the model is trained on labeled examples from the same domain to predict labels for new examples. The output of each part serves as input for the next step.

Although the ULMFiT model is complex, it can be trained efficiently on GPUs with appropriate implementation. Once trained, the first part of the model remains unchanged, and we can use WT103 pre-trained vectors [3] to reduce training time.

To speed up fine-tuning (as depicted in Figure 5.2b), we adopt a strategy of using different learning rates for each LSTM layer. The layer that calculates the softmax has the largest learning rate, denoted as η^L, while the learning rates for the remaining layers are set to $n^{l-1} = n^l/2.6$ for $l \in (1, L)$, following the suggestion of a previous study Howard and Ruder (2018). Rather than using a constant learning rate, we apply slanted triangular

[3] WT103 word-vectors are available at http://files.fast.ai/models/wt103

learning rates for every layer, as recommended in Howard and Ruder (2018), to improve the model's accuracy. Specifically, the learning rate sharply and linearly increases initially, so the model can quickly learn from the first examples. Once the learning rate reaches η^L, it linearly and slowly declines, as shown in the top-right corner of Figure 5.2.

In the third step (Figure 5.2c), we gradually train the layers. Initially, only the top layer is trained with labeled data for one epoch, while the other layers are frozen. In each new epoch, we add the next frozen layer from the top to the training.

5.4 Experimental Evaluation

The study uses two sets of data: Twitter data related to USA midterm elections in 2018 and news data from the USA elections in 2016.

For the **Twitter data**, the researchers collected posts published between November 4th and 7th, 2018, that contained at least one of seven hashtags related to the election. In total, 936,462 tweets were collected, and after removing retweets, 244,320 unique posts remained. The text of each post was pre-processed by removing all non-alphanumeric characters. The researchers manually labeled 1,526 of the posts with a score of 0, 1, or 2. A score of 0 was assigned to posts that expressed support for the left political spectrum or criticized the right, a score of 1 was given to politically neutral posts (such as those encouraging voting), and a score of 2 was assigned to posts that supported the right or criticized the left. About 25% of the collected posts (500 examples) were discarded because they were not related to the election.

News data is collected from six media outlets that are rated to have varying degrees of political partisanship, ranging from left-oriented to right-oriented sources, as determined by the Media Bias/Fact Check website (Table 5.1). We select articles published between October 2015 and May 2017 that contain any of the following keywords: "election", "ballot", "republican", "GOP", or "democrat". The news articles are highly diverse in terms of content, writing style, number of articles, number of words, and bias, as shown in Table

Table 5.1: Outlets

Outlet	Bias	#Words
CNN News (CNN)	left	426,778
Washington Post (WP)	left-center	9,229,176
BBC News (BBC)	neutral-left	1,247,437
MarketWatch (MW)	neutral-right	1,505,107
Wall Street Journal (WSJ)	right-center	547,548
FoxNews (FN)	right	3,082,912

5.1. However, like the tweets, the news articles do not always pertain to the U.S. elections, and may instead focus on topics such as Brexit or foreign elections. To preprocess the news data, we remove all non-alphanumeric characters from the articles.

5.4.1 Experimental settings

In the first step of our approach, we use pre-trained token-vectors from WT103. This pre-trained model has been trained on a large dataset of Wikipedia texts and consists of 103 million tokens for training, 217K tokens for validation, and 245K tokens for testing Merity et al. (2016). Our system is trained using the architecture shown in Figure 5.2a, and the vocabulary used has 267K unique tokens. For clarity, we use the terms ”word” and ”token” interchangeably in this paper.

Next, for the fine-tuning step, we explore ten different settings. The first setting includes all news text from different outlets, along with the tweets text. The second setting includes only the tweets text. The third and fourth settings include text from left-biased and right-biased outlets, respectively, along with the tweets text. The remaining six settings involve text from a single outlet and the tweets text. Before usage, we randomly permute the examples in the fine-tuning dataset.

The third step of the experiments involves testing two labeled Twitter data settings, namely Mix 1 (balanced mix) and Mix 2 (unbalanced mix). Mix 1 includes 380 examples labeled as 0 (left), 323 examples as 1 (neutral), and 323 examples as 2 (right). Mix 2 in-

Table 5.2: Classification results

News sources included (Left : Neutral : Right)	**Mix 1 (380 : 323 : 323)**	**Mix 2 (380 : 823 : 323)**
All news	$53.2 \pm 3\%$	$59.4 \pm 3.7\%$
No news	$56 \pm 5.3\%$	$\mathbf{66.6 \pm 2.5\%}$
Left-biased (CNN+WP+BBC)	$49.2 \pm 2.9\%$	$61.1 \pm 3.3\%$
Right-biased (MW+WSJ+FN)	$51.7 \pm 3.8\%$	$63.0 \pm 3.2\%$
CNN	$\mathbf{58.7 \pm 1.2\%}$	$62.7 \pm 3.0\%$
Washington Post (WP)	$55.6 \pm 3.0\%$	$60.7 \pm 1.4\%$
BBC	$55.1 \pm 3.1\%$	$64.1 \pm 2.7\%$
MarketWatch (MW)	$56.5 \pm 2.6\%$	$64.2 \pm 1.8\%$
Wall Street Journal (WSJ)	$57.7 \pm 3.7\%$	$60.0 \pm 4.3\%$
FoxNews (FN)	$53.2 \pm 2.9\%$	$61.9 \pm 3.3\%$

cludes 380 examples labeled as 0 (left), 823 examples as 1 (neutral), and 323 examples as 2 (right). The labeled data is randomly divided into three separate parts for testing, validation, and training. The training sets contain 626 examples in Mix 1 and 1126 examples in Mix 2. Each of the ten settings is repeated four times, and the accuracy mean and standard deviation are reported.

To emulate real-world situations, we intentionally refrain from cleaning Twitter and news data of irrelevant examples. This approach mirrors the information extraction process frequently employed in research papers Mostafa (2013); Celli et al. (2016); Howard et al. (2017); Bastos and Mercea (2018). The goal of these experiments is to evaluate the model's robustness to bias and noise in the data and the unbalanced classes.

5.5 Results and discussion

The experiment is repeated four times, and the accuracy mean and standard deviation are reported in Table 5.2. A high standard deviation $(1.2 - 5.3\%)$ indicates that the model is sensitive to the order of examples in the fine-tuning data and requires more labeled examples.

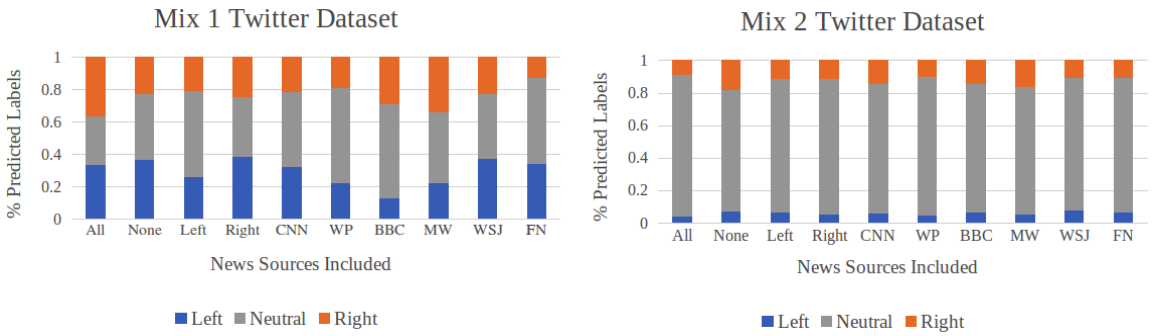


FIGURE 5.3: Percentage of predicted labels for each class when the model is fine-tuned with ten different combinations of news outlet texts. The chart on the left (a) corresponds to the balanced dataset, while the chart on the right (b) corresponds to the unbalanced dataset.

The results suggest that the model is not resilient to unbalanced datasets. When comparing the results for Mix 1 and Mix 2, the model consistently achieved better results for Mix 2 (Table 5.2), which has 54% of neutral labels compared to Mix 1's 31.5% of neutral labels. As shown in Figure 5.3b, $80 - 90\%$ of the predicted labels for Mix 2 are neutral. Consequently, the model performs better for Mix 2 because it overestimates the most prevalent (neutral) label in the imbalanced dataset, which contains 54% of examples of that class.

The fine-tuning experiments show that the largest difference in classification accuracy (11.9%) occurs when "left-biased news" is used for fine-tuning. In this scenario, both Mix 1 and Mix 2 show decreased accuracy compared to when "No news" is used, but the outlet bias has a greater impact on the accuracy of Mix 1. Although using "all news" data for fine-tuning achieves a better balance among predicted labels for Mix 1, almost half of the predicted labels are incorrect, resulting in low accuracy. This is illustrated in Figure 5.3.

The labeled Twitter data exhibit variation in the posts labeled as "left". Such posts often revolve around a specific topic and contain fewer hashtags that support left-leaning views. Moreover, compared to those labeled as "right", the "left" posts display a greater diversity of people and entities mentioned, with the focus being less on President Trump. Thus, the best performance for Mix 1 is observed when the model is fine-tuned with data

from ”CNN” since it helps the model to prioritize left-leaning contexts.

The next best results for Mix 1 are obtained when fine-tuning with news articles from The Wall Street Journal. This is because such articles typically provide a detailed discussion of both sides, often in the same sentence, which enables the model to understand relevant phrases and predict ”left” and ”right” labels more accurately. However, this experiment predicts the ”right” label more frequently for ”left-labeled” examples compared to other experiments.

The accuracy results and confusion matrices for each experiment reveal that the algorithm has an easier time recognizing the right label in Mix 2 compared to the left label. This may be due to the different writing style of left-labeled tweets, which cover a more diverse range of topics and entities, as discussed earlier. Interestingly, the best accuracy score for Mix 2 is achieved when ”no news” data is used for fine-tuning, possibly because news data tends to be more left or right-oriented and biased, which negatively impacts the accuracy.

As predicted, the results demonstrate that fine-tuning with biased news datasets can affect accuracy in varying ways. The impact of the bias is particularly evident in Mix 1, where the difference between the best and worst accuracy for different fine-tuning settings is 9.5%, and in Mix 2, where the difference is 7.2%. However, the influence of the bias is not uniform. For example, fine-tuning with ”left-biased news” yields the worst result for Mix 1 but performs averagely for Mix 2 compared to other experiments. In contrast, fine-tuning with ”all news” yields the worst results for Mix 2 but average results for Mix 1.

The results suggest that the size of the fine-tuning data has no significant impact on the model’s performance. Despite having the largest amount of words, ”Washington Post” achieves average results in both Mix 1 and Mix 2, while ”CNN,” the smallest dataset, achieves the best result for Mix 1. It’s noteworthy that fine-tuning with ”all news” results in lower accuracy than fine-tuning with ”no news” for both Mix 1 and Mix 2. This finding indicates that the content or bias of the fine-tuning dataset plays a more important role in

the model's performance than its size, which contradicts the belief that training with more data leads to better results.

Further analysis is required to better comprehend the behavior of accuracy in many experiments and to investigate the effect of non-relevant text on the model's performance. As the results indicate that the model is sensitive to bias and noise, it is crucial to explore other novel methods to address this issue. It's also essential to create unbalanced and biased datasets for fine-tuning and testing future models, enabling the development of robust methods that are useful for real-world applications.

5.6 Summary

This chapter highlights the importance of accounting for bias, noise, and text properties when constructing data for fine-tuning language models. The results of the experiments using ULMFiT language model on Twitter and news data show that the algorithm is not robust to these factors, and the characteristics of the fine-tuning dataset, such as bias, vocabulary diversity, and writing style, have a significant impact on the final classification results. The size of the data, however, is less consequential. The study also reveals that the algorithm is not robust on unbalanced Twitter datasets and tends to exaggerate when predicting the most frequent label. These findings suggest that social scientists using short-text classification models for opinion mining should carefully consider the properties of their training and fine-tuning datasets to improve the accuracy and robustness of their models. Future research could explore novel methods for constructing unbalanced and biased datasets that would be beneficial for real-world applications.

CHAPTER 6

IMPROVING DOMAIN ADAPTATION PERFORMANCE ON DATASETS WITH HIGH MISSING DATA RATES

Domain adaptation seeks to improve the generalizability of models trained on a source domain to an unseen target domain. However, many real-world datasets suffer from high missing data rates, making it difficult to obtain accurate and robust models and achieve good domain transfer.

In this chapter, we propose different iterative imputation methods and demonstrate that iterative imputation with a bag of regularized linear models achieves the best performance. This is in line with the findings of the multiple imputation approach, which involves generating multiple plausible imputed datasets from the original data by randomly filling in the missing values.

We also propose using data from multiple studies as a source dataset and show that the proposed imputation methods lead to better results on the source and target datasets. We demonstrate the effectiveness of this approach using clinical anxiety data, where the source datasets represent patients who completed the treatment program, and the target dataset consists of patients who dropped out of the program. The study aims to predict the treatment outcome for both groups using one or multiple data sources.

6.1 Introduction

According to a recent longitudinal study Costello et al. (2003), almost 37% of youth aged 9-13 (N=1420) will meet criteria for at least one psychiatric disorder diagnosis by the age of 16. Among the range of youth psychiatric disorders, anxiety disorders are the most prevalent. For instance, The National Comorbidity Survey of US Adolescents Merikangas et al. (2010) (N = 10,123) found that anxiety disorders are the most common disorders observed among adolescents aged 13-18, with an earlier age of onset (6 years) than behavioral (11 years), mood (13 years), and substance use (15 years) disorders. Additional evidence was presented in a separate study conducted by the World Health Organization WHO (2021), which estimated that 3.6% of children aged 10-14 and 4.6% of adolescents aged 15-19 experience an anxiety disorder. A recent study Bitsko et al. (2018) discovered that lifetime diagnoses of anxiety or depression among children aged 6-17 increased from 5.4% in 2003 to 8.4% in 2011-12. Besides being frequent, youth anxiety disorders can lead to substantial short- and long-term functional impairments across multiple domains if left untreated Swan and Kendall (2016), Essau et al. (2014).

In order to make evidence-based treatment allocation decisions, it is important to utilize efficacious interventions, and machine learning can be a valuable tool for this purpose. However, before implementing personalized treatments using machine learning, it is crucial to address data quality issues. Youth anxiety treatment studies pose multiple challenges for clinical psychologists, and off-the-shelf statistical tools are often ineffective in analyzing such data. To date, only one youth anxiety treatment study Lebowitz et al. (2021) has used a machine learning model to predict treatment outcomes within a small sample (N=124), and there has been no analysis of the methods used for this application. Thus, this paper aims to explore how machine learning can be effectively utilized to understand youth anxiety data and aid in predicting prognosis and assigning treatments.

In analyzing multi-informant, randomized clinical trial data, several challenges arise

that must be addressed when attempting to combine data across trials to create harmonized datasets, especially considering the longitudinal nature of the treatment. Firstly, the number of features collected in a single study greatly exceeds the number of patients, even after clinical feature selection. For instance, the largest anxiety treatment study to date Walkup et al. (2008) included 488 patients, but over 1000 features were collected from each study. Secondly, a significant number of patients withdraw from treatment or do not provide all the necessary information during the follow-up period, resulting in a large amount of missing data. Thirdly, the protocol can differ slightly across studies depending on the research objectives. Fourthly, youth often have comorbid diagnoses, leading to multiple predictive variables. Finally, data can be influenced by recorded diagnoses, the physicians conducting the studies, and geographical location. For example, while ethnicity is commonly documented in the US, it may not be recorded in many other parts of the world.

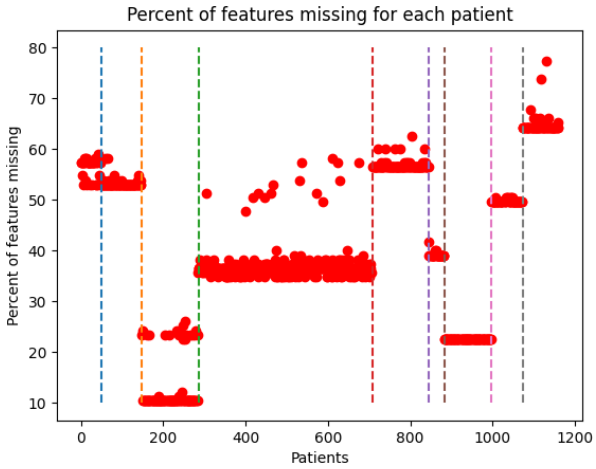


FIGURE 6.1: This figure displays the percentage of missing features per patient, where a total of 108 features were considered. The patients are separated by vertical lines to indicate the different studies they belong to.

To tackle these challenges, we have utilized data from nine different youth anxiety studies Kendall (1994), Kendall et al. (1997), Kendall et al. (2008), Walkup et al. (2008), Villabø et al. (2018), Wood et al. (2006), Bodden et al. (2008), Nauta et al. (2003), and Silverman et al. (2009). Figure 6.1 displays the missing rate for each patient, which is

clearly influenced by the study they are a part of, ranging from 10% to 80%. None of the patients have fully observed data, indicating the severity of the issue.

Our proposed methods and results can be applied to other longitudinal studies with high missing rates, such as clinical trials and observational studies, and may also be relevant for future personalized medical diagnostic and treatment approaches utilizing self-observational, device-based, and other longitudinal data dependent on patient promptness.

Our contributions to this research are as follows:

1. We have proposed and evaluated more complex machine learning-based imputation methods than those typically utilized in clinical psychology intervention studies.

2. We have evaluated whether data from multiple studies with different biases and missing rates can improve imputation and enhance data understanding.

3. We have examined the tool's performance on domain transfer, specifically targeting patients who did not complete the treatment.

6.2 Related work

6.2.1 Imputation state of the art

According to Dong and Peng (2013), missing data rates of 15-20% are common in psychological studies. A review of articles published up to 2006 found that 48% of studies had missing data, and 97% of these studies used either listwise or pairwise deletion, leading to biased estimates. In a more recent analysis of articles from 2009 and 2010, 67.6% of publications had missing data, with 37% not addressing missing data and 28.3% using deletion methods. The remaining studies used more sophisticated techniques such as full information maximum likelihood and expectation maximization, with the latter performing the best.

A recent review of longitudinal studies in older adults by Okpara et al. (2022) found even more concerning results. Among the published work, 62.5% did not have a clear de-

scription of how missing data were handled, with missingness rates varying from 0.1% to 55%. Among the studies that did address missing data, 75% simply removed samples with missing data, with only 6% using iterative imputation methods. A review of 100 clinical trial articles published from 2005 to 2012 by Powney et al. (2014) reported similar findings, with only 9% of studies having no missing data and more than half not providing an explanation of how they handled missing data. Only 20% of the studies used any imputation method, with multiple imputation being used in only 4% of the studies.

The authors of Petrazzini et al. (2021) assessed commonly used R imputation packages, including missForest[1], nearest neighbors (DMwR[2]), multiple imputation by chained equations (MICE[3]), multivariate normal distribution using EMB (Amelia[4]), and Bayesian-based approximation (mi[5]), on genomic data. The genomic data has similar properties to longitudinal clinical psychology data, such as a large number of features compared to the number of patients, requiring domain knowledge to reduce the number of features, a high rate of missing values, and datasets that are typically underpowered and must be merged to generate adequate sample sizes. The authors found that the nearest neighbors and missForest approaches worked best, with missForest performing better on datasets with lower missing rates and nearest neighbors on datasets with higher missing rates. However, we argue that our proposed imputation methods will perform better on the anxiety dataset and other datasets with numerical features and predicted values.

The authors in Schmitt et al. (2015) conduct a comparison of various off-the-shelf methods for imputation, including mean, KNN, fuzzy K-means (FKM[6]), singular value decom-

[1] missForest:`https://cran.r-project.org/web/packages/missForest`

[2] DMwR:`http://www2.uaem.mx/r-mirror/web/packages/DMwR/index.html`

[3] MICE: `https://cran.r-project.org/web/packages/mice/mice.pdf`

[4] Amelia:`https://cran.r-project.org/web/packages/Amelia/index.html`

[5] mi: `https://cran.r-project.org/web/packages/mi/mi.pdf`

[6] fclust:`https://cran.r-project.org/web/packages/fclust/index.html`

position (SVD), Bayesian principal component analysis (bPCA[7]), and MICE. Their results indicate that FKM is the top performer across multiple datasets. However, FKM and bPCA require at least some samples to have fully observed data, which is not the case for many clinical studies, including our own.

A survey in Emmanuel et al. (2021) outlines statistical and machine learning techniques for handling missing data, including LD, PD, and imputation using methods such as median, regression, hot-deck, expectation-maximization, multiple imputations (MI), k-nearest neighbors (KNN), support vector machine (SVM), decision trees, clustering, and ensembles. However, the study only tests KNN and decision tree (DT) approaches on data with missing rates up to 15%. Given the longitudinal nature of anxiety and mental health datasets, the missing rate is often much higher, and it is necessary to evaluate how these imputation methods perform under high-missingness conditions.

Several longitudinal studies, including Mainzer et al. (2021), Bell and Fairclough (2014), DeSouza et al. (2009), and Molenberghs et al. (2004), face the problem of high missing rates. These studies reviewed best practices for imputation and cautioned about the difficulty of understanding the type of missingness present in the data.

In Fouad et al. (2021), the authors proposed a new approach called KI, which integrates iterative imputation (II) and KNN, and an even more effective FCKI approach that splits the dataset using fuzzy c-means and applies KI to each cluster. As a result, FCKI and KI demonstrate an average imputation RMSE that is ten times better than that of simple statistical methods, such as mean, kNNI, SoftImpute, and SVDImpute. However, both KI and FCKI require that at least some patients are fully observed.

6.2.2 Machine learning in psychological studies

Many psychology studies still use LD and PD to handle missing values, but these approaches are not appropriate for longitudinal studies due to their higher percentage of

[7] bPCA:`https://cran.r-project.org/web/packages/bpca/index.html`

missing data. More recent studies Koesmahargyo et al. (2020) have used R packages that support missing data, where predicted variables are fully observed, and explanatory variables have some fully observed features and patients. For example, the study Gradus et al. (2021) investigated suicide attempts of n = 22,974 people with scarcely missing explanatory features, using independent fully observed surrogate variables with rpart[8]. The authors of Gradus et al. (2017) analyzed only participants with fully observed predicted variables (n=2,244), and applied a tree algorithm with surrogate variables using ctree[9]. Similarly, the analysis of cognitive processing of n=94 people Augsburger and Galatzer-Levy (2020) with only 10% of missing data used a random forest recursive partitioning approach to impute those values. In Schultebraucks et al. (2021), missing values for n=417 were imputed using bagged imputation.

The only study on youth anxiety that employed machine learning was conducted by Lebowitz et al. Lebowitz et al. (2021). They used Markov Chain Monte Carlo (MCMC) for imputation and random forest for prediction. According to Lin et al. Lin (2010), MCMC performs similarly to EM imputation. However, none of the studies compared the influence of imputation on prediction tasks, and they all worked with datasets with low missing rates, some even requiring fully observed patients or features.

Since longitudinal youth anxiety data has a small sample size and high missingness by design, imputation is often required. The best machine learning-based imputation techniques, as suggested by the aforementioned references, are: 1) KI and FCKI - iterative learning integrated with fuzzy clustering and KNN, 2) MICE - iterative chained equations, and 3) missForest - iterative learning over trees. However, KI, FCKI, and MICE require at least some samples to be fully observed. Additionally, our results demonstrate that missForest may not be the best solution when most of the features and all outcomes are numerical.

[8] rpart:`https://cran.r-project.org/web/packages/rpart/`

[9] ctree: https://cran.r-project.org/web/packages/partykit/vignettes/ctree.pdf

Algorithm 1 Data cleaning

Procedure Cleaning (D) /* D - domain-expert dataset */
$D_{null} \leftarrow$ Set_Missing_And_Not_Recorded_Values_To_Null(D)
$D_{features} \leftarrow$ One_Hot_Encode(D_{null}, race, study, treatment)
$D_{finished}$, $D_{drop} \leftarrow$ Split_Patients($D_{features}$)
$D_i \leftarrow$ Split_Studies($D_{finished}$) /* $i \in (1..9)$ */

for D$_{clean} \in (D_{finished}, D_{drop}, D_i)$ $/ * i \in (1..9) * /$
doX$\leftarrow$ Features_Of(D_{clean})
$y \leftarrow$ Labels_Of(D_{clean})
$X_{normalized} \leftarrow$ Normalize(X)
yield $X_{normalized}$, y

6.3 Methods

The Methodology section outlines both the machine learning techniques that were proposed and utilized, while the Experiments section provides further information on the experimental parameters and the data used.

6.3.1 Imputation methods

Many modern imputation methods, such as MICE, KI, FCKI, and missForest, use iterative processes to improve imputation accuracy. We propose integrating machine learning models that are good at modeling small linear data, such as Ridge, Elastic Net, or Lasso regression, with iterative imputation. We believe that these techniques can take advantage of correlations among features that decision tree and random forest-based models (such as missForest) may not capture. Moreover, we expect that linear models will learn better representations given that the data is mainly numerical. Unlike MICE, KI, and FCKI, our proposed model will not require fully observed samples.

We begin by using initial mean imputation to impute unknown values and then propose a two-level iterative imputation approach for regularized regression models, which require fully imputed data. In our application, only a few features (number of sessions, age, sex, study, and starting condition) are fully observed, while some relevant features have a missing rate of 80%. Therefore, the inner iteration step always handles the feature with the

Algorithm 2 Training Imputation Model

Procedure ImputationTraining ($data$, $model$)

for $feature$ $\in$ Features($data$) **do** $feature_{imp}$ $\leftarrow$ Mean_Impute($feature$) $data_{imp}$ $\leftarrow$ Concatenate($feature_{imp}$)

for iter $\in$ $range(0, 10)$ /* 10 imputation iterations */
dofeatures $\leftarrow$ Sort_By_Percent_Retained(Features($data_{imp}$))

for $feature \in features$ **do** f_{imp} $\leftarrow$ Train_And_Impute($model$, $data_{imp}$, $feature$)

$data_{imp}$ $\leftarrow$ Concatenate(f_{imp})

return $data_{imp}$, $model$

Algorithm 3 Data Imputing

Procedure Imputing ($X_{normalized}$, y, $model$)
/* input comes from a single study or from joined studies dataset */
X_{masked} $\leftarrow$ Mask_10_Percent($X_{normalized}$)
X_{masked} $\leftarrow$ Mask_10_Percent(y)
$data_{masked}$ $\leftarrow$ Concatenate(X_{masked}, y_{masked})
X_{imp}, $model$ $\leftarrow$ Imputation_Training(X_{masked}, $model$)
y_{imp}, $model$ $\leftarrow$ Imputation_Training(y_{masked}, $model$)
/* Bag of ElasticNets is the best model for imputation */
$data_{imp}$ $\leftarrow$ Concatenate(X_{imp}, y_{imp})
Evaluate_Masked_Imputed_Data($data_{imp}$, $data_{masked}$)
return $data_{imp}$, $model$

lowest missing rate among the features not handled in the current outer iteration step, and the outer level iteration step finishes after all features have been imputed within that step.

Since missForest relies on decision trees, we suggest using AdaBoost and Bagging to impute features instead of simple regularized regression. We accomplish this by combining multiple elastic net (ET) models using boosting and bagging techniques to predict missing values of the feature with the lowest missing rate that was not already imputed in the current outer iteration step. We evaluate the effectiveness of this approach in our experiments.

The longitudinal youth anxiety dataset suffers from missing values in both explanatory (X) and predicted (y) variables, with predicted variables exhibiting partial correlation due to comorbidities associated with youth anxiety. The proposed imputation algorithm leverages correlations among the observed data, which can lead to an underestimation of the

Algorithm 4 Training Prediction Model

Procedure Prediction ($data_{imp}$)
/* input comes from a single study or from joined studies dataset */
$X_{tr}, y_{tr}, X_{val}, y_{val}, X_{test}, y_{test} \leftarrow$ Split($data_{imp}$, 0.70, 0.15)
$best_model \leftarrow LinearPrediction$
$best_RMSE \leftarrow 10$

for $hyperparameter \in hyperparameters$ **do** $model \leftarrow$ Prediction_Alg_Training(X_{tr}, y_{tr})
$predicted \leftarrow$ Predict($X_{val}, model$)
$RMSE_{res} \leftarrow$ Calculate_RMSE($y_{val}, predicted$)

if $RMSE_{res} < best_{RMSE}$ **then** $best_{RMSE}$ =RMSE$_{res}$
best$_{model}$ = $model$ **return** Predict($X_{test}, best_{model}$)

prediction error, as with all imputation methods. To mitigate the risk of leakage between X and y variables and reduce the underestimation of prediction error, we suggest training separate imputation models for X and y variables. Detailed steps for data cleaning, imputation model training, and data imputation are provided in Algorithms 1, 2, and 3.

6.3.2 Prediction methods

After imputing the datasets, we proceeded to train and test various regression models. We hypothesized that the random forest model used in Lebowitz et al. (2021) might not be the best choice for a regression task where the majority of the variables are numerical. For more information about training the prediction model, please refer to Algorithm 4.

6.3.3 Joining multiple studies

We propose combining multiple studies into a single dataset to address the challenges posed by the high percentage of missing values, a small number of samples, and a large number of features. Combining datasets can increase prediction confidence and allow for the prediction of diagnoses that were not recorded or reported in a single study. Additionally, it allows us to evaluate methods for studies with small sample sizes. However, due to different biases and missing rates in each study, we also investigate when combining data is most advantageous.

Algorithm 5 Domain Adaptation

Procedure DomainAdaptation ($X_{finished}$, $y_{finished}$, X_{drop})
/* X and y finished come from a single study or joined dataset */
$model \leftarrow$ Bag_Of_Elastic_Nets_Imputation_Model
$fin_{imp}, model_{imp} \leftarrow$ Imputing($X_{finished}$, $y_{finished}$, $model$)
$drop_{imp}, model_{imp} \leftarrow$ Imputing(X_{drop}, y_{drop}, $model_{imp}$)
$model \leftarrow$ OMP_Training($X_{finished}$, $y_{finished}$)
$predicted_{drop} \leftarrow$ Predict($drop_{imp}$, $model$)
return $predicted_{drop}$

6.3.4 Domain adaptation

Youth anxiety studies often experience participant dropouts over the extended study period, leading to missing data that can bias the results. To address this issue, it would be useful to predict the outcomes of withdrawn patients. However, these data have a higher missing rate and an additional bias, making accurate predictions challenging. In this paper, we propose an imputation approach that combines data from multiple studies to improve transfer learning predictions for withdrawn patients. The domain adaptation procedure is outlined in Algorithm 5.

6.4 Experimental Evaluation

We evaluate the proposed approach alongside state-of-the-art methods on data from youth anxiety studies.

6.4.1 Data

Data from various studies conducted in the last 30 years by different research teams across different countries (primarily in the United States) are integrated, utilizing a similar randomized controlled trial approach within samples of youth diagnosed with primary anxiety disorders. The primary difference between studies lies in the treatments used, such as treatment protocol and session number. The variables in the data are obtained from surveys. Independent evaluators trained for reliability used a semi-structured diagnostic interview to assess the diagnoses of the youth and their caregivers. Diagnoses were assigned a sever-

Study	Samples	Features	Predicted variables	% missing
S1	49	27	6	43%
S2	99	30	6	61%
S3	138	66	7	57%
S4	422	55	6	14%
S5	137	26	4	40%
S6	38	33	6	61%
S7	114	37	7	42%
S8	76	21	6	53%
S9	88	25	0	70%
Joined	1161	108	7	40%

Table 6.1: Statistical overview of youth anxiety data. Data comes from studies S1-S9 in the order referenced in introduction.

ity level between 0-8, with higher numbers indicating greater severity. Basic demographic information was also collected from the youth and their caregivers through questionnaires. Since the original data from studies contained too many variables, we selected the features based on domain knowledge from the clinical psychology literature on treatment outcome predictor and moderators. [10]

In this study, the severity of diagnoses before treatment is included as a feature, while the severity of diagnoses after treatment is considered the predicted variable. Numerical variables are used to represent severity scores. The categorical features, such as sex, race, ethnicity, study, and starting condition, are fully observed. On the other hand, all imputed features are numerical.

Table 6.1 presents the statistical information for each individual study and the combined dataset. Notably, most studies have a number of features that is similar to or greater than the number of samples, and all studies except for the largest one have fewer than 140 patients. The missing rate in the data is greater than 40% in all studies except for the largest one. Moreover, seven out of nine studies lack data for at least one of the predicted variables.

[10] The list of features and software implementation can be found here: https://github.com/marija-stanojevic/anxiety_ml.

Please note that these statistics do not take into account patients who dropped out of the treatment. To examine the transfer learning properties to withdrawn patients, we extracted data for the 202 youth who dropped out early. Study 9 exhibits an extremely high missing rate (70%) and does not contain any data for predicted variables.

6.4.2 Baseline methods

The commonly used imputation methods for missing data can be categorized into four groups: 1) statistics-based (mean, median, mode, random, MCMC); 2) nearest-neighbors-based (KNN, FKM, KI, FCKI, MICE); 3) decision trees and random forests-based (decision tree with a surrogate, missForest); and 4) matrix decomposition techniques-based (soft imputation, singular value decomposition (SVD)). We assess the applicability of all of these algorithms to the youth anxiety dataset.

However, FKM, KI, FCKI, and MICE cannot be used as they require fully observed samples. Therefore, we use EM instead of MCMC. Our missForest algorithm is named II with Extra Trees, and DT with a surrogate is one iteration of II with DT.

While our proposed method uses iterative imputation with regularized regression (ridge, elastic net, or lasso) and boosting (AdaBoost) and bagging algorithms, we also test other regression options in integration with iterative imputation, including: 1) linear regression; 2) Bayesian ridge (BR) regression; 3) orthogonal matching pursuit (OMP); 4) Bayesian automatic relevance determination regression (ARD); 5) KNN; 6) random forest regression (RF); and 7) gradient boosting regression ensembled with decision trees (GB).

To evaluate the influence of imputation on the prediction task and to understand domain adaptation performance, we use the same regression algorithms that are used as part of iterative imputation.

6.4.3 Dataset Cleaning

Values 888, indicating that the measure was not included in the trial, and 999, representing other missing values, are imputed with unknown values. Study code and treatment information are encoded as binary features, resulting in nine features for each study and eight for each treatment type. The race variable is transformed into four features: ”White”, ”Black”, ”Asian”, and ”Other”. Patients who withdrew from the treatment are separated into a distinct dataset, which will be used for domain adaptation tasks.

6.4.4 Imputation evaluation

To assess the effectiveness of the proposed imputation methods in comparison to the state-of-the-art techniques, we randomly mask 10% of the non-missing values in the dataset and train the imputation algorithms on the masked dataset. We calculate the root mean square error (RMSE) by comparing the original and imputed values on the masked data (Full RMSE). To evaluate the robustness of the methods, we repeat the imputation process ten times on different masked dataset versions and report the mean and standard deviation (STD). When imputing data separately for each study, we join the datasets before calculating the RMSE on all data combined (Avg study RMSE). We also calculate the imputation RMSE for each study separately to understand the performance of imputation on each study. For this, we perform imputation on five randomly masked versions of data from each study and report the average and standard deviation of RMSE.

6.4.5 Comparison of the imputated datasets on prediction task

While some pre-existing imputation methods are limited to imputing data only on the same dataset on which the imputation was trained, the proposed methods can be trained on the available data and used to impute unseen data. For the prediction task, the dataset is divided into training, validation, and test sets with a ratio of 70:15:15, using only data from patients who completed the treatment. The features in each set are normalized, and the im-

putation model is trained on the training set. The validation and test sets are imputed using the trained imputation model, and the imputed training set is used to develop prediction models. The prediction models are then evaluated using RMSE on non-imputed predicted values of the validation and test sets to avoid underestimation due to imputation of predicted variables. RMSE is also calculated on all predicted values in the validation and test sets and compared to the previously calculated RMSE to understand the underestimation caused by imputation. In both cases, RMSE is calculated on the validation and test sets, and the results are averaged to find the RMSE confidence interval.

To evaluate the prediction performance for each study separately, we would ideally split the data into training, validation, and testing sets, and then use the same imputation and prediction process as described in the previous paragraph. However, many of the datasets are small and have a high missing rate, resulting in many validation and test sets having more features than samples, making them difficult to process. Therefore, the studies are first imputed using the joined dataset as described earlier. Then, the imputed data is split into studies, and each study's data is further divided into training, validation, and test sets. Finally, the prediction process described in the previous paragraph is applied.

6.4.6 Domain adaptation task

First, the features of withdrawn patients are normalized. Then, we calculate the mean and standard deviation of imputation RMSE on the withdrawn data using five randomly masked and then imputed versions of the dataset. Next, we train and evaluate the imputation model on the same dataset. In addition, we evaluate the imputation model trained on the original dataset by using five randomly masked and then imputed versions of the withdrawn patients' dataset. As before, we use two independent imputing models for features and predicted variables to prevent data leaks. We use models trained on both the joined dataset and each study separately to predict the withdrawn patients' dataset values. Finally, we calculate the RMSE of predicted values on non-imputed predicted values and all predicted

	Regression alg.	Full RMSE	Avg study RMSE
Baseline methods	Mean	5.758 ± 0.188	5.683 ± 3.916
	Median	5.832 ± 0.210	5.965 ± 4.275
	KNN	4.665 ± 0.202	4.837 ± 3.183
	Soft Impute	4.990 ± 0.188	5.298 ± 2.834
	SVD	6.527 ± 0.377	9.104 ± 5.293
	EM	7.960 ± 0.265	7.857 ± 5.626
	II DT	5.633 ± 0.085	6.131 ± 3.489
	II Extra Trees	3.984 ± 0.177	4.300 ± 2.038
Proposed methods	II Linear	8.619 ± 1.947	Not valid
	II BR	4.056 ± 0.115	4.451 ± 2.830
	II Ridge	4.284 ± 0.398	5.882 ± 3.251
	II Elastic Net	3.886 ± 0.145	4.430 ± 2.992
	II Lasso	3.935 ± 0.129	4.410 ± 2.909
	II OMP	3.982 ± 0.229	4.291 ± 2.669
	II ARD	7.225 ± 3.549	8.863 ± 9.864
	II KNN	4.206 ± 0.188	4.716 ± 2.829
	II RF	4.059 ± 0.204	4.213 ± 2.183
	II GB	3.913 ± 0.148	4.457 ± 2.379
	II Ada Boost	3.903 ± 0.180	4.476 ± 3.322
	II Bagging	$\mathbf{3.809 \pm 0.181}$	$\mathbf{4.195 \pm 2.869}$

Table 6.2: Imputation evaluated on joined dataset (Full RMSE) and average RMSE of imputation on separate studies (Avg study RMSE)

variables.

6.5 Results

The root mean square error (RMSE) values for imputation are computed by comparing the actual values with 10% of randomly masked values in the dataset. In contrast, the RMSE values for prediction measure how well the model predicts the severity of diagnoses on an intensity scale of 0-8, where values 0-3 indicate milder cases and 4-8 denote severe cases. Only the difference between predicted and observed data, excluding imputations of predicted variables, is considered when reporting the prediction RMSE values.

6.5.1 Imputation results

The performance of imputation methods is assessed using RMSE on a normalized dataset in two scenarios: 1) imputation on the joined dataset (Full RMSE), and 2) imputation on each study separately followed by combining the imputed studies and calculating the RMSE (Avg study RMSE). The results for both cases are presented in Table 6.2, which is divided into two parts: baseline methods and proposed methods.

II with extra trees is the best performing baseline method, and its results are consistent with the existing literature. However, six proposed methods exhibit better Full RMSE than II with extra trees, with the bag of elastic nets performing the best. The two best results are obtained by combining II with boosting and bagging of regularized regression models, but all six models that outperform II with extra trees combine II with regularized regression. Additionally, three proposed methods exhibit better average study RMSE, and the bag of elastic nets yields the best result in this scenario as well.

6.5.2 Performance comparison between joined dataset and single studies

The results presented in Table 6.2 indicate that using a joined dataset improves the RMSE performance of the best imputation model (II with a bag of elastic net regressions) by 9.3%. More notably, it also reduces the variability of imputation performance across randomly masked data. Specifically, the best imputation method exhibits nearly 16 times smaller standard deviation (STD) when applied to the joined dataset as compared to single studies' imputation.

However, it's worth noting that the iterative imputation method with a linear model may lead to infinite predictions in some cases due to a small percentage of known values within a single study.

The RMSE of imputation for each study is presented in Table 6.3. Study 9 exhibits high imputation RMSE due to its exceptionally high missing rate (70%) and lack of prediction variables. Because of this discrepancy in the joined dataset, only three studies have worse

Study	Imp. RMSE	Pred. RMSE	Coverage
S1	2.885 ± 0.842	1.029 ± 0.311	25%|1.7%
S2	3.983 ± 0.201	0.9034 ± 0.041	28%|4.1%
S3	3.229 ± 0.164	1.335 ± 0.192	60%|16%
S4	3.018 ± 0.087	1.359 ± 0.062	51%|39%
S5	2.805 ± 0.188	1.529 ± 0.030	24%|5.3%
S6	4.942 ± 1.207	1.439 ± 0.087	30%|2.1%
S7	1.594 ± 0.120	1.618 ± 0.358	35%|8.0%
S8	3.661 ± 0.549	1.814 ± 0.475	19%|2.1%
S9	11.639 ± 1.284	Impossible	23%|3.0%

Table 6.3: Performance of imputation (Bagging with ElasticNet) and prediction (Bayesian Regression) on imputed data for each study.

imputation RMSE than the joined dataset.

The coverage column indicates the percentage of observed columns and missing values used to evaluate the study's imputation in comparison to the joined dataset. Coverage displays a wide range of variability in known values (1.7% - 39%) and covered features (19% - 60%) among the studies.

It is noteworthy that the studies with high imputation RMSE (S2, S6, S9) also exhibit low confidence in the results, indicating that the correlation among available data is insufficient to impute effectively. This is not surprising as these three datasets have a missing rate of 61%-70%.

Figure 6.1 highlights the similarity between studies 1 and 2, but Table 6.1 reveals a significantly higher RMSE for S2, which can be attributed to its higher overall missing rate.

Among the single studies, S4 stands out as the most significant dataset, with a high level of confidence in its imputation and lower RMSE mean and standard deviation. Therefore, if the 55 features and six predicted variables in S4 are sufficient, training an imputer and predictor using only this dataset instead of the joined dataset would be a good option.

By combining statistics from Tables 6.1 and 6.3, it becomes clear that imputation confidence and RMSE are most influenced by a dataset's missing rate, followed by its size

and structure of missing values. Additionally, imputation RMSE is not a reliable predictor of prediction RMSE, but imputation confidence is highly correlated with prediction confidence. While the proposed imputation methods perform better than those commonly used in the literature, this advantage does not guarantee better prediction results for a small study with a high missing percentage.

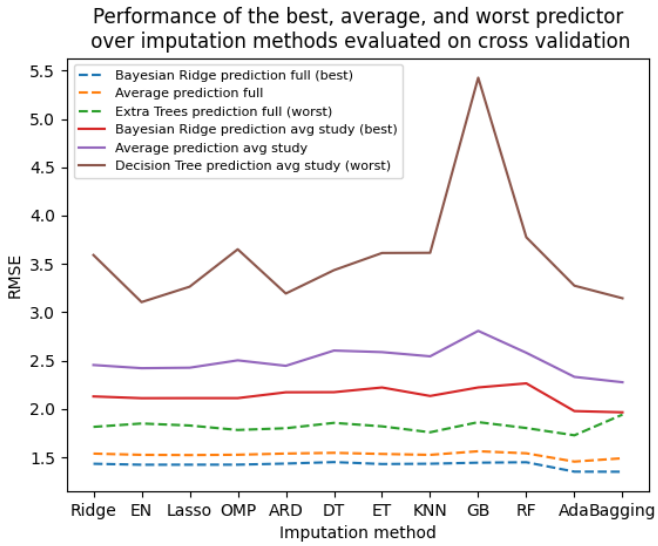


FIGURE 6.2: RMSE of the best, an average of all predictors, and worst over different imputation methods on test data. **Prediction full** lines describe prediction made on a joined dataset, and **prediction avg study** lines show average prediction RMSE when a different model is trained on each study.

In Figure 6.2, it is evident that using a joined dataset for training leads to improved and stabilized imputation RMSE. The choice of predictors has a greater impact on prediction results than the imputation method, especially when selecting from our proposed predictors, which are all performing well. However, AdaBoost and bagging of elastic net regressors still outperform the other imputation methods slightly. It is worth noting that imputation using random forest and gradient boosting of decision trees results in the worst prediction outcomes.

Although the best prediction models yield acceptable RMSE, the worst model trained on each study separately provides insignificant results due to the scale of the predicted

value. The significance of the joined dataset is also evident, as even the worst prediction model on the joined dataset outperforms the best prediction model trained on single datasets.

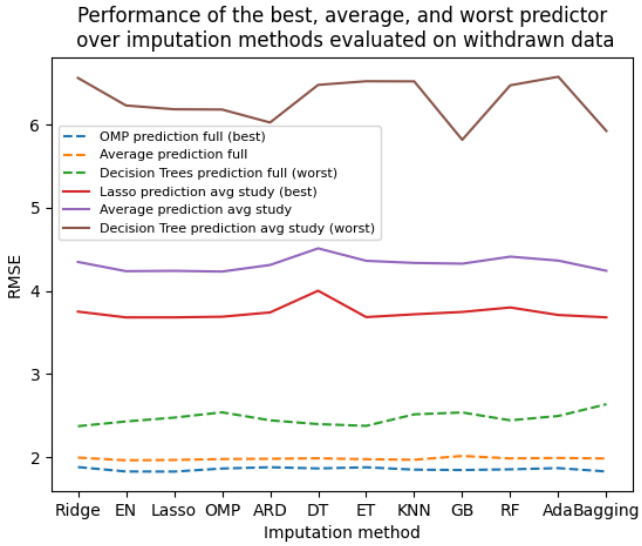


FIGURE 6.3: RMSE of the best, an average of all predictors, and the worst predictor over different imputation methods on data from withdrawn patients. Lines have the same meaning as in Figure 6.2.

6.5.3 Domain adaptation - withdrawn patients

To assess the domain adaptation capabilities of the learned predictors, we replace the test dataset with the dataset of withdrawn patients and measure the RMSE for each combination of imputation and prediction methods. Figure 6.3 illustrates the prediction RMSE given different imputation methods. The performance patterns on the withdrawn patients dataset are comparable to the test dataset's, but the prediction RMSE is higher for all predictor types. Although the best predictors are still linear, they differ from those in Figure 6.2. As hypothesized, the significance of joined datasets is even more pronounced, as all predictors trained on single datasets produce meaningless predictions with RMSE exceeding 4 points. Predictors achieve a 50% RMSE reduction when using the joined dataset. While all the imputation models perform similarly on the best predictors, the bagging of elastic nets has

a slight advantage.

6.6 Summary

This chapter explores domain adaptation when both source and target datasets have high-missing rates. We address this challenge using a youth anxiety clinical dataset. Youth anxiety disorders are highly prevalent and associated with significant functional impairments, making it a pressing problem that needs to be addressed. Using machine learning methods for the imputation and prediction of outcomes can greatly improve the understanding of treatments in such studies. Our proposed iterative imputation with a bag of elastic net regressions imputes missing data better than traditional imputation methods and allows for the best prediction results. Leveraging joint datasets is significant in successful prediction, especially when the target variable is from a different domain of withdrawn patients. Our results showed that pooling data from multiple studies resulted in smaller imputation and prediction errors and higher confidence than when studies were considered separately.

Our study highlights the importance of building and evaluating helpful machine-learning tools for psychological and other longitudinal medical research studies with small samples, many features, and high missing rates. The proposed machine learning system needs to be evaluated for potential ethical challenges around imputation that may induce bias on an individual level. Overall, the application of machine learning methods in the analysis of such data can provide valuable insights into the treatment of youth anxiety disorders and other related mental health disorders.

CHAPTER 7

MITIGATING THE CURSE OF DIMENSIONALITY IN DOMAIN ADAPTATION

One of the challenges in domain adaptation is the curse of dimensionality, which refers to the difficulty in analyzing and modeling high-dimensional data. The curse of dimensionality can cause overfitting, underfitting, and generalization issues, which ultimately lead to poor domain transfer performance. To mitigate the curse of dimensionality in domain adaptation, several approaches have been proposed. One of the most common approaches is dimensionality reduction, which reduces the number of features in the data while retaining its essential characteristics. Another approach to mitigating the curse of dimensionality is feature selection, which involves selecting a subset of relevant features from the original dataset.

In this chapter, the goal is to predict if a biology research paper contains phylogenetics timetrees. The source dataset is collected by timetree database curators, but target datasets are downloaded from various internet sources and therefore have different distributions and quality. We examine the source and target similarity and propose methods to reduce their distribution differences. We also present a method to select relevant parts of research articles and reduce the input dimensionality in order to improve the classification performance

on both source and target datasets.

7.1 Introduction

Text mining has the capacity to transform the discovery of research articles that contain specialized knowledge in various domains from the vast and rapidly growing corpus of scientific literature. In the field of evolutionary biology, the availability of genome sequencing technology at an affordable cost has brought about a revolution in the construction of the tree of life. Peer-reviewed articles featuring timetrees - phylogenetic trees scaled to time - are being published in large numbers each year across numerous journals (Kumar et al. (2022)). These timetrees are presented in visual formats, displaying hierarchical trees of species, and are not easily accessible through text searches when seeking information on species divergence times and relevant articles. The TimeTree of Life (TToL) project was launched in 2004 to locate research articles with timetrees, collect and standardize them, and perform meta-analysis to generate a global timetree (Hedges et al. (2006); Kumar et al. (2022)). The most recent release of the global timetree comprises over 135,000 species gathered from more than 4,000 published articles (Kumar et al. (2022)), and this information is available through a user-friendly web resource, which has been cited thousands of times. The TToL website is widely used by students and the general public, as indicated by over 250,000 database queries per year.

The TimeTree database has been relying on manual curation by its curators since 2004, who use a combination of keywords to search the archives of Google Scholar and PubMed to generate a list of articles on the web. The curators then scan the article titles to identify potential timetree articles, which is a laborious, expensive, and visually taxing process. After this, the curators click the PDF or text-download link of the selected articles, which is yet another time-consuming step, followed by a manual inspection of the content. When a timetree is found, the article is retained for further processing.

Over nearly two decades, the curators have manually examined the titles of thousands

of articles and scanned 14,366 downloaded full-text articles in search of timetrees. Of these, 4,292 research articles were identified as containing timetrees, resulting in a manual true positive rate of 29.88%. These timetrees were then considered for curation into the TimeTree database (www.timetree.org). However, this approach of locating relevant research articles is burdensome, time-consuming, and prone to errors because the presence of a timetree in an article cannot be accurately predicted based solely on the title or abstract.

The TimeTree curators did not rely on text-mining tools, apart from internet searches, to locate research articles containing timetrees because no tools were available for analyzing the scientific literature corpus and discovering evolutionary knowledge. While computational text-mining methods and tools have been extensively used in medical and molecular biology research, our survey revealed that there have been no investigations into their effectiveness for retrieving articles containing evolutionary trees, nor are there any guidelines on the optimal tools for this domain. Therefore, the primary objective of this study was to assess the performance of simple and advanced text-mining techniques for identifying articles that contain timetrees. The TimeTree Finder (TTF) processes the relevant scientific literature corpus available from resources such as Google Scholar, PMC Open Access Database (PMC-OA), PMC Historical, bioRxiv, and scientific journals' websites.

In this study, we present the results of the implementation and analysis of TTF that utilizes algorithms such as TFIDF (Jones (1972); Luhn (1957)), Lasso (Tibshirani (1996)), and word2vec (Mikolov et al. (2013b,a)) to construct Google search queries, as well as state-of-the-art machine learning algorithms such as BERT (Devlin et al. (2019)), BioBERT, and SciBERT to detect the presence of timetrees in research articles. Since BERT-based state-of-the-art models can process up to 512 words, we trained them to classify research articles using only titles and abstracts, similar to the method used by the TimeTree project's manual curators.

As demonstrated in the results below, relying solely on the title and abstract to identify relevant timetree papers was the least effective approach. This led us to test the hypothesis

that analyzing full-text articles could lead to more accurate retrieval. While this approach did improve results, it still produced many errors due to misleading information in the text. To address this, we proposed a new approach of training models on carefully excerpted texts around the figures, as timetrees are typically displayed in image form in research articles.

In the following sections, we describe the TTF tool we developed, which achieved an impressive F1 score of 0.88 in classification, despite the high dimensionality of the problem. We also discuss the machine learning models and features, such as word and word pairs, that are critical for accurately identifying articles containing timetrees. Finally, we explore how this system can be adapted to extract relevant data and extract valuable insights from various types of texts, such as grant proposals, reports, research papers, web pages, and news articles, for scientific and non-scientific applications.

7.2 Related Work

The field of text mining has been widely explored in the medical and molecular biology domains, where it has been applied to extract knowledge from the scientific literature. For instance, research has been conducted to extract gene-disease associations (Bhasuran and Natarajan (2018)), identify relevant articles for clinical practice guidelines (Whittington et al. (2019)), and discover drug-drug interactions (Percha et al. (2012)). In contrast, text mining techniques have not been widely used in evolutionary biology, and there is a lack of investigation into their potential for discovering evolutionary knowledge.

Previous work in automated literature curation has focused on using machine learning and natural language processing techniques to extract information from scientific articles. For instance, PubMed's core bioinformatics program, Textpresso, uses natural language processing and ontology annotation to extract specific information from scientific articles (Müller et al. (2018)). Similarly, several studies have applied machine learning techniques to automatically extract relationships between biological entities Perera et al. (2020). How-

ever, these studies have not focused on the identification of articles containing evolutionary trees. However, to our knowledge, no previous work has investigated the use of machine learning techniques for identifying articles containing evolutionary trees.

Our study seeks to fill this gap by investigating the performance of text-mining techniques for identifying research articles containing timetrees. We used a combination of keyword searches, machine learning algorithms, and advanced natural language processing techniques to create a tool that achieves high accuracy in identifying relevant articles. Our work also demonstrates the potential for text-mining techniques to discover valuable evolutionary knowledge and highlights how these techniques can be applied to other scientific and non-scientific domains.

7.3 Methodology

Figure 7.1 presents the outline of the TimeTreeFinder (TTF) system, which automatically searches, extracts, and analyzes the scientific literature and individual articles to identify relevant articles.

7.3.1 The ground truth TimeTree (TT) dataset

For the past two decades, the TimeTree project team has conducted a manual examination of 14,366 articles, categorizing them as positive (4,292) if they contained descriptions, tables, or figures pertaining to molecular divergence times. Conversely, articles that did not meet this criterion were labeled negative (10,074). This dataset was utilized as the ground truth in all our analyses and experiments.

7.3.2 Processing the corpus of scientific literature

For this study, we obtained scientific articles from various online resources, namely Google Scholar, journals that contributed most of the articles in the TimeTree dataset, PMC Open Access Subset (PMC-OA), PMC Historical Dataset, and bioRxiv machine access and text/data

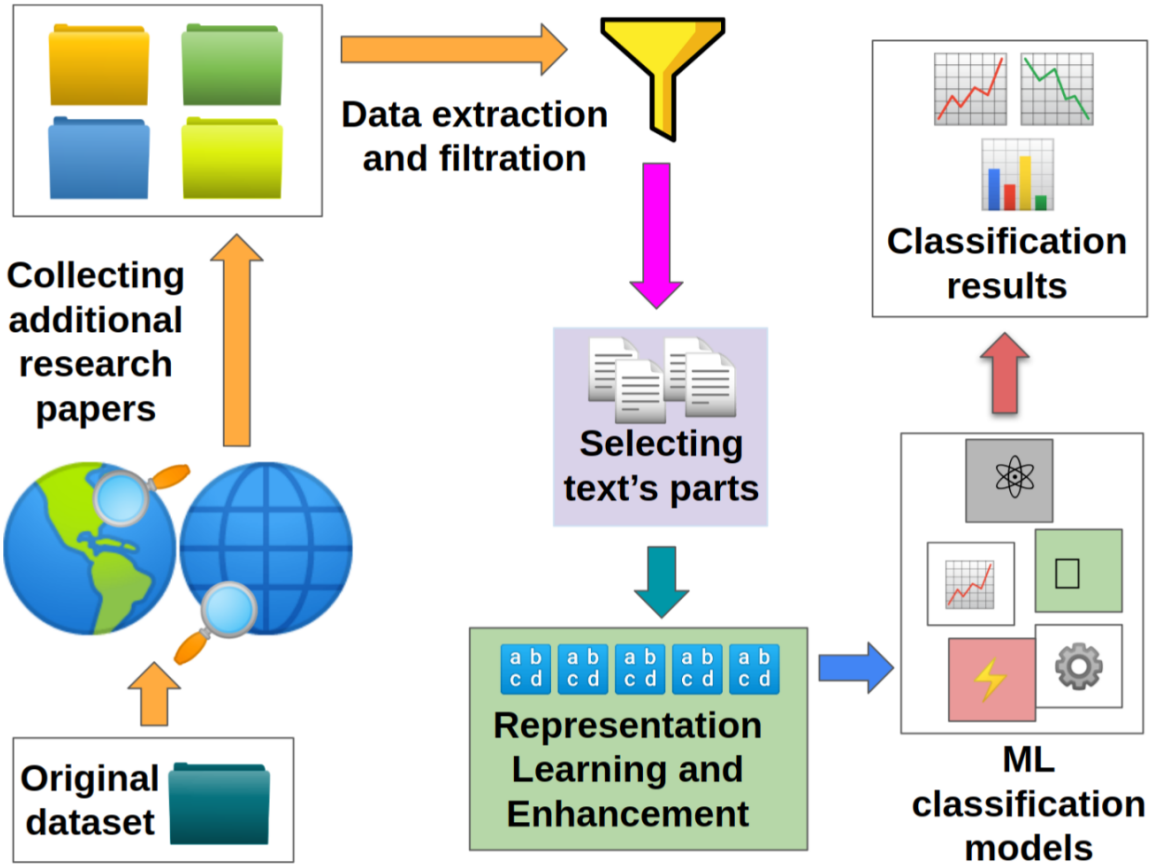


FIGURE 7.1: Overview of the proposed TimeTreeFinder (TTF) system, including searching, collecting, and analyzing research articles to discover time tree-containing articles.

mining resources. As these resources contain millions of articles, it is impractical to download and analyze all the information manually or automatically. To address this issue, we limited our search to relevant articles only, which we retrieved when possible from Google Scholar, and filtered out irrelevant articles from all sources.

To process the Google Scholar dataset, we employed the following approach. First, we identified representative phrases (words and word pairs) from the ground truth TT dataset. We used the term frequency-inverse document frequency (TFIDF) method (Jones (1972); Luhn (1957)) to create features (i.e., words and word pairs) and applied L1 regularized logistic regression, LASSO (Tibshirani (1996)), to classify articles from the TT dataset. LASSO suppresses irrelevant features and assigns positive coefficients to the most impor-

tant phrases for detecting research articles containing timetrees. This list of phrases, named PL, included terms such as *mya*, *divergence time*, *myr*, *2015*, *the divergence*, *miocene*, *node*, *diversification*, *beast*, *million years*, *mybp*, *split*, *extant*, *diverged*, and *chronogram*.

Next, we applied the word2vec algorithm (Mikolov et al. (2013b,a)) to identify additional informative words that were frequently used in the same contexts as the PL words. We then filtered the resulting collection of phrases by frequency to remove rarely occurring combinations and misspellings. The resulting list of features, WL, was combined with PL to form the final list of phrases, FL.

We designed two types of queries using phrases from FL. The **FL-simple query** utilized one phrase, and the **FL-complex query** used a logical combination of phrases. For creating FL-simple queries, we had to select phrases not often used in other sciences. In FL-complex queries, we could search for papers with more common phrases, such as ”million years”, if we give additional context to help the search focus on relevant papers. Four FL-complex search queries are created, and they are given below. We have also selected different publishing years for both types of queries, focusing on research from the last two decades only. **Simple Search Query Phrases:**

1. *mya*
2. *pliocene*
3. *phylogenetic*
4. *mybp*
5. *carboniferous*
6. *phylogram*
7. *myr*

8. *timetree*

9. *mybp*

10. *treeannotator*

Complex Search Queries (each query was run with a different span of years so that all papers from the 1990s onwards are searched):

1. *species AND (ma OR mya OR myr OR divergence OR miocene OR diversification OR beast OR pliocene OR carboniferous OR outgroups OR mybp OR extant OR diverged OR treeannotator OR timetree OR kya OR million years) AND year=2016*

2. *(chronogram OR phylogeny OR time tree OR phylogram OR BEAST drummond common ancestor OR crown age OR dated phylogeny OR dating analysis divergence time OR fossil calibrated OR fossil calibration OR fossil record) AND year = 2019*

3. *molecular AND (ma OR mya OR myr OR divergence OR miocene OR diversification OR beast OR pliocene OR carboniferous OR outgroups OR mybp OR extant OR diverged OR treeannotator OR timetree) AND year = 2018*

4. *(molecular clock OR MrBayes node age OR relaxed clock OR stem age OR strict clock time calibrated) AND from_year = 2009 AND to_year=2015*

These efforts resulted in the retrieval of 5,070 articles using the FL-simple query and 8,013 articles using the FL-complex query. Although we had anticipated a greater number of articles, Google Scholar's restrictive policies prevented us from downloading too many PDFs, and our university library lacked access to certain journals. However, we can rerun the automatic search and collect the articles once they become available.

In contrast to Google Scholar's search engine, we had to develop customized approaches for collecting articles from other sources. We designed software to gather data from 18 journals using Temple University's access. Additionally, we manually downloaded PMC

datasets (PMC-OA and PMC Historical) using FTP bulk download. We also created software that utilizes the bioRxiv API to obtain information about all articles published since 2013, the year the server was launched. Downloaded data was in PDF format from all sources except PMC, which already contained textual files. During data collection, we produced metadata files for each source, which included the title and other available information, such as a list of authors, a link, the year of publication, and more.

In this instance, we obtained all available PDFs without applying any keyword filtering. The PMC files were compressed in a text format, so we extracted individual files automatically and gathered the necessary text. Since all other resources comprised PDFs, we utilized the Python library ”textract” to extract textual information. Some journals had PDFs of scanned papers published over a century ago, from which we extracted the text using the Python library ”ocrmypdf.” We excluded conversions from PDF to text that resulted in extracting fewer than 100 characters. We also removed errors from downloading, such as duplicate or missing files.

As expected, we downloaded many articles, ranging from 117,000 articles from biological journals to 4.17 million articles from PMC-OA. Then, we filtered them using keywords to decrease irrelevant articles.

A few filtering techniques were tested using the following criteria:

1. If the sum of occurrences count of all words in the list is greater than a threshold;
2. If more than a threshold of words from the list appears in the text;
3. If one of the words in the reduced list exists in the paper;
4. Combination of criteria 1 and 2.

Criterion three was chosen as it was the simplest and fastest. In addition, it ensured that only relevant papers that might contain time tree data needed for the database are chosen,

and it is susceptible to bias of year published, journal, publisher, or dataset source. Reduced list required text to contain one of the strings:

1. *mrbayes*;
2. *beast*;
3. *timetree*;
4. *treeannotator*;
5. *phylogen*;
6. *beauti*;
7. *reltime*;
8. *multidivtime*;
9. *r8s*;
10. *treefinder*;
11. *path o gen*;
12. *phytime*;
13. *time tree*;
14. *phylogram*;
15. *relaxed clock*;
16. *mega*;
17. *molecular clock*;

18. *strict clock*.

Phrases that contain dashes instead of space between words are also considered.

This procedure removed 51% of the articles downloaded from journals, 90% of PMC-OA articles, 94% of PMC Historical articles, and 75% of bioRxiv articles. The resulting collections were candidates for finding timetrees. These data collections were then subjected to representation learning and classification analysis.

7.3.3 Representation learning

We evaluated six different representation learning methods: TFIDF (Jones (1972); Luhn (1957)), doc2vec (Le and Mikolov (2014)), BERT (Devlin et al. (2019)), SciBERT Beltagy et al. (2019), DistilBERT (Sanh et al. (2019)), and BioBERT (Lee et al. (2020)). Doc2vec utilizes a small neural network to create sentence and paragraph vector representations. TFIDF produces a matrix with document vectors as rows, word vectors as columns, and TFIDF scores as values for each word-document combination. While TFIDF captures information from the entire document, doc2vec relies on a local context, making it less effective for long research articles with distant dependencies. BERT-based models, like TFIDF, learn from the entire sequence but can handle sequences up to 512 words. We selected SciBERT and BioBERT, which specialize in research and biological articles, respectively. Additionally, we tested DistilBERT, a faster and lighter version of BERT that has been shown to achieve similar performance (Sanh et al. (2019)).

7.3.4 Representation enhancing

As the representation learning process is separate from classification, the use of labels is unnecessary at this stage. To enhance the data and contexts available for representation learning, we incorporated the ground truth TT dataset along with the filtered datasets we downloaded. To assess whether the use of additional data for representation learning leads to improved classifications, we applied classification models to the representations learned

solely from the TT dataset, as well as those learned from all available data.

7.3.5 Classification

We aimed to classify research articles into the biological domain of timetrees using nine diverse, commonly used classification methods. These methods are L1-norm regularized logistic regression (L1), decision tree (DT), random forest (RF), k-nearest neighbors (KNN), support vector machines (SVM), adaptive boosting (Ada), gradient boosting (GB), bagging (Bag), and classification layers of BERT-based methods (BERT, DistilBERT, SciBERT, BioBERT). Research articles are complex and high-dimensional data, making it challenging to form hypotheses about their classification. These diverse classification methods represent various desirable properties, enabling us to explore which approaches perform best.

7.3.6 Evaluation

We examined the classification outcomes of the models to comprehend the instances when they made errors. The underperforming models missed a significant number of relevant articles (type II error) while retaining many irrelevant articles (type I error). Given that curators are required to review data from all research articles predicted as positive, the objective is to minimize the type I error to reduce human effort. We also aimed to minimize the type II error as much as possible since the primary objective is to identify as many articles containing a timetree as possible. The F1 score (ranging from 0 to 1) was used as the main metric since it is maximized when both types I and II errors are minimized. We conducted a grid search of hyperparameters to fine-tune each model and selected those with the best F1 score on the cross-validation dataset. Documents predicted as positive were verified by human curators to ensure that they contained a timetree and that the analysis of empirical datasets produced it.

7.3.7 Generation of text excerpts datasets

Three primary types of errors were observed in the misclassification of articles by machine learning models. The first error involved identifying articles that only discussed methodology or software for visualization, without containing any data on new timetrees. The second error resulted in many false positives, where articles discussed geological periods or species but lacked information on molecular divergence times. The third error resulted in false negatives, where research articles that described the time of divergence of species in small parts of the text were wrongly excluded.

To address these issues, we hypothesized that selecting relevant information from articles and classifying articles based on that information instead of the entire text would reduce these errors. This technique offers two benefits: firstly, shorter texts are easier for machine learning models to comprehend; and secondly, irrelevant text (such as introductions, references, and other descriptions) is automatically excluded in the search for a timetree.

To extract relevant information from research articles, we focused on figure captions and text passages mentioning figures, which often display timetrees. We included figures described and shown in both the main text and the supplement. To identify the relevant text, we used newline spacing to determine the start and end of the paragraph or caption mentioning the figure. However, because of varying formatting, including OCR transformation, we also took 400 (or 150) characters before and after the figure if we could not identify the exact start or end of the paragraph. We then labeled figures containing a timetree as positive based on 816 positively labeled papers and assigned a positive label to all extracted text discussing those figures. Text discussing other figures from those papers were labeled negative. We also labeled all text excerpts extracted from 504 negatively labeled papers in the TT dataset as negative. The resulting dataset included 7,094 extracted text excerpts, with 3,550 positively labeled.

In order to assess the amount of text context required for a model to effectively learn if a figure contains a timetree, we conducted a performance comparison using both 800 and 300 characters surrounding the location where a figure is mentioned. We then applied the previously described process, which includes representation learning, representation enhancing, classification methods, and evaluation, but using datasets comprised of text excerpts and short text excerpts, rather than full text datasets.

7.3.8 TimeTreeFinder (TTF) tool

The TimeTreeFinder (TTF) tool is a comprehensive solution for conducting research on phylogenetics. It allows for efficient searching, collecting, extracting, filtering, and normalizing data from various relevant sources. The TTF tool also facilitates representation learning and classification method training on research articles from the TimeTree Dataset. It offers a text excerpt selection feature for timetree-related content and enables representation and classifier learning over those data. To ensure the accuracy and reliability of the learned models, the TTF tool validates them using the TimeTree Dataset, which is distinct from the training data, and evaluates their performance on newly collected datasets. A more in-depth explanation of the experimental settings, as well as the training and validation split, is available below.

7.4 Experimental Evaluation

In this section, we report the experiments and results conducted using TTF, a tool that automates the search, collection, filtering, and discovery of timetrees in the research articles described in Section 7.3. We first describe the properties of the collected datasets and compare them to the original data. Next, we compare the performance scores of various machine learning techniques used for data representation and classification, and suggest methods for selecting relevant text excerpts from papers.

7.4.1 Experimental Settings

Text normalization involves removing all non-alphabetic characters and converting the remaining text to lowercase. We shuffled the examples in the labeled dataset and divided them into training, cross-validation, and testing sets for whole paper and selected text classification. The ratios used were 73:7:20 and 66:14:20, respectively. To perform representation learning using TF-IDF, we chose words and word pairs that appeared in at least 5% and no more than 90% of the documents. These limits were selected to remove common stop words and rare words that are not essential for classification. When creating doc2vec representations, we trained a neural network for 20 epochs to generate the final document vector representations, each of 100 dimensions. The hyperparameters for enhanced representation learning remained the same.

For training BERT-based classifier models, we adopted a learning rate of 0.00002, as recommended by the original authors (cite). Two epochs of training were found to be optimal, and a batch size of 8 samples was used to overcome hardware limitations. We employed the HuggingFace transformers library implementation of the following BERT-based models: bert-base-uncased, distilbert-base-uncased, allenai/scibert_scivocab_uncased, and dmis-lab/biobert-base-cased-v1.1, as they are the best-performing versions of these models. We developed two trainers to handle different settings: a HuggingFace transformers trainer for simple training and evaluation, and a PyTorch trainer with an accelerator for more complex scenarios.

To discover the best classification model, we employ a grid search for hyperparameters. Standard practices are followed to set the hyperparameters for text representation and classification. We adjust errors by multiplying them with the inverse of class size, ensuring that the model is equally penalized for errors in each class, regardless of the number of available samples with that label. For DT, RF, SVM, Ada, GB, and Bag, we use a random state of 0 to produce consistent results. To ensure consistency, we conduct all experiments

five times, except for those that involve BERT, BioBERT, or SciBERT, which are repeated only twice due to limited access to GPU hardware. The results are presented as the mean and standard deviation.

Hyperparameters for classification models.

- **Logistic regression with L1 regularization** - regularization coefficient alpha tested on values 0.000001, 0.00001, 0.0001, 0.001, 0.01, and 0.1. Models are trained using the SGD algorithm with logistic regression loss, L1 penalty, and up to 500 iterations.
- **Decision Tree** - impurity decrease in decision node tested on values 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, and 1 using entropy criterion.
- **Random Forest** - impurity decrease in decision node tested on values 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, and 1 using entropy criterion.
- **K nearest neighbors** - k tested on values 1, 2, 3, 4, 5, 6, and 8.
- **Support vector machines** - regularization parameter tested on values 0.1, 0.3, 1, 3, 10, 30, and 100 combined with linear, polynomial, RBF and sigmoid kernels.
- **Adaboost** - number of estimators tested on values 40, 80, 160, and 320
- **Gradient boosting** - number of estimators tested on values 40, 80, 160, and 320
- **Bagging** - number of estimators tested on values 40, 80, 160, and 320

7.5 Results and Discussion

In this section, we discuss the results generated using TTF, which streamlines the search, collection, filtering, and timetree discovery processes in the research articles outlined in Section 7.3. We delve into the attributes of the collected datasets and evaluate how they stack up against the original data. We then compare scores obtained from various machine

learning techniques utilized for data representation and classification and suggest methods for extracting pertinent text excerpts from papers.

7.5.1 Most common words in the TimeTree Dataset

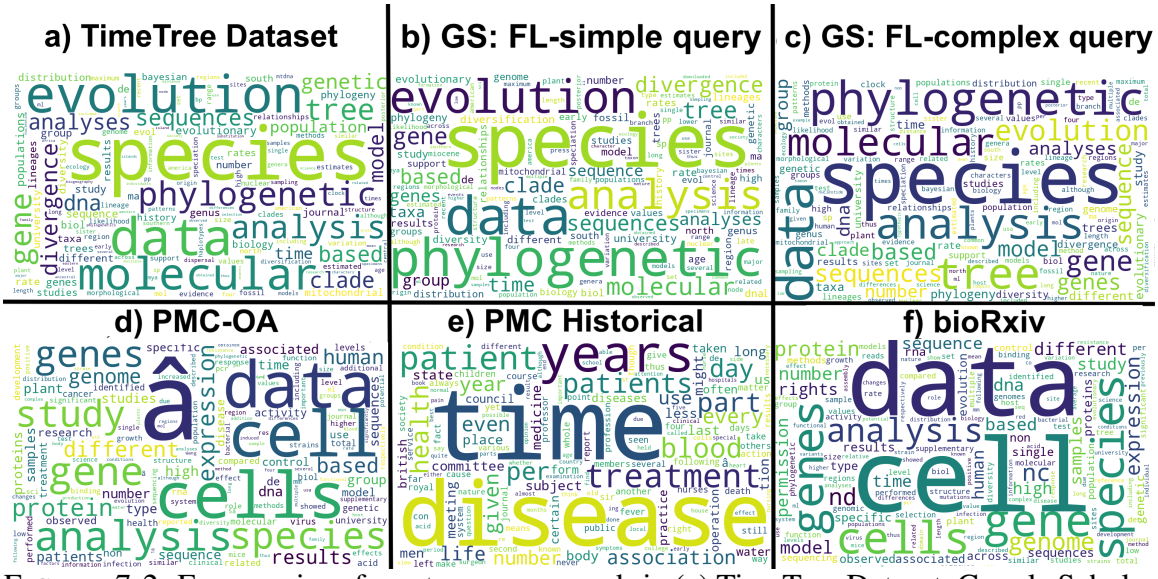


FIGURE 7.2: Frequencies of most common words in (a) TimeTree Dataset, Google Scholar Datasets from (b) FL-simple query and (c) FL-complex query, (d) PMC-OA, (e) PMC Historical, and (f) bioRxiv dataset.

The most frequent words in the TimeTree Dataset that was manually curated are displayed in Figure 7.2 a as a word cloud. The size of each word is proportional to its frequency in the TT Dataset. The most commonly occurring words are species, evolution, molecular, phylogenetic, data, gene, analysis, tree, genetic, sequence, clade, and divergence. We removed generic stop words and other common words that are irrelevant to phylogenetics from the word cloud illustrations (Figures 7.2, 7.3, and 7.5). Those words are *fig*, *figure*, *et*, *al*, *two*, *within*, *figs*, *three*, *table*, *found*, *among*, *using*, *shown*, *may*, *one*, *see*, *used*, *supported*, *new*, *well*, *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *i*, *j*, *k*, *l*, *m*, *n*, *o*, *p*, *q*, *r*, *s*, *t*, *u*, *v*, *w*, *x*, *y*, *z*, *dr*, *mr*, *org*, *biorxiv*, *preprint*, *peer*, *doi*, *review*, *copyright*, *version*, *http*, *https*, *author*, *license*, *international*, *certified*, *granted*, *reserved*, *posted*, *funder*, *holder*, *available*, *perpetuity*, *made*, *display*, *will*, *cases*, *first*, *hospital*, *case*, *medical*, *many*, *must*, *acc*, *allowed*,

without, *reuse*, *much*, *upon*, *now*, *general*, *great*, *present*, *said*, *work*, *small*, *man*, *london*, *large*, *little*, and *good*.

7.5.2 Properties of collected and filtered datasets

In this section, we examine the characteristics of the datasets collected using the approach outlined in Section 7.3. Figures 7.2b and 7.2c demonstrate the distributions of the most prevalent words in the datasets acquired from Google Scholar (GS) through FL-simple and FL-complex queries. These datasets exhibit considerable overlap with the TT Dataset, indicating that classifiers trained using the TT Dataset would yield satisfactory results when applied to GS datasets, streamlining the process of locating articles with timetrees. However, datasets procured from journals, PMC-OA, and bioRxiv could only be obtained in bulk and could not be scanned for relevant articles. Hence, we implemented a filtering step to identify relevant papers, which eliminates articles that do not feature at least one of the most relevant timetree terms (please see the complete list in the Supplementary materials). To verify its effectiveness, we tested this filtering system on the TimeTree Dataset. Notably, none of the positively labeled documents in the original dataset were removed, indicating that the false-negative rate was zero. Furthermore, only 9.6% of TTD papers in the true negative set were not retained, underscoring the effectiveness of the filtering system.

We utilized the filtering system on all datasets, resulting in a varying degree of retained articles. The PMC Historical datasets had the highest number of filtered-out articles (94% removed) due to their inclusion of unrelated topics to phylogenetics and timetrees. Moreover, this dataset contained articles published before the 1990s, lacking timetree information due to the field's significant growth in the last thirty years. The PMC-OA dataset, despite its medical focus, retained 10% of articles as it is more recent and has more phylogenetic data. The bioRxiv dataset retained 25% of articles since it is biology-focused and contains more recent research articles (published since 2013). Timetree publications have been more common in the past two decades, driven by the analysis of molecular datasets.

Finally, 50% of research articles from selected journals were retained since we collected solely from relevant journals.

After applying the filtering system to all datasets, we found that most articles were filtered out from the PMC Historical datasets, retaining only 6% of the articles due to the inclusion of articles on various topics unrelated to phylogenetics and timetrees. Additionally, this dataset included phylogenetics articles published before the 1990s, which did not contain information on timetrees due to the rapid growth of the field in recent decades. On the other hand, the PMC-OA dataset, while still mainly focused on medicine, retained 10% of the articles due to the more recent publications containing more phylogenetic information. For the bioRxiv articles, the filtering system retained 25% of the articles, as this platform primarily focuses on biology and contains more recent research articles (since 2013). Over the last two decades, the publication of timetrees has become more frequent, driven by the analysis of molecular datasets. Finally, when processing research articles from selected journals, we were able to retain 50% of the articles as we only collected research articles from relevant journals.

Figure 7.2 d-f shows the most common words in the PMC-OA, PMC Historical, and bioRxiv datasets. However, compared to the TimeTree Dataset, the PMC and bioRxiv datasets are not very similar even after applying filtering. While some common words such as data, species, and analysis are shared with the TT Dataset, the PMC datasets contain many medical words that are not frequent in the ground truth.

7.5.3 Classification models and their predictions

The distribution of common words in positively and negatively labeled papers in the TimeTree Dataset is presented in Figure 7.3, highlighting the challenge of the classification task. The Lasso model analysis of the dataset is illustrated in Figure 7.4, where phrases with positive and negative coefficients are depicted in blue and dark red, respectively. The diagonal axis represents words that have similar frequencies in both positively and negatively

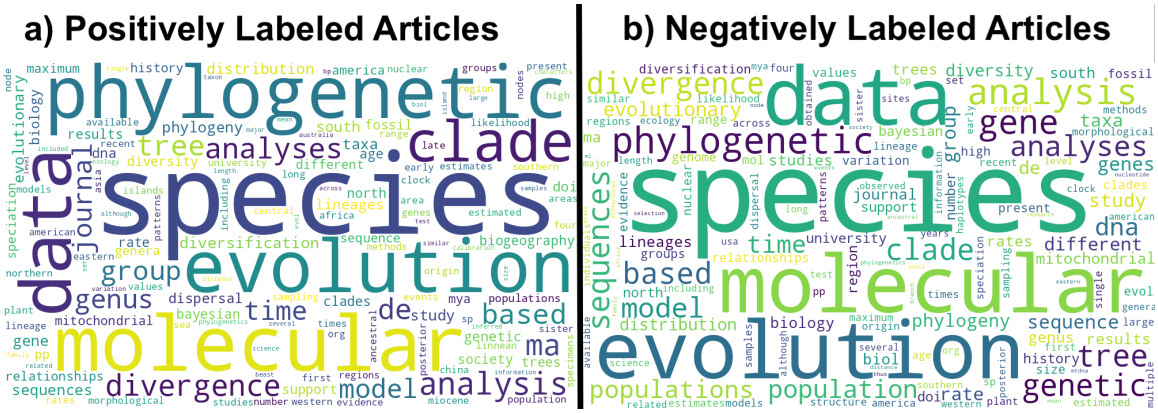


FIGURE 7.3: Comparison of the distributions of the most common words among papers labeled as a) positive and b) negative in TimeTree Dataset. There is quite a bit of overlap between the two sets.

labeled papers. The column on the right of the figure displays the words with the most significant positive and negative influences. The similarity in the distribution of words between the two labeled datasets makes classification difficult. The best model, a gradient boosting classifier over TFIDF features, achieved an F1 score of 0.72 (second row of Table 7.1).

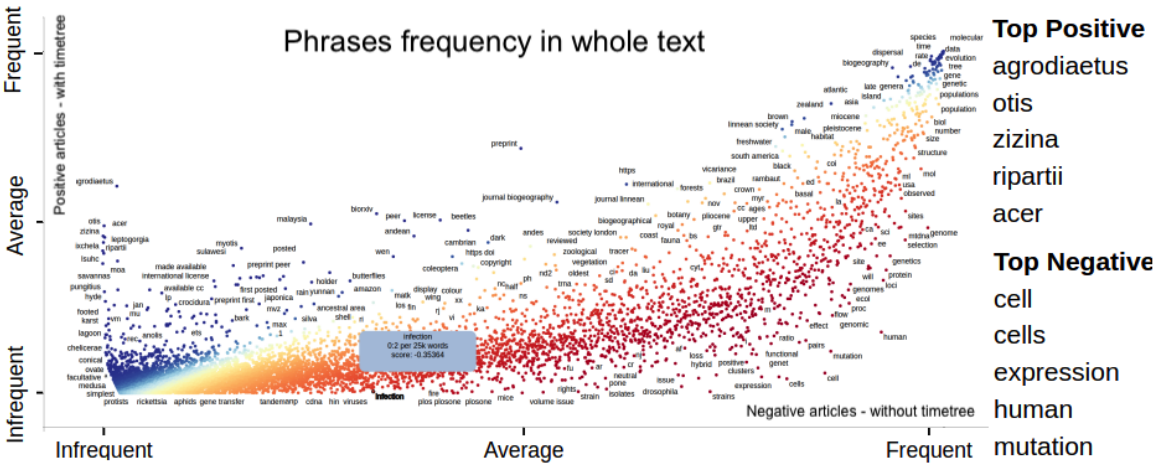


FIGURE 7.4: Scatter plot of words from articles in original datasets. The horizontal axis represents word frequency in negatively labeled documents and the vertical in positively labeled. The top positive and negative words are displayed on the right.

We employed the best-performing classifier to categorize papers in the gathered datasets, identifying positive papers to expand the TimeTree database. Our analysis of the datasets

obtained through Google Scholar using FL-simple and FL-complex queries predicted that 13% and 17.8%, respectively, contained timetrees. Furthermore, 7.8% of the research articles downloaded from relevant journals were predicted to contain timetrees. However, only a small percentage of articles from PMC (0.79%), PMC Historical (0.00039%), and bioRxiv (1.4%) received a positive label.

7.5.4 Improving predictions using text excerpts

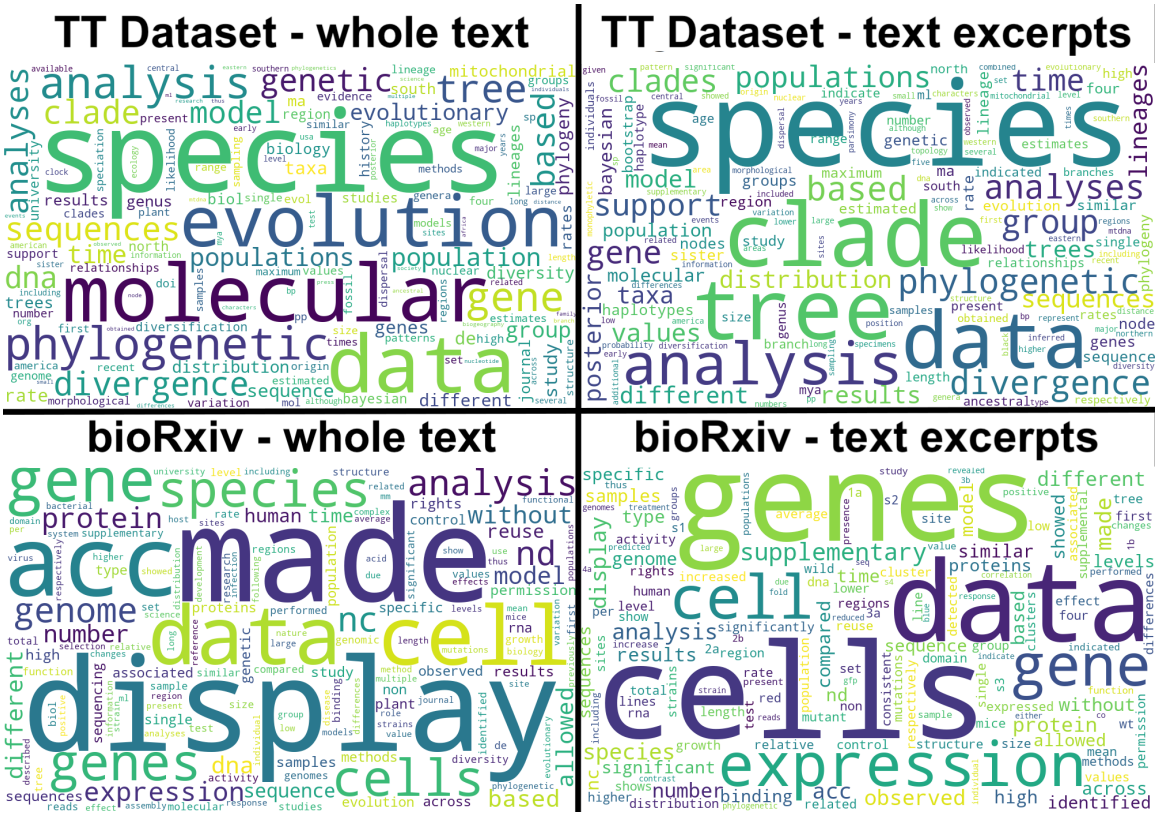


FIGURE 7.5: Comparison of distributions of most common words between whole texts (left) and selected texts (right) from two research papers datasets: original (top) and bioRxiv (bottom).

In Figure 7.5 (top), we observe that choosing only relevant text excerpts to represent articles from the TT dataset leads to a change in word distribution while keeping important words such as species, analysis, data, divergence, and tree as the most common. Additionally, keywords commonly associated with finding timetrees, such as time, trees, clade,

analysis, analyses, and distribution, are more prominent in the graph of selected text excerpts than when using the entire texts. This impact of text excerpts was more significant in the bioRxiv dataset (Figure 7.5 bottom), where many unimportant words (e.g., made, acc, display) were removed, and some relevant words (such as genes and analysis) became more prominent. Figure 7.6 has a similar focus as Figure 7.4, but it is created for the text excerpts in the TT dataset. The point distribution is different, with many more words showing a positive or negative influence, making the problem easier to solve. Furthermore, the list of top positive words in Figure 7.6 is more relevant to timetrees.

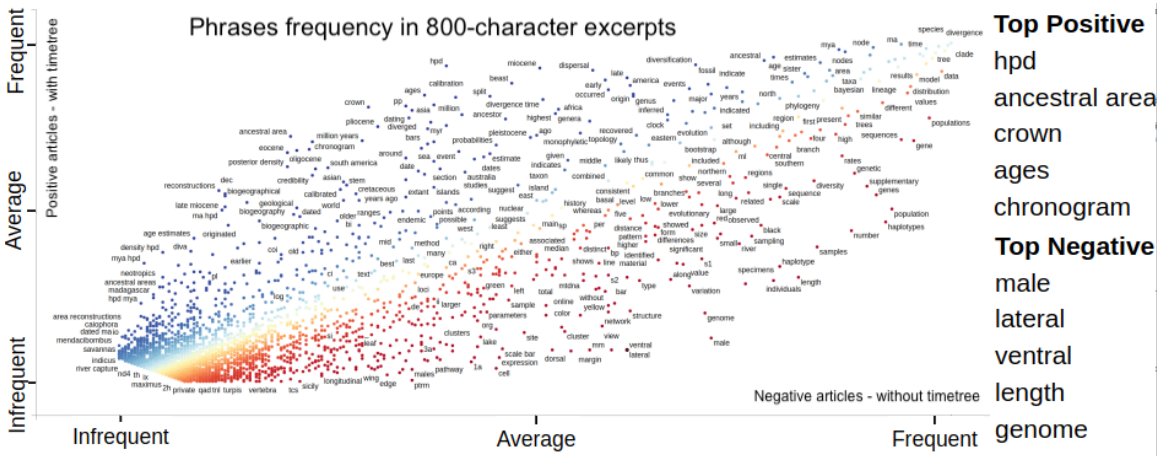


FIGURE 7.6: Scatter plot of words in selected texts from articles in original datasets. The horizontal axis repressents word frequency in negative documents and the vertical in positive. The top positive and negative words are displayed on the right.

The figure 7.7 showcases a comparison of the contexts in which the words ”mega” (left) and ”tree” (right) appear, as learned by the doc2vec algorithm. Although both words are relevant to molecular timetrees, they have distinct characteristics. ”Mega” is a unigram that refers to a widely used software name and is not expected to be used in other contexts in the corpus. On the other hand, ”tree” is a common English word that can have other uses and often appears as part of a bigram in phylogenetics, such as ”time tree” or ”molecular tree.”

We have presented the ten most similar words obtained from the learned vector space,

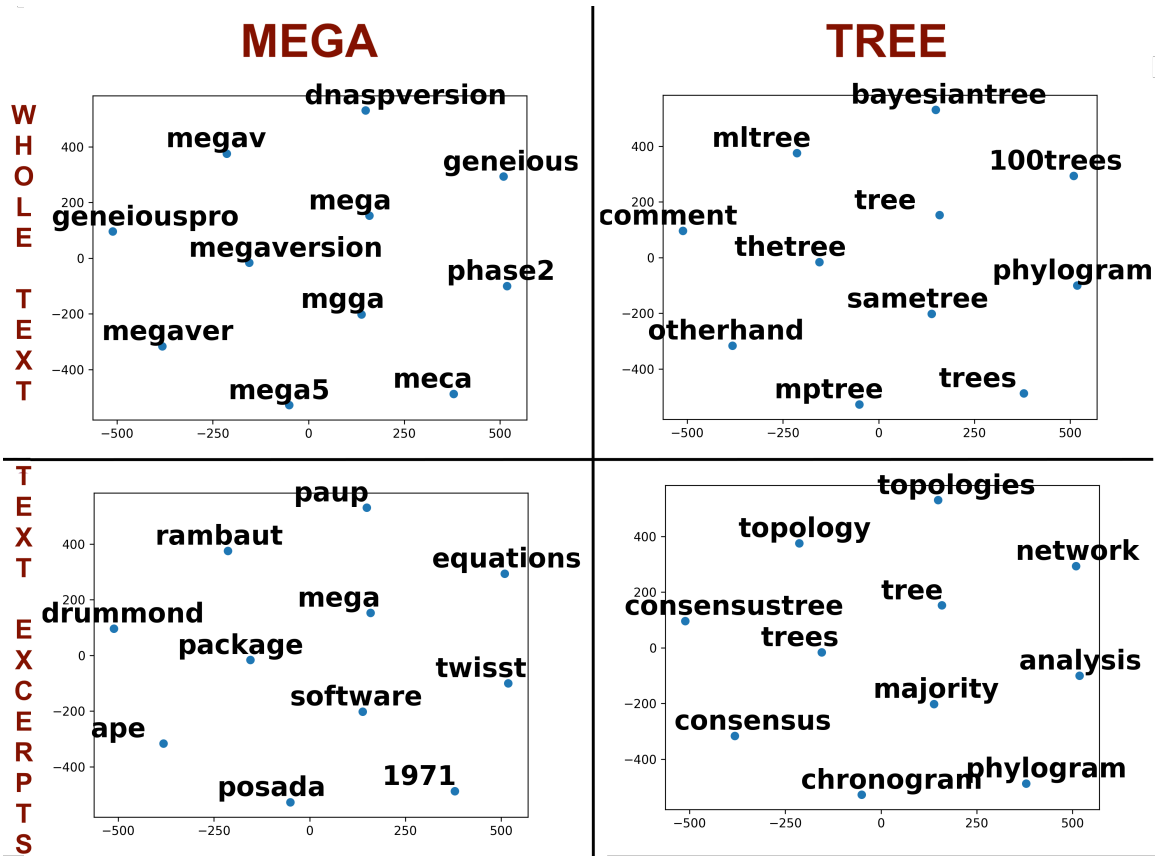


FIGURE 7.7: Comparison of projected 100-dimensional embeddings of words in contexts of words mega (left), and tree (right) learned on the whole papers (top) and on selected texts (bottom).

and it is evident that the learned vectors are meaningful. However, there is a noticeable distinction between the top and bottom graphs. The top graphs mainly consist of misspellings, whereas the bottom graphs contain words that are contextually similar. This is because the representation learned from selected text excerpts only uses relevant portions of text where misspellings or errors due to OCR or parsing are less frequent. Additionally, it is easier to learn the context from shorter selected texts. Therefore, most algorithms perform better on the excerpt dataset (as shown in Table 7.1). BERT achieved the best results for the dataset of 800-character excerpts, followed by the performance for excerpts that were 300 characters long.

After applying the best classifier to selected text excerpts from collected datasets, we

found that a higher percentage of positive text excerpts were predicted for datasets collected through Google Scholar (27.4% for FL-simple and 20.8% for FL-complex queries) and journal collections (11.3%) compared to bioRxiv (2%), PMC (2%), and PMC Historical (1.3%) sources. This is because the selected text excerpts contain informative words only from relevant papers, making it easier for the classifier to identify positive cases. However, the low percentage of positive predictions for bioRxiv and PMC sources suggests that a more complex filtering technique is needed to improve the accuracy of classification for these datasets.

7.5.5 Classification results and discussion

Table 7.1 shows the F1 score results for various combinations of techniques. The columns indicate the F1 scores for the techniques when the representation is learned only on the original dataset. Supplementary materials provide results with improved representation. We fine-tuned hyperparameters to optimize the F1 score, aiming to minimize both type I and type II errors. The best classification method for each column is shown in bold. BERT classification using the full text produced an F1 score of 0.672, which is much lower than BERT trained on selected text excerpts of length 800. The latter achieved an F1 score of 0.880 with a small standard deviation of 0.003.

The performance of representation and classification methods depends on the selection of text for representation. For whole texts, TFIDF representation performs the best, while BERT-based models have limitations and can only consider the first 512 words. In contrast, when using selected text excerpts, BERT-like models perform better. Among them, BERT is the top performer for selected text excerpts of 800 characters, while SciBERT is the best for 300-character selected text excerpts. DistilBERT has a slightly lower performance than BERT, but it is faster and more efficient. Surprisingly, BioBERT did not perform better than BERT on any of our datasets, despite its fine-tuning for biological and medical data. Similarly, although SciBERT is fine-tuned on 80% of biological scientific articles, its

	Features from labeled data only		
Method	Whole text	800-character excerpts	300-character excerpts
L1 + TFIDF	0.711 ± 0.004	0.792 ± 0.012	0.689 ± 0.009
L1 + doc2vec	0.642 ± 0.015	0.750 ± 0.013	0.751 ± 0.018
DT + TFIDF	0.634 ± 0.001	0.757 ± 0.005	0.643 ± 0.010
DT + doc2vec	0.532 ± 0.011	0.616 ± 0.004	0.654 ± 0.039
RF + TFIDF	0.695 ± 0.001	0.839 ± 0.003	0.717 ± 0.009
RF + doc2vec	0.617 ± 0.001	0.750 ± 0.003	0.750 ± 0.007
KNN + TFIDF	0.557 ± 0.019	0.760 ± 0.005	0.608 ± 0.008
KNN + doc2vec	0.582 ± 0.007	0.674 ± 0.037	0.639 ± 0.029
SVM + TFIDF	0.711 ± 0.017	0.817 ± 0.005	0.689 ± 0.001
SVM + doc2vec	0.659 ± 0.019	0.788 ± 0.004	0.754 ± 0.005
Ada + TFIDF	0.673 ± 0.027	0.783 ± 0.010	0.683 ± 0.023
Ada + doc2vec	0.589 ± 0.013	0.720 ± 0.011	0.687 ± 0.015
GB + TFIDF	$\mathbf{0.720 \pm 0.007}$	0.810 ± 0.002	0.693 ± 0.005
GB + doc2vec	0.599 ± 0.030	0.757 ± 0.023	0.729 ± 0.007
Bag + TFIDF	0.707 ± 0.010	0.812 ± 0.003	0.707 ± 0.005
Bag + doc2vec	0.547 ± 0.013	0.733 ± 0.008	0.728 ± 0.004
Distilled BERT	0.661 ± 0.009	0.873 ± 0.006	0.852 ± 0.002
BERT	0.672 ± 0.001	$\mathbf{0.880 \pm 0.003}$	0.830 ± 0.044
SciBERT	0.618 ± 0.078	0.837 ± 0.018	$\mathbf{0.864 \pm 0.002}$
BioBERT	0.657 ± 0.003	0.868 ± 0.001	0.852 ± 0.001

Table 7.1: Results from analysis. Average F1 scores and their standard are shown.

performance is 5% lower than BERT's. Nevertheless, SciBERT demonstrates remarkable performance on short text excerpts. These results are summarized in Table 7.1.

In our experiments, we did not observe a significant improvement in performance with enhanced representation, although it is generally beneficial for the doc2vec algorithm to learn from the entire text. Learning from the entire text, however, resulted in lower performance compared to learning from text excerpts. The Supplementary materials provide results that show the limited performance of enhanced representation, which may be due to the significant differences between the word distributions in the ground truth TT dataset and the target datasets. Further experiments are needed to understand this issue better. Overall, TFIDF representation outperforms doc2vec representation except when doc2vec is combined with enhanced representation on selected texts. Gradient-boosting classifi-

	Features from labeled data only		
Method	Whole text	800-character excerpts	300-character excerpts
L1 + TFIDF	0.698 ± 0.011	0.724 ± 0.013	0.622 ± 0.042
L1 + doc2vec	0.637 ± 0.010	0.727 ± 0.007	0.717 ± 0.005
DT + TFIDF	0.602 ± 0.001	0.654 ± 0.031	0.650 ± 0.001
DT + doc2vec	0.526 ± 0.023	0.663 ± 0.018	0.636 ± 0.019
RF + TFIDF	0.700 ± 0.004	0.737 ± 0.011	0.650 ± 0.001
RF + doc2vec	0.604 ± 0.040	0.764 ± 0.007	0.727 ± 0.014
KNN + TFIDF	0.606 ± 0.001	0.674 ± 0.020	0.584 ± 0.007
KNN + doc2vec	0.590 ± 0.014	0.751 ± 0.009	0.670 ± 0.001
SVM + TFIDF	0.701 ± 0.007	0.739 ± 0.002	0.621 ± 0.003
SVM + doc2vec	0.672 ± 0.003	0.774 ± 0.011	0.751 ± 0.013
Ada + TFIDF	0.700 ± 0.001	0.700 ± 0.016	0.594 ± 0.004
Ada + doc2vec	0.571 ± 0.001	0.727 ± 0.005	0.699 ± 0.003
GB + TFIDF	$\mathbf{0.700 \pm 0.020}$	0.714 ± 0.012	0.602 ± 0.021
GB + doc2vec	0.574 ± 0.028	0.763 ± 0.010	0.731 ± 0.002
Bag + TFIDF	0.681 ± 0.011	0.727 ± 0.006	0.601 ± 0.012
Bag + doc2vec	0.531 ± 0.009	0.750 ± 0.015	0.708 ± 0.003

Table 7.2: Results from analysis. Average F1 scores and their standard are shown when models are trained with enhanced representation.

cation achieved the best F1 score of 0.720 when trained on the whole text with TFIDF representation, supporting our hypothesis that TFIDF can perform better by learning from the entire document. However, we also observed that when working with shorter data (selected texts) and more data (enhanced representation), doc2vec can achieve similar results to TFIDF.

We conducted non-deep learning experiments again, incorporating the representation enhancement method described in section 7.3.4. The results are presented in Table 7.2, indicating that the additional representation learned from the downloaded data has a detrimental effect on the performance of all models. This can be attributed to the significant distributional difference between the ground truth TT dataset and the downloaded datasets, which results in additional noise rather than valuable information when learning additional representations from downloaded data.

7.5.6 *The proportion of additional timetree articles predicted*

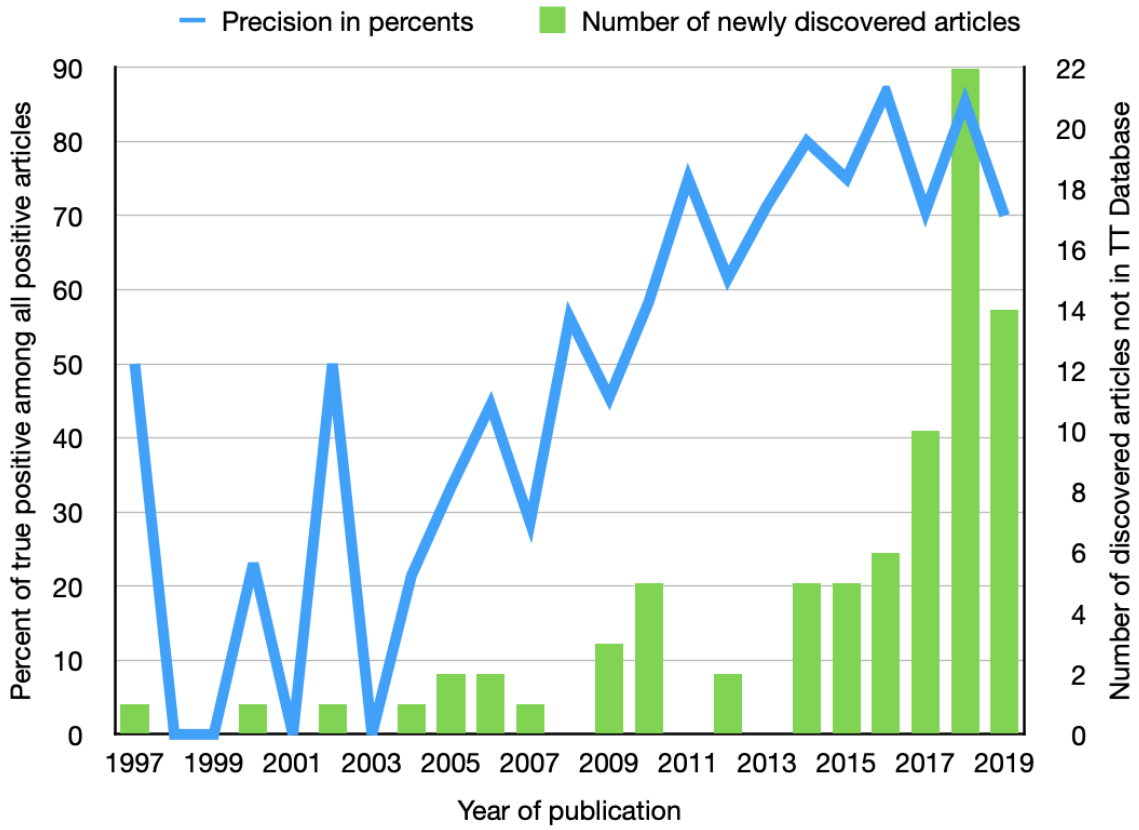


FIGURE 7.8: The precision per year of BERT, trained on 800-character excerpts and evaluated on articles from Molecular Phylogenetics and Evolution, Systematic Biology, and Zoologica Scripta, reveals that some articles containing information about timetrees are not present in the TimeTree database.

In order to evaluate the precision of the best-performing model (BERT with 800-character excerpts) on out-of-distribution data, we manually reviewed journal articles that were predicted to contain timetree information. Our assessment was limited to articles from three relevant journals (Molecular Phylogenetics and Evolution, Systematic Biology, and Zoologica Scripta) because they required manual inspection by TimeTree curators. BERT predicted that 304 articles from these journals contained timetrees, of which 163 actually did, resulting in a true positive rate of 53.6%. The left y-axis of Figure 7.8 shows the precision (true positive among all positive) per year. For recently published articles, the

TTF accuracy was 87%. However, some older articles were mislabeled as positive because they contained trees but lacked a time component. Therefore, future improvements to this approach could focus on reducing false positives for earlier articles.

Out of the correctly identified timetree-containing articles, 84 were absent from the TimeTree (TT) database, accounting for 51.5% of the total. The percentage of missing timetrees from the TT database shows a trend over time, with a greater proportion of recent articles being absent (shown by the green bars in Figure 7.8, right y-axis). This is due to the increasing number of published articles, and the significant cost of manually standardizing and incorporating them into the TT database. Therefore, the TTF system could potentially help to double the number of research articles in the TimeTree database.

7.6 Summary

In this chapter, we developed an automated system for identifying research articles containing molecular timetrees, which can save significant time and effort previously spent on manual searches. Through experimentation, we found that text-mining article excerpts surrounding the mentioning of figures achieved better results than other methods tested. The resulting TimeTreeFinder tool, TTF, achieved an F1 score of 0.88, a significant improvement over the BERT baseline, and can process millions of articles to discover timetree-containing articles automatically.

Using TTF, we estimate that we can discover twice as many timetree-containing articles as those found manually, potentially doubling the knowledge accessible to the wider community. Our analyses also revealed that many articles containing timetrees are missing from the TimeTree database. By increasing the number of species covered and the number of studies per species in the database, TTF can make the TimeTree resource more comprehensive.

Furthermore, the TTF tool can be adapted to search for other phylogenetic information in scientific literature and other relevant sources. With a small labeled dataset and positive

and negative labels, TTF can extract the most important phrases for finding relevant documents, search relevant sources based on those phrases, and make predictions. The TTF tool can significantly speed up the collection and curation of timetrees with much lower human and time costs.

CHAPTER 8

CONCLUSION

This thesis explores the use of machine learning domain adaptation methods for producing representations and predictions for natural language processing (NLP), medical, and biological applications. Domain adaptation is the process of transferring knowledge across domains and tasks, and it's superior to supervised learning, except in cases where the target domain has a sufficient number of good quality-labeled training examples available or when there is no relevant data source that can be utilized to improve learning on the target data. Neural networks (NNs) are used in most cases for domain adaptation, as they perform better than models that do not utilize related information from other domains and tasks.

The thesis proposes six objectives that are addressed by the approaches presented in the previous chapters, including optimizing source data modeling, analyzing the effects of unexpected bias on domain adaptation, investigating the impact of imbalanced class distribution, enhancing domain adaptation performance on datasets with a high rate of missing data, understanding the influence of multiple data sources in domain adaptation, and alleviating the curse of dimensionality in domain adaptation. The approaches are evaluated on NLP tasks, the US elections, and the youth anxiety clinical dataset, among others. The thesis provides insights and methods for improving the accuracy of predictive models in

real-world applications.

Chapter 2 has provided essential background knowledge in machine learning required to comprehend the subsequent chapters. The neural network methods, natural language processing tasks, and models used throughout this thesis have also been introduced. Furthermore, this chapter has presented an overview of the role of machine learning in various fields, including NLP, medicine, healthcare, biology, and bioinformatics.

In chapter 3, we introduced domain adaptation and discussed many techniques developed to improve the performance on the target task in an unsupervised or semi-supervised manner. Various techniques have been developed, including representation-based methods, data weighting and selection techniques, self-labeling techniques, combining multiple source domains, and adapters to neural networks. Recently, the integration of distributional similarity approaches with latent feature learning techniques has been the most impactful trend in the field. Despite progress in domain adaptation, there is still much research to be done to improve the ability of models to generalize to out-of-distribution examples. The next chapters proposed different techniques to mitigate the domain adaptation challenges in real-world applications and to measure the quality of domain adaptation given various settings often occurring in real-world scenarios.

Chapter 4 proposed a semi-automatic labeling approach for surveying public opinion from large domain-specific textual data in social media. The approach is based on a two-part framework that uses a small subset of posts to label the rest of the data in a semi-supervised manner. The proposed approach outperformed alternative algorithms by capturing additional contexts from the unlabeled data. It was evaluated on four datasets and achieved better performance than existing models with 96.5% accuracy and 0.68 F1 score in labeling Twitter posts related to gun control/rights advocacy. The study highlighted the importance of the choice of training component based on the dataset characteristics. Overall, the proposed approach is an efficient and effective method for surveying public opinion from large textual data in social media.

Chapter 5 emphasized the significance of considering bias, noise, and text properties while constructing data for fine-tuning language models. Experiments with ULMFiT on Twitter and news data revealed that characteristics such as bias, vocabulary diversity, and writing style affect the final classification results more than data size. The study also found that the algorithm is not robust on unbalanced Twitter datasets and exaggerates predictions for the most frequent label. The study suggested that social scientists must consider the properties of their training and fine-tuning datasets to improve model accuracy and robustness, and future research should focus on constructing bias-free datasets for fine-tuning.

Chapter 6 discussed how to address high-missing rates in both source and target datasets for domain adaptation, using youth anxiety clinical datasets. Machine learning methods for imputation and prediction can greatly improve the understanding of treatments in such studies. The proposed iterative imputation with a bag of elastic net regressions method provides the best results. Leveraging combination of datasets from different studies is significant in successful prediction, especially when the target variable is from a different domain, the domain of withdrawn patients. The study highlighted the importance of building and evaluating machine-learning tools for psychological and other longitudinal medical research studies but noted the need to evaluate potential ethical challenges around imputation. Overall, the application of machine learning methods can provide valuable insights into the treatment of youth anxiety and related mental health disorders.

Chapter 7 presented the development of an automated system called TimeTreeFinder (TTF), for identifying research articles that contain molecular timetrees. The TTF tool used text-mining to search article excerpts surrounding figure mentions and achieved an F1 score of 0.88, significantly better than the BERT baseline. Using TTF, twice as many timetree-containing articles can be discovered compared to manual searches. The TTF tool can also be adapted to search for other phylogenetic information in scientific literature and other relevant sources, making the collection and curation of timetrees more efficient with lower human and time costs.

Throughout this thesis, we have extensively reviewed the existing literature. However, it is noteworthy that many current approaches replace classic feature-based approaches with different tools without considering the lessons learned from past approaches. While we also draw inspiration from classic approaches, we explicitly combine them with current methods. We aimed to ensure that our models perform well on in-domain data and data outside the training distribution. Hence, we evaluated them on multiple domains, including product reviews, news, Twitter, topic classification, research papers, treatment outcomes, and more.

8.1 Future Directions

Domain adaptation is a key challenge in machine learning, particularly in cases where the distribution of the training data is different from that of the test data. In recent years, there has been significant progress in domain adaptation research, with a variety of techniques and algorithms proposed to address this challenge. However, there is still much work to be done in this area, and future research is likely to focus on a number of key directions.

One of the most promising directions for future research is the development of deep-learning approaches to domain adaptation. Deep learning has proven highly effective in a range of machine learning tasks, and its ability to learn high-level representations of data in pre-trained and multi-task manners are particularly valuable in the context of domain adaptation. There is already some work in this area, with deep neural networks being used to learn domain-invariant features that can be used across multiple domains. Also, adapters to deep neural networks have been developed to learn the adaptation to a new domain with very few new parameters. New domain adaptation approaches are being developed, but some of the most promising works are:

Domain Adversarial Neural Network (DANN) introduces a domain classifier into a neural network. This classifier is trained to predict the domain of the input data, while the main network is trained to minimize both the task loss and the domain loss. By doing so,

the network learns to extract domain-invariant features that can be applied to new domains.

Adversarial Discriminative Domain Adaptation (ADDA) uses a domain discriminator to align the feature distributions of the source and target domains. The discriminator is trained to distinguish between source and target features, while the main network is trained to minimize task loss and confuse the discriminator.

Deep Adaptation Network (DAN) aligns the mean and covariance of the feature distributions of the source and target domains. The algorithm uses a deep neural network to learn a feature representation that is both discriminative for the task and domain-invariant.

Joint Adaptation Network (JAN) is a variant of DAN that learns multiple domain-invariant feature representations for different domains. The algorithm uses a joint maximum mean discrepancy (JMMD) loss to align the feature distributions of the different domains and a classification loss to perform the main task.

Transfer Learning Network (TLN) uses pre-trained networks on a source domain to perform a target domain task. The pre-trained network is fine-tuned on the target domain, and a domain classifier is used to measure the domain discrepancy between the source and target domains. By doing so, the network can leverage the knowledge learned from the source domain to improve performance on the target domain.

Another key direction for future research is the development of new techniques for unsupervised domain adaptation. In many cases, labeled data may not be available in the target domain, making supervised approaches unfeasible. Unsupervised domain adaptation techniques aim to overcome this challenge by leveraging the information contained in both the source and target domains to learn domain-invariant representations of the data. There has already been some progress in this area, with techniques such as adversarial training and maximum mean discrepancy used to learn such representations. Besides above mentioned DAAN and DAN algorithms, here are other promising directions for unsupervised domain adaptation.

Deep Reconstruction-Classification Networks (DRCN) learns a joint reconstruction-

classification model for both source and target domains. DRCN trains a deep neural network to reconstruct the inputs and then performs classification on the reconstructed features. The reconstruction step helps to learn domain-invariant features, while the classification step learns to predict the labels of the source domain.

Maximum Mean Discrepancy (MMD) is a non-parametric kernel-based method that measures the distance between two probability distributions. MMD can be used to align the distributions of the source and target domains by minimizing their discrepancy.

Autoencoder-based Domain Adaptation (ADA) uses an autoencoder to learn domain-invariant representations. The autoencoder is trained to reconstruct the inputs from both the source and target domains. The encoder network is then used as a feature extractor, and the weights are fine-tuned to minimize the discrepancy between the source and target domain features.

While improving the models is important, collecting better datasets and understanding settings and data properties that lead to the best domain adaptation are even more crucial to real-world applications. There are several important future directions of data handling for domain adaptation, including:

- **Incorporating more diverse and complex data**: Future domain adaptation methods should be able to handle more complex and diverse data types, including text, images, and video, among others. This will require the development of new techniques that can handle these types of data and adapt to their unique characteristics.

- **Addressing the issue of class imbalance**: Imbalanced datasets are common in many real-world scenarios, which can lead to poor performance of domain adaptation methods. Future research should focus on developing methods that can handle class imbalance and improve domain adaptation performance in such scenarios.

- **Dealing with noisy and incomplete data**: Many real-world datasets are noisy or incomplete, which can make it difficult to train accurate domain adaptation models.

Future research should develop methods to handle and extract useful information from such data.

- **Addressing the ethical concerns of domain adaptation**: There are ethical concerns associated with domain adaptation, such as bias and privacy issues. Future research should address these concerns and develop fair, unbiased methods that protect individuals' privacy.

- **Developing more efficient algorithms**: Domain adaptation can be computationally expensive, which can limit its practical application. Future research should focus on developing more efficient algorithms that can handle large datasets and complex models.

BIBLIOGRAPHY

Alshehri, J., Stanojevic, M., Khan, P., Rapp, B., Dragut, E., and Obradovic, Z. (2022), "MultiLayerET: A Unified Representation of Entites and Topics Using Multilayer Graphs," in *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*.

Andjelkovic, J., Ljubic, B., Hai, A. A., Stanojevic, M., Pavlovski, M., Diaz, W., and Obradovic, Z. (2022), "Sequential machine learning in prediction of common cancers," *Informatics in Medicine Unlocked*, 30, 100928.

Appel, O., Chiclana, F., Carter, J., and Fujita, H. (2016), "A hybrid approach to the sentiment analysis problem at the sentence level," *Knowledge-Based Systems*, 108, 110–124.

Atanasova, P., Nakov, P., Màrquez, L., Barrón-Cedeño, A., Karadzhov, G., Mihaylova, T., Mohtarami, M., and Glass, J. (2019), "Automatic fact-checking using context and discourse information," *Journal of Data and Information Quality (JDIQ)*, 11, 1–27.

Augsburger, M. and Galatzer-Levy, I. R. (2020), "Utilization of machine learning to test the impact of cognitive processing and emotion recognition on the development of PTSD following trauma exposure," *BMC psychiatry*, 20, 1–11.

Baik, S., Choi, J., Kim, H., Cho, D., Min, J., and Lee, K. M. (2021), "Meta-learning with task-adaptive loss function for few-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9465–9474.

Baroni, M., Dinu, G., and Kruszewski, G. (2014), "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 238–247.

Bastos, M. and Mercea, D. (2018), "Parametrizing Brexit: mapping Twitter political space to parliamentary constituencies," *Information, Communication & Society*, 21, 921–939.

Bell, M. L. and Fairclough, D. L. (2014), "Practical and statistical issues in missing data for longitudinal patient-reported outcomes," *Statistical methods in medical research*, 23, 440–459.

Beltagy, I., Lo, K., and Cohan, A. (2019), "SciBERT: A Pretrained Language Model for Scientific Text," 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. 3615–3620.

Bhasuran, B. and Natarajan, J. (2018), "Automatic extraction of gene-disease associations from literature using joint ensemble learning," *PloS one*, 13, e0200699.

Bishop, C. M. (2006), *Pattern recognition and machine learning*, vol. 4, Springer.

Bitsko, R. H., Holbrook, J. R., Ghandour, R. M., Blumberg, S. J., Visser, S. N., Perou, R., and Walkup, J. T. (2018), "Epidemiology and impact of health care provider–diagnosed anxiety and depression among US children," *Journal of developmental and behavioral pediatrics: JDBP*, 39, 395.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003), "Latent dirichlet allocation," *Journal of machine Learning research*, 3, 993–1022.

Blitzer, J., McDonald, R., and Pereira, F. (2006), "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128.

Blum, A. and Mitchell, T. (1998), "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, ACM.

Bodden, D. H., Bögels, S. M., Nauta, M. H., De Haan, E., Ringrose, J., Appelboom, C., Brinkman, A. G., and Appelboom-Geerts, K. C. (2008), "Child versus family cognitive-behavioral therapy in clinically anxious youth: An efficacy and partial effectiveness study," *Journal of the American Academy of Child & Adolescent Psychiatry*, 47, 1384–1394.

Brown, P. F., Della Pietra, V. J., Desouza, P. V., Lai, J. C., and Mercer, R. L. (1992), "Class-based n-gram models of natural language," *Computational linguistics*, 18, 467–480.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020), "Language models are few-shot learners," *Advances in neural information processing systems*, 33, 1877–1901.

Buntain, C., McGrath, E., and Behlendorf, B. (2018), "Sampling Social Media: Supporting Information Retrieval from Microblog Data Resellers with Text, Network, and Spatial Analysis," in *Proceedings of the 51st Hawaii International Conference on System Sciences*.

Caruana, R. (1998), *Multitask learning*, Springer.

Celli, F., Stepanov, E., Poesio, M., and Riccardi, G. (2016), "Predicting Brexit: Classifying agreement is better than sentiment and pollsters," in *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pp. 110–118.

Chen, T., Xu, R., He, Y., and Wang, X. (2017), "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN," *Expert Systems with Applications*, 72, 221–230.

Chopra, S., Hadsell, R., and LeCun, Y. (2005), "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, IEEE.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020), "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*.

Collobert, R. and Weston, J. (2008), "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167.

Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016), "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017), "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv:1705.02364*.

Costello, E. J., Mustillo, S., Erkanli, A., Keeler, G., and Angold, A. (2003), "Prevalence and development of psychiatric disorders in childhood and adolescence," *Archives of general psychiatry*, 60, 837–844.

Crawshaw, M. (2020), "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*.

Dai, A. M. and Le, Q. V. (2015), "Semi-supervised sequence learning," *Advances in neural information processing systems*, 28.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990), "Indexing by latent semantic analysis," *Journal of the American society for information science*, 41, 391–407.

DeSouza, C. M., Legedza, A. T., and Sankoh, A. J. (2009), "An overview of practical approaches for handling missing data in clinical trials," *Journal of biopharmaceutical statistics*, 19, 1055–1073.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019), "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, Association for Computational Linguistics.

Dharmadhikari, S., Ingle, M., and Kulkarni, P. (2012), "Analysis of semi supervised learning methods towards multi label text classification," *International Journal of Computer Applications*.

Diep, B., Stanojevic, M., and Novikova, J. (2022), "Multi-modal deep learning system for depression and anxiety detection," in *Empowering Communities: A Participatory Approach to AI for Mental Health*.

Elman, J. L. (1990), "Finding structure in time," *Cognitive science*, 14, 179–211.

Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B., and Tabona, O. (2021), "A survey on missing data in machine learning," *Journal of Big Data*, 8, 1–37.

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010), "Why does unsupervised pre-training help deep learning?" in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 201–208, JMLR Workshop and Conference Proceedings.

Essau, C. A., Lewinsohn, P. M., Olaya, B., and Seeley, J. R. (2014), "Anxiety disorders in adolescents and psychosocial outcomes at age 30," *Journal of affective disorders*, 163, 125–132.

Finn, C. and Levine, S. (2017), "Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm," in *International Conference on Learning Representations*.

Fouad, K. M., Ismail, M. M., Azar, A. T., and Arafa, M. M. (2021), "Advanced methods for missing values imputation based on similarity learning," *PeerJ Computer Science*, 7, e619.

Fukushima, K. (1975), "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, 20, 121–136.

Fukushima, K. (1980), "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, 36, 193–202.

García-Pablos, A., Cuadros, M., and Rigau, G. (2018), "W2VLDA: almost unsupervised system for aspect based sentiment analysis," *Expert Systems with Applications*, 91, 127–137.

Glynn, C. J. (2018), *Public opinion*, Routledge.

Goodfellow, I., Bengio, Y., and Courville, A. (2016), *Deep learning*, MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020), "Generative adversarial networks," *Communications of the ACM*, 63, 139–144.

Gradus, J. L., King, M. W., Galatzer-Levy, I., and Street, A. E. (2017), "Gender differences in machine learning models of trauma and suicidal ideation in veterans of the Iraq and Afghanistan Wars," *Journal of traumatic stress*, 30, 362–371.

Gradus, J. L., Rosellini, A. J., Horváth-Puhó, E., Jiang, T., Street, A. E., Galatzer-Levy, I., Lash, T. L., and Sørensen, H. T. (2021), "Predicting sex-specific nonfatal suicide attempt risk using machine learning and data from Danish national registries," *American journal of epidemiology*, 190, 2517–2527.

Grave, E., Mikolov, T., Joulin, A., and Bojanowski, P. (2017), "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pp. 3–7.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013), "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, Ieee.

He, K., Zhang, X., Ren, S., and Sun, J. (2016), "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hedges, S. B., Dudley, J., and Kumar, S. (2006), "TimeTree: a public knowledge-base of divergence times among organisms," *Bioinformatics*, 22, 2971–2972.

Hill, F., Cho, K., and Korhonen, A. (2016), "Learning distributed representations of sentences from unlabelled data," *arXiv preprint arXiv:1602.03483*.

Hochreiter, S. and Schmidhuber, J. (1997), "Long short-term memory," *Neural computation*, 9, 1735–1780.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019), "Parameter-efficient transfer learning for NLP," in *International Conference on Machine Learning*, pp. 2790–2799, PMLR.

Howard, J. and Ruder, S. (2018), "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*.

Howard, P. N., Bolsover, G., Kollanyi, B., Bradshaw, S., and Neudert, L.-M. (2017), "Junk news and bots during the US election: What were Michigan voters sharing over Twitter," *Computational Propaganda Research Project, Oxford Internet Institute, Data Memo*, 1.

Huang, F. and Yates, A. (2009), "Distributional representations for handling sparsity in supervised sequence-labeling," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 495–503.

Java, A., Song, X., Finin, T., and Tseng, B. (2007), "Why we twitter: understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pp. 56–65, ACM.

Jernite, Y., Bowman, S. R., and Sontag, D. (2017), "Discourse-based objectives for fast unsupervised sentence representation learning," *arXiv preprint arXiv:1705.00557*.

Jones, K. S. (1972), "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*.

Jurafsky, D. and Martin, J. H. (2023), *Speech and Language Processing*, https://web.stanford.edu/ jurafsky/slp3/ed3book$_{jan72023.pdf}$.

Kendall, P. C. (1994), "Treating anxiety disorders in children: results of a randomized clinical trial." *Journal of consulting and clinical psychology*, 62, 100.

Kendall, P. C., Flannery-Schroeder, E., Panichelli-Mindel, S. M., Southam-Gerow, M., Henin, A., and Warman, M. (1997), "Therapy for youths with anxiety disorders: A second randomized clincal trial." *Journal of consulting and clinical psychology*, 65, 366.

Kendall, P. C., Hudson, J. L., Gosch, E., Flannery-Schroeder, E., and Suveg, C. (2008), "Cognitive-behavioral therapy for anxiety disordered youth: a randomized clinical trial evaluating child and family modalities." *Journal of consulting and clinical psychology*, 76, 282.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015), "Skip-thought vectors," *Advances in neural information processing systems*, 28.

Koesmahargyo, V., Abbas, A., Zhang, L., Guan, L., Feng, S., Yadav, V., and Galatzer-Levy, I. R. (2020), "Accuracy of machine learning-based prediction of medication adherence in clinical research," *Psychiatry Research*, 294, 113558.

Kumar, S., Suleski, M., Craig, J. M., Kasprowicz, A. E., Sanderford, M., Li, M., Stecher, G., and Hedges, S. B. (2022), "TimeTree 5: An expanded resource for species divergence times," *Molecular Biology and Evolution*, 39, msac174.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019), "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*.

Le, Q. and Mikolov, T. (2014), "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, pp. 1188–1196.

Le, Q. V. (2013), "Building high-level features using large scale unsupervised learning," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8595–8598, IEEE.

Lebowitz, E. R., Zilcha-Mano, S., Orbach, M., Shimshoni, Y., and Silverman, W. K. (2021), "Moderators of response to child-based and parent-based child anxiety treatment: a machine learning-based analysis," *Journal of Child Psychology and Psychiatry*, 62, 1175–1182.

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

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020), "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, 36, 1234–1240.

Levy, O., Goldberg, Y., and Dagan, I. (2015), "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the association for computational linguistics*, 3, 211–225.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019), "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*.

Lin, T. H. (2010), "A comparison of multiple imputation with EM algorithm and MCMC method for quality of life missing data," *Quality & quantity*, 44, 277–287.

Liu, B., Li, X., Lee, W. S., and Yu, P. S. (2004), "Text classification by labeling words," in *AAAI*, vol. 4, pp. 425–430.

Liu, P., Qiu, X., and Huang, X. (2016), "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019), "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*.

Ljubic, B., Hai, A. A., Stanojevic, M., Diaz, W., Polimac, D., Pavlovski, M., and Obradovic, Z. (2020), "Predicting complications of diabetes mellitus using advanced machine learning algorithms," *Journal of the American Medical Informatics Association*, 27, 1343–1351.

Llewellyn, C., Cram, L., and Favero, A. (2016), "Avoiding the Drunkard's search: Investigating collection strategies for building a Twitter dataset," in *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 205–206, IEEE.

Luhn, H. P. (1957), "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of research and development*, 1, 309–317.

Mainzer, R., Apajee, J., Nguyen, C. D., Carlin, J. B., and Lee, K. J. (2021), "A comparison of multiple imputation strategies for handling missing data in multi-item scales: Guidance for longitudinal studies," *Statistics in Medicine*, 40, 4660–4674.

Manning, C. and Schutze, H. (1999), *Foundations of statistical natural language processing*, MIT press.

Manning, C. D. (2008), *Introduction to information retrieval*, Syngress Publishing.

Mao, H. H. (2020), "A survey on self-supervised pre-training for sequential transfer learning in neural networks," *arXiv preprint arXiv:2007.00800*.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017), "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems*, pp. 6294–6305.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018), "The natural language decathlon: Multitask learning as question answering," *arXiv preprint arXiv:1806.08730*.

Merikangas, K. R., He, J.-p., Burstein, M., Swanson, S. A., Avenevoli, S., Cui, L., Benjet, C., Georgiades, K., and Swendsen, J. (2010), "Lifetime prevalence of mental disorders in US adolescents: results from the National Comorbidity Survey Replication–Adolescent Supplement (NCS-A)," *Journal of the American Academy of Child & Adolescent Psychiatry*, 49, 980–989.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016), "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*.

Merity, S., Keskar, N. S., and Socher, R. (2017), "Regularizing and optimizing LSTM language models," *arXiv preprint arXiv:1708.02182*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a), "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, 26.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b), "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018), "Advances in Pre-Training Distributed Word Representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009), "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011.

Molenberghs, G., Thijs, H., Jansen, I., Beunckens, C., Kenward, M. G., Mallinckrodt, C., and Carroll, R. J. (2004), "Analyzing incomplete longitudinal clinical trial data," *Biostatistics*, 5, 445–464.

Mostafa, M. M. (2013), "More than words: Social networks text mining for consumer brand sentiments," *Expert Systems with Applications*, 40, 4241–4251.

Müller, H.-M., Van Auken, K. M., Li, Y., and Sternberg, P. W. (2018), "Textpresso Central: a customizable platform for searching, text mining, viewing, and curating biomedical literature," *BMC bioinformatics*, 19, 1–16.

Munkhdalai, T. and Yu, H. (2017), "Meta networks," in *International conference on machine learning*, pp. 2554–2563, PMLR.

Nauta, M. H., Scholing, A., Emmelkamp, P. M., and Minderaa, R. B. (2003), "Cognitive-behavioral therapy for children with anxiety disorders in a clinical setting: No additional effect of a cognitive parent training," *Journal of the American Academy of Child & Adolescent Psychiatry*, 42, 1270–1278.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000), "Text classification from labeled and unlabeled documents using EM," *Machine learning*, pp. 103–134.

Otte, E. and Rousseau, R. (2002), "Social network analysis: a powerful strategy, also for the information sciences," *Journal of information Science*, pp. 441–453.

Pak, A. and Paroubek, P. (2010), "Twitter as a corpus for sentiment analysis and opinion mining." in *LREc*, vol. 10, pp. 1320–1326.

Pan, S. J. and Yang, Q. (2010), "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, 22, 1345–1359.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019), "Continual lifelong learning with neural networks: A review," *Neural networks*, 113, 54–71.

Pennington, J., Socher, R., and Manning, C. D. (2014), "GloVe: Global Vectors for Word Representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Percha, B., Garten, Y., and Altman, R. B. (2012), "Discovery and explanation of drug-drug interactions via text mining," in *Biocomputing 2012*, pp. 410–421, World Scientific.

Perera, N., Dehmer, M., and Emmert-Streib, F. (2020), "Named entity recognition and relation detection for biomedical information extraction," *Frontiers in cell and developmental biology*, p. 673.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018), "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*.

Petrazzini, B. O., Naya, H., Lopez-Bello, F., Vazquez, G., and Spangenberg, L. (2021), "Evaluation of different approaches for missing data imputation on features associated to genomic data," *BioData mining*, 14, 1–13.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018), "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020), "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, 21, 5485–5551.

Rangrej, A., Kulkarni, S., and Tendulkar, A. V. (2011), "Comparative study of clustering techniques for short text documents," in *Proceedings of the 20th international conference companion on World wide web*, pp. 111–112, ACM.

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. (2016), "Data programming: Creating large training sets, quickly," *Advances in neural information processing systems*, 29.

Ruder, S. (2019), "Neural transfer learning for natural language processing," Ph.D. thesis, NUI Galway.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986), "Learning representations by back-propagating errors," *nature*, 323, 533–536.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019), "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*.

Schmitt, P., Mandel, J., and Guedj, M. (2015), "A comparison of six methods for missing data imputation," *Journal of Biometrics & Biostatistics*, 6, 1.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015), "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

Schultebraucks, K., Sijbrandij, M., Galatzer-Levy, I., Mouthaan, J., Olff, M., and van Zuiden, M. (2021), "Forecasting individual risk for long-term posttraumatic stress disorder in emergency medical settings using biomedical data: a machine learning multicenter cohort study," *Neurobiology of stress*, 14, 100297.

Shah, D. J., Lei, T., Moschitti, A., Romeo, S., and Nakov, P. (2018), "Adversarial domain adaptation for duplicate question detection," *arXiv preprint arXiv:1809.02255*.

Silver, D. L., Yang, Q., and Li, L. (2013), "Lifelong machine learning systems: Beyond learning algorithms," in *2013 AAAI spring symposium series*.

Silverman, W. K., Kurtines, W. M., Jaccard, J., and Pina, A. A. (2009), "Directionality of change in youth anxiety treatment involving parents: an initial examination." *Journal of consulting and clinical psychology*, 77, 474.

Simon, H. A. (1954), "Spurious correlation: A causal interpretation," *Journal of the American statistical Association*, 49, 467–479.

Sindhwani, V. and Melville, P. (2008), "Document-word co-regularization for semi-supervised sentiment analysis," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 1025–1030, IEEE.

Smith, L. N. (2017), "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472, IEEE.

Snell, J., Swersky, K., and Zemel, R. (2017), "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, 30.

Sobkowicz, P., Kaschesky, M., and Bouchard, G. (2012), "Opinion mining in social media: Modeling, simulating, and forecasting political opinions in the web," *Government Information Quarterly*, 29, 470–479.

Stanojevic, M., Norris, L. A., Kendall, P. C., and Obradovic, Z. (2022), "Predicting Anxiety Treatment Outcomes with Machine Learning," in *The 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

Stanojevic, M., Andjelkovic, J., Kasprowicz, A., Huuki, L. A., Chao, J., Hedges, S. B., Kumar, S., and Obradovic, Z. (2023), "Discovering research articles containing evolutionary timetrees by machine learning," *Bioinformatics*.

Steinbach, M., Karypis, G., Kumar, V., et al. (2000), "A comparison of document clustering techniques," in *KDD workshop on text mining*.

Swan, A. J. and Kendall, P. C. (2016), "Fear and missing out: Youth anxiety and functional outcomes." *Clinical Psychology: Science and Practice*, 23, 417.

Thompson, B., Khayrallah, H., Anastasopoulos, A., McCarthy, A., Duh, K., Marvin, R., McNamee, P., Gwinnup, J., Anderson, T., and Koehn, P. (2018), "Freezing Subnetworks to Analyze Domain Adaptation in Neural Machine Translation," *arXiv preprint arXiv:1809.05218*.

Tibshirani, R. (1996), "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, 58, 267–288.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014), "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*.

Van de Ven, G. M. and Tolias, A. S. (2019), "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*.

Van Engelen, J. E. and Hoos, H. H. (2020), "A survey on semi-supervised learning," *Machine learning*, 109, 373–440.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017), "Attention is all you need," *Advances in neural information processing systems*, 30.

Villabø, M. A., Narayanan, M., Compton, S. N., Kendall, P. C., and Neumer, S.-P. (2018), "Cognitive–behavioral therapy for youth anxiety: An effectiveness evaluation in community practice." *Journal of consulting and clinical psychology*, 86, 751.

Walkup, J. T., Albano, A. M., Piacentini, J., Birmaher, B., Compton, S. N., Sherrill, J. T., Ginsburg, G. S., Rynn, M. A., McCracken, J., Waslick, B., et al. (2008), "Cognitive behavioral therapy, sertraline, or a combination in childhood anxiety," *New England Journal of Medicine*, 359, 2753–2766.

Watts, D. J. and Dodds, P. S. (2007), "Influentials, networks, and public opinion formation," *Journal of consumer research*, pp. 441–458.

Werbos, P. J. (1990), "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, 78, 1550–1560.

Whittington, C., Feinman, T., Zelman Lewis, S., Lieberman, G., and del Aguila, M. (2019), "Clinical practice guidelines: Machine learning and natural language processing for automating the rapid identification and annotation of new evidence." .

WHO (2021), "Adolescent mental health," *https://www.who.int/news-room/fact-sheets/detail/adolescent-mental-health*.

Wolpert, D. H. and Macready, W. G. (1997), "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, 1, 67–82.

Wood, J. J., Piacentini, J. C., Southam-Gerow, M., Chu, B. C., and Sigman, M. (2006), "Family cognitive behavioral therapy for child anxiety disorders," *Journal of the American Academy of Child & Adolescent Psychiatry*, 45, 314–321.

Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., and Weston, J. (2017), "Starspace: Embed all the things!" *arXiv preprint arXiv:1709.03856*.

Wu, Y., Liu, L., Bae, J., Chow, K.-H., Iyengar, A., Pu, C., Wei, W., Yu, L., and Zhang, Q. (2019), "Demystifying learning rate policies for high accuracy training of deep neural networks," in *2019 IEEE International conference on big data (Big Data)*, pp. 1971–1980, IEEE.

Xiao, Y. and Cho, K. (2016), "Efficient character-level document classification by combining convolution and recurrent layers," *arXiv preprint arXiv:1602.00367*.

Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., and Hao, H. (2015), "Short Text Clustering via Convolutional Neural Networks." in *VS@ HLT-NAACL*, pp. 62–69.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (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*, pp. 1480–1489.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019), "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, 32.

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

Yin, J. and Wang, J. (2014), "A dirichlet multinomial mixture model-based approach for short text clustering," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 233–242, ACM.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. (2018), "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722.

Zhang, L., Wang, S., and Liu, B. (2018a), "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8, e1253.

Zhang, S., Wei, Z., Wang, Y., and Liao, T. (2018b), "Sentiment analysis of Chinese microblog text based on extended sentiment dictionary," *Future Generation Computer Systems*, 81, 395–403.

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

Zhao, J., Liu, K., and Xu, L. (2016), "Sentiment analysis: mining opinions, sentiments, and emotions," .

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016), "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*.

Zhu, X. J. (2005), "Semi-supervised learning literature survey," .

Zoph, B., Yuret, D., May, J., and Knight, K. (2016), "Transfer learning for low-resource neural machine translation," *arXiv preprint arXiv:1604.02201*.